Genetic Programming for Instance Transfer Learning in Symbolic Regression

Qi Chen, *Member, IEEE*, Bing Xue, *Member, IEEE*, and Mengjie Zhang, *Fellow, IEEE*,

***Abstract*—Transfer learning has attracted more attention in the machine-learning community recently. It aims to improve the learning performance on the domain of interest with the help of the knowledge acquired from a similar domain(s). However, there is only a limited number of research on tackling transfer learning in genetic programming for symbolic regression. This article attempts to fill this gap by proposing a new instance weighting framework for transfer learning in genetic programming-based symbolic regression. In the new framework, differential evolution is employed to search for optimal weights for source-domain instances, which helps genetic programming to identify more useful source-domain instances and learn from them. Meanwhile, a density estimation method is used to provide good starting points to help the search for the optimal weights while discarding some irrelevant or less important source-domain instances before learning regression models. The experimental results show that compared with genetic programming and support vector regression that learn only from the target instances, and learning from a mixture of instances from the source and target domains without any transfer learning component, the proposed method can evolve regression models which not only achieve notably better cross-domain generalization performance in stability but also reduce the trend of overfitting effectively. Meanwhile, these models are generally much simpler than those generated by the other GP methods.**

***Index Terms*—Genetic programming, instance weighting, transfer learning.**

I. Introduction

TRANSFER learning is a relatively new learning scenario in machine learning which aims to enhance the learning performance in the domain of interest (known as the target domain), by utilizing the knowledge acquired from a similar domain(s), which is usually called the source domain(s). The motivation/rationale behind this new learning scenario is that human beings can solve new problems more effectively with knowledge acquired from similar problems. The necessity of transfer learning has been recognized in two typical scenarios, where sufficient instances are available in the source domain only but not available in the target domain, or the distribution of the data might change along with time. In these scenarios, traditional machine-learning techniques, which assume that the training data and the unseen data follow the same distribution, generally cannot work well. Denote the underlying distributions of the source domains as $P_s(Y_s, X_s)$, and $P_t(Y_t, X_t)$ for the target domain, where Y_s and Y_t are the outputs, and X_s and X_t are the corresponding input variables, transfer learning utilizes $P_s(Y_s, X_s)$ for better approximation of $P_t(Y_t, X_t)$ since $P(Y, X) = P(Y|X) \cdot P(X)$, $P_t(Y_t, X_t)$ can be derived from $P_s(Y_s, X_s)$ in two ways, that is, by adapting $P_s(Y_s|X_s)$ and adapting $P_s(X_s)$ [1].

In recent years, transfer learning techniques have been proposed in many fields, for example, reinforcement learning and supervised learning (e.g., classification) [2]–[6]. Compared with the many studies in those fields, transfer learning in GPSR is still rare and only limited pieces of work can be found [7]–[10]. All of them aim to transfer the knowledge learned from the source domain to the target domain by using the learned models. In GPSR, transfer learning via the relative importance of instances has not been considered to date.

Recently, we have proposed a new GP method TLGP for transfer learning in symbolic regression [11]. TLGP has been shown to extract reusable knowledge from the source domain to improve learning in the target domain. This is accomplished by utilizing a new instance weighting framework which evolves the weights for the source-domain instance along with the evolutionary process in GP. Despite the promising transfer learning ability of TLGP, there are some potential limitations in TLGP. Along with the increasing ratio between the numbers of the source-domain instances and the target-domain instances, that is, when a large number of source-domain instances but a relatively small number of target-domain instances are available, it becomes much more difficult for TLGP to identify useful source-domain instances, which limits its effort on enhancing the learning in the target domain. Meanwhile, it usually takes a high computation cost when searching for optimal weights for a large number of source-domain instances.

A. Goals

The goal of this article is to propose a new instance weighting framework for transfer learning in GPSR. It is designed

Manuscript received October 3, 2019; revised January 7, 2020 and January 13, 2020; accepted January 21, 2020. This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1509 and Contract VUW1615, in part by the Science for Technological Innovation Challenge (SfTI) Fund under Grant E3603/2903, and in part by the University Research Fund at Victoria University of Wellington under Grant 216378/3764 and Grant 223805/3986. This article was recommended by Associate Editor Y. S. Ong. *(Corresponding author: Qi Chen.)*

The authors are with the Evolutionary Computation Research Group, School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: qi.chen@ecs.vuw.ac.nz; bing.xue@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.2969689

to solve the above potential limitations and make the weights learning process be more effective and efficient than [11]. The proposed method aims to further improve the cross-domain generalization capability of the evolved regression models by effectively selecting and weighting the source-domain instances and correcting the difference between the marginal distributions $P_t(X_t)$ and $P_s(X_s)$ more precisely. Specifically, this article has three major objectives as follows.

1) It develops the instance weighting framework for GPSR by incorporating the density ratio estimation to effectively select and weigh the source-domain instances, and investigates whether the new weighting framework can outperform TLGP on enhancing learning in the target domain.
2) It investigates whether the new weighting framework can effectively obtain knowledge by selecting and weighting the source-domain instances to gain a better generalization capability of GPSR over learning from only the target-domain data.
3) It investigates whether the new weighting framework can detect more useful and transferable knowledge, thus leading to a better cross-domain generalization capability than GPSR learning from both the source- and target-domain data.

II. Background and Concepts

A. Transfer Learning

A formal definition of transfer learning [12] is as follows.

Definition 1 (Transfer Learning): Given a source domain D_s and a source learning task T_s, a target domain D_t and a target learning task T_t, where $T_s \neq T_t$ or $D_t \neq D_s$, the transfer learning process uses the knowledge in D_s and T_s to improve the learning of T_t in D_t.

In this definition, there are two important elements, *domain* and *task*. A domain $D = \{\mathcal{X}, P(X)\}$ has two components, that is, the feature space $\mathcal{X}$ and the marginal probability distribution $P(X)$, where $X = \{x_1, x_2, \ldots, x_d\} \in \mathcal{X}$, d is the number of features. A task $T = \{\mathcal{Y}, f(X)\}$, where $\mathcal{Y}$ is the label space and $f(X)$ is the predict function. The condition $T_s \neq T_t$ means that either $\mathcal{Y}_t \neq \mathcal{Y}_s$ or $f_t(X) \neq f_s(X)$, while $D_t \neq D_s$ indicates that either $\mathcal{X}_t \neq \mathcal{X}_s$ or $P_t(X) \neq P_s(X)$. The later case, which is also known as domain adaptation [12], [13], is the focus of this article.

Based on what to transfer, transfer learning techniques can be grouped into four categories, that is: 1) instance-based; 2) feature representation-based; 3) parameter-based; and 4) relational knowledge-based transfer learning approaches [12]. This article focuses mainly on instance-based transfer learning.

B. Instance-Based Transfer Learning

Instance weighting, as a generalized form of statistical bias correction techniques, has the potential to correct the sample selection bias. Therefore, it is considered as an important type of transfer learning method to correct the distance between the source and target domains. The rationale under instance weighting for transfer learning is that, due to the different distributions in the source and target domains, some of the source-domain data could be harmful while some others can be reused for learning in the target domain by reweighting to correct the marginal distribution difference between the two domains.

Given the source-domain distribution/density $P_s(x)$ and the target-domain distribution $P_t(x)$, the density ratio $w(x) = (P_s(x)/P_t(x))$ has shown to be effective in alleviating the distribution difference between the two domains. Thus, it has been extensively used for transfer learning. Since $P_s(x)$ and $P_t(x)$ are typically unknown in reality, density ratio estimation thus becomes a key problem for instance weighting. A number of methods, which directly estimate the density ratio, have been proposed for transfer learning/domain adaptation [14], [15].

The kernel mean matching (KMM) [14] is proposed to make the weighted distribution of the source domain be similar to the distribution of the target domain. KMM learns $w(x)$ by matching the means between the source-domain data and the target-domain data in a reproducing-kernel Hilbert space (RKHS). The Kullback–Leibler importance estimation procedure (KLIEP) [15] proposed a similar idea to estimate $w(x)$ directly by minimizing the Kullback–Leibler divergence to estimate weights. These studies proved that theoretically, it is able to learn precise models in the target domain by weighting the source-domain instances with proper values of $w(x)$. However, it is difficult to have a general method for an accurate estimation of $w(x)$ for various learning algorithms.

A boosting algorithm called TrAdaBoost [16] is proposed to identify and make use of more relevant source-domain data iteratively. TrAdaBoost decreases the relative weight of a source instance that is misclassified while increasing the weight of a misclassified target instance. In this way, TrAdaBoost will focus on the source instances that are more similar to the target instances as well as the more difficult instances. Later, TrAdaBoost is extended for regression tasks [17]. However, as reported in [17], the transfer learning performance of TrAdaBoost on regression tasks is not as good as expected since the instances in the source domain would dominate the learning process.

Instance selection of the source data is also a sensitive method for transfer learning. Lawrence and Platt [18] extended the informative vector machine (IVM) for selecting instances for multitasking problems. The extended IVM method selects more instances from tasks which provide more information to the learning process.

C. Evolutionary Transfer Learning

Min *et al.* [19] have developed a metaheuristic-based instance selection method (MIST) for transfer learning. MIST uses a population-based metaheuristic where each solution represents a subset of source-domain instances. The quality of the solution is evaluated by the estimated generalization error of the Gaussian process, which learns from a combination of the select source data and the target training data. MIST searches for an optimal subset of source-domain instances guided by their contributions to the transfer learning process. This is similar to the transfer learning component proposed in this article. On the other hand, each of the selected source instance in MIST will contribute equally to the transfer learning process, while by assigning weights to the selected source, the proposed instance weighting framework is expected to help the transfer learning process more precisely. Gupta *et al.* [20] proposed a multifactorial evolutionary algorithm (MFEA) for multitask learning. MFEA performs an implicit parallelism search of optimal solutions for multiple tasks. They introduce an important concept of skill factor to determine the task on which each individual is most effective among the given tasks. The skill factor is allowed to inherit from parents to the children as a kind of vertical cultural transmission. They have shown that MFEA can solve the diverse problem concurrently and converge faster. Feng *et al.* [21] studied the goodness of sharing knowledge across different tasks and proposed a new method to solve evolutionary multitasking via explicit genetic transfer across tasks. A denoising autoencoder is utilized to construct the mapping of any two tasks. The solutions are transferred from one task to another via these task mappings. The proposed method is shown to be able to solve both single and multiobjective multitasking tasks more effectively than state-of-the-art methods. Chandra *et al.* [22], [23] proposed co-evolutionary algorithms, for solving multitasking problems via neural networks. They have proposed various strategies for decomposing the problem into multiple tasks. The knowledge is transferred via the cascades in the structure of neural networks. The efficacy of the algorithms has been shown via comparing with state-of-the-art neuroevolution methods.

D. Transfer Learning in Symbolic Regression

Comparing with the many existing works in transfer learning for traditional regression, there are only a small number of studies on transfer learning in GPSR in the literature. All of them focus on modular-based transfer learning. Dinh *et al.* [7] transferred the knowledge acquired from the source domain in three types of models, including the best fitted models and submodels at the final generation, and the best of generation models of GP on the source-domain data. When tackling the target-domain problem, these models will join the initial population of GP. The transfer learning effort using all these models is limited. Haslam *et al.* [8] further the study in [7]. They added additional components to detect the reusable models and also approached the feature transfer. O'Neill *et al.* [9] collected potentially useful building blocks in GP on the source-domain data and extended the function set of GP when learning from the target domain. Zhong *et al.* [24] have extended the MFEA [20] to solve GPSR as a multitasking problem. In their multifactorial GP method, knowledge could be transferred across symbolic regression tasks and some other tasks, for example, even parity.

III. Proposed Method

This article addresses the transfer learning task in symbolic regression using GP with a new instance weighting framework. The rationale behind the proposed framework is to search for the optimal weights for the source-domain instances, which can effectively reuse the informative source-domain instances while eliminating the effort of the harmful source instances.

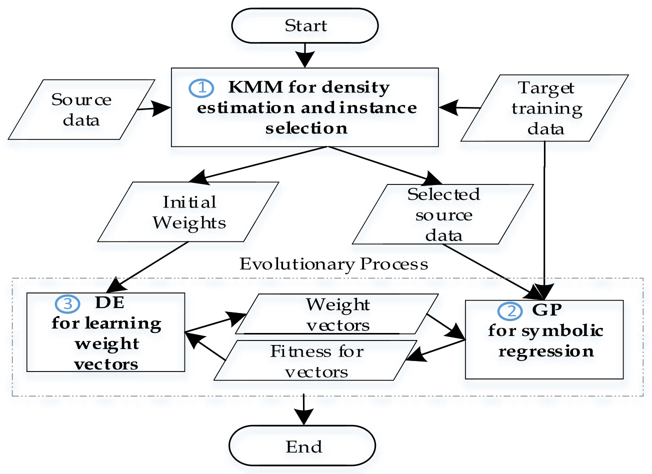

Fig. 1. Three key components (1–3) with data flows in ITGP.

We have done some preliminary investigation in [11]. Due to the page limit, Chen *et al.* [11] did not have a comprehensive framework. When an overly large number of source-domain instances are available along with a relatively small number of target-domain instances, searching for the optimal weights using the method in [11] becomes less effective and efficient. In this case, the source-domain data might dominate the transfer learning process. A new instance weighting framework which attempts to alleviate the dominance of the source-domain data in the learning process is proposed in this article. More specifically, the method in [11] searches for an optimal weight vector for the source-domain instances, while the instance weighting framework in this article performs instance selection on the source-domain instances while searching for optimal weight vectors for the selected instances.

The new instance framework seeks to search for the optimal weights, therefore, a numerical search method is needed. Evolutionary algorithms, such as particle swarm optimization (PSO), genetic algorithms (GAs), and differential evolution (DE) [25], can be utilized to fulfill the search process in the proposed framework. Comparing with PSO and GAs, DE is generally simpler and has a more diverse population [26], thus a very flexible version of DE, that is, SaDE [27], is employed.

A. Overall Structure

GP equipped with the new instance weighting framework is called instance transferring GP (ITGP). The overall structure and the data flow in ITGP are shown in Fig. 1.

As shown in Fig. 1, ITGP has three key components. KMM, which is a density ratio estimation approach, is employed to estimate the initial weights for the source-domain instances. These initial weights are considered as a good starting point for searching for the optimal weights. Meanwhile, an instance selection process based on these weights is performed simultaneously. The evolutionary process of ITGP is performed on the selected source-domain instances which are weighted by the weight vectors. At every generation, GP evolves a population of regression models while DE [27] evolves a population of weight vectors. The relationship between GP and DE is that GP learns from different sets of weighted source-domain instances using the weight vectors provided by DE, while DE optimizes a set of weight vectors by taking the feedback from GP, that is, the performance of the learned regression models on the target training data, as the fitness of the weight vectors. The transfer learning process in ITGP is as follows.

1) *Step 1:* KMM initializes the m weight vectors and selects the source-domain instances with a high weight. The weight vectors are defined as $V_i = \{[v_{1,d}, v_{1,w}], [v_{2,d}, v_{2,w}], \ldots, [v_{n,d}, v_{n,w}]\}$. V_i is a vector having n elements. Each element has two dimensions $[v_{j,d}, v_{j,w}]$. The first dimension $v_{j,d}$ refers to the index of the source-domain instance while the second dimension $v_{j,w}$ is the weight of this instance.
2) *Step 2:* GP initializes a population of models.
3) *Step 3:* DE uses the weight vectors provided by KMM as its initial population.
4) *Step 4:* Repeat the following evolutionary process of ITGP until the stopping criterion is met.
    a) *Step 4.1:* GP takes weight vectors from DE and evaluates all the GP individuals on the weighted source data accordingly.
    b) *Step 4.2:* GP re-evaluates the top GP individuals on the target training data and provides the fitness values to DE.
    c) *Step 4.3:* DE takes the fitness values from GP and generates the children.
    d) *Step 4.4:* GP selects parents and generates the child individuals.

The details of the each component in ITGP will be presented in the following sections.

B. Essential Data Components in ITGP

This section introduces the two essential data components in ITGP. The description of the data components is necessary for understanding the transfer learning process in ITGP.

1) Weight Vectors in ITGP: In ITGP, a population of weight vectors is initialized by KMM and evolved by DE. As shown in Fig. 2, these weight vectors have a fixed length, which is equal to the number of selected source-domain instances. The length is generally smaller than the number of original source-domain instances. Each weight vector represents a subset of weighted source-domain instances. Different weight vectors in the population of DE refer to different subsets of source-domain instances with various weights. (Note that during the evolutionary process, the index of the instance will not change.) This is a key difference from the weight vector in our previous work [11], where the weight vectors represent different weights for the same source-domain instance.

2) Fitness Vector Containing Error Values on the Source-Domain Data: Another data component is the fitness vector, which is also shown in Fig. 2. As it shows, ITGP considers the performance of a model on the source data using all the weight vectors provided by DE, that is, $V_1, V_2, \ldots, V_m$. Thus, different from a real number standing for a kind of error value in standard GP, the fitness of a model in ITGP is a vector having m dimensions, where m is the number of weight vectors. Each dimension is the weighted error on the selected source-domain data. The details on how to obtain the weighted error are described in Section III-D.

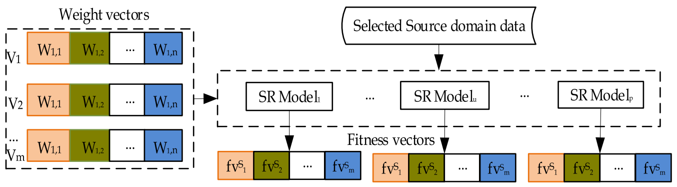

Fig. 2. Weight vectors and fitness vectors in ITGP.

C. KMM for the Initialization of Weight Vectors and the Selection of Source-Domain Instances

In ITGP, KMM [14], which is a well-known density ratio estimation algorithm, is used for the source-domain instance selection and providing the initial population which consists of a set of weight vectors for DE. KMM is based on infinite-order moment matching and effectively matches all the moments using kernel functions. KMM estimates density ratios by minimizing the maximum mean discrepancy (MMD) between the weighted distribution $\beta(x)P_s(x)$ and the distribution $P_t(x)$ in an RKHS $\Phi(x)$ which is the feature map from x to $\mathcal{F}$. Using two Gram kernel matrices [usually radial basis function kernel (RBF) is used]

$$K_{i,j} = k\left(x_i^s, x_j^s\right) \tag{1}$$

and

$$k_i = \frac{n_s}{n_t}\sum_{j=1}^{n_t} k\left(x_i^s, x_j^t\right) \tag{2}$$

where x_i^s is the source-domain instance, x_j^t is the target-domain instance, k is the kernel, and n_s and n_t are the number of the source and target instances, respectively. MMD can be obtained by

$$\begin{aligned}\text{MMD} &= \left\| \frac{1}{n_s}\sum_{i=1}^{n_s}\beta_i\Phi\left(x_i^s\right) - \frac{1}{n_t}\sum_{i=1}^{n_t}\Phi\left(x_i^t\right)\right\|^2 \\ &= \frac{1}{n_s^2}\beta^T K\beta - \frac{2}{n_s^2}k^T\beta + \text{constant}\end{aligned}$$

and a suitable set of weights β can be found by solving a quadratic problem via

$$\text{minimize}\ \left(\frac{1}{2}\beta^T K\beta - k^T\beta\right) \tag{3}$$

which is subject to $\beta \in (0, B)$, and $|\sum_{i=1}^{n_s}\beta_i - n_s| \le n_s\epsilon$, $\epsilon = O(B/\sqrt{n_s})$. Here, B is the boundary value for β.

As shown in Algorithm 1, by providing m various B values, that is, $B = \{B_1, B_2, \ldots, B_m\}$ (typically $B \le 1000$ as suggest in [14]), KMM will obtain m set of weights. To prevent the source-domain data from dominating the learning process, only a fixed number of source-domain instances with higher weights are selected. This number of the selected source-domain instances n is determined by n_t and a scale parameter t, that is, $n = n_t \times t$. Typically, t is set to a small value. A set of trail experiments shows that $t \le 5$ is a good

Algorithm 1: KMM for Weight Vectors Initialization and Instance Selection in ITGP

```
Input: $n_s$ source domain instance $S(X)$, and $n_t$ target-domain instances $T(X)$, the scale value $t$, the number of weight vectors $m$, a set of $B = \{B_1, B_2, \ldots, B_m\}$;
Output: a set of weight vectors $V = \{V_1, V_2, \ldots, V_m\}$;
for $q := 1$ to $m$ do  Obtaining the weights of the $n_s$ source domain instance by KMM and select the source instances
    Initialise $V_q$ with a length of $n = n_t \times t$;
    Obtain $K_{i,j}$ (according to Equation (1), using Radial Basis Function kernel);
    Obtain $k_i$ (using Equation (2));
    Calculate $\epsilon = B_q/\sqrt{n_s}$;
    Obtain $\beta$ = quadratic problem solver( $K_{i,j}$, $k_i$, $B_q$, $\epsilon$) (according to Equation (3));
    Rank $\beta = \{\beta_o, \beta_p, \ldots, \beta_q\}$, where $\beta_o > \beta_p > \beta_q$ and o, p, and q are indexes of source instances;
    for $r := 1$ to $n_t \times t$ do
        $V_{q,r0} = index(\beta_r)$;
        $V_{q,r1} = \beta_r$;
    Add $V_q$ to $V$;
Return $V = \{V_1, V_2, \ldots, V_m\}$;
```

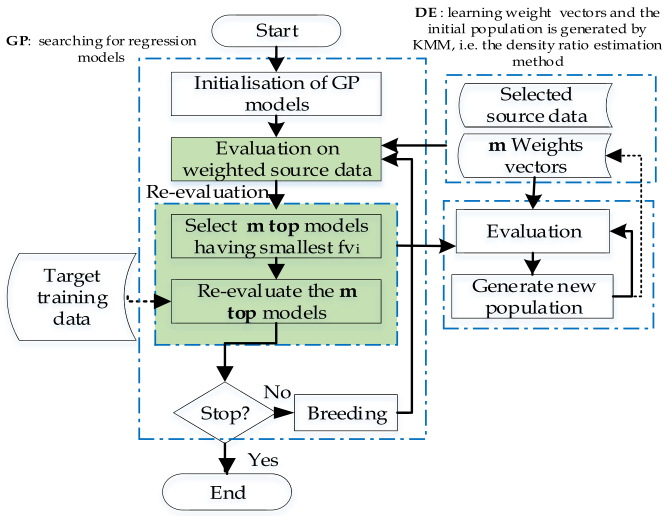
GP: searching for regression models
Start
DE: learning weight vectors and the initial population is generated by KMM, i.e. the density ratio estimation method
Initialisation of GP models
Selected source data
m Weights vectors
Evaluation on weighted source data
Re-evaluation
Select m top models having smallest fvi
Evaluation
Target training data
Re-evaluate the m top models
Generate new population
Stop?
No
Breeding
Yes
End

Fig. 3. Evolutionary process of ITGP.

choice. In this article, for tasks where the number of available target-domain instances is small, t is set to 3. Otherwise, t is set to 2. More importantly, by reducing the number of source-domain instances while keeping the relatively important ones estimated by KMM, it will shrink the search space of DE, thus can potentially improve the efficiency and effectiveness of searching for the optimal weight vector.

D. Evolutionary Process of ITGP

Assuming GP and DE have generated their initial populations, that is, a population of p trees/models in GP and m weight vectors in DE, where each vector contains n bits corresponding to the weights for the n selected source-domain instances, the evolutionary process in ITGP, which is shown in Fig. 3, consists of the following steps.

1) GP evaluates the p individuals/models on the selected source-domain data, and each model has m weighted error values (corresponding to the m weight vectors).
2) GP selects the m models by comparing the fitness vectors of all the models in each dimension. For each dimension k, we can find the model with the smallest weighted error fv^S_k. This model is referred to as the best model in dimension k. In other words, the best model in dimension k has the smallest fv^S_k.
3) GP re-evaluates the m models by measuring their errors on the target-domain training instances.
4) DE takes the feedback from GP, that is, the re-evaluate errors of the m models, as the fitness of the weight vectors.
5) DE generates a population of trail weight vectors utilizing the mutation strategies and the crossover operator.
6) Repeats steps 1)–4) using the trail weight vectors and obtains their fitnesses.
7) DE generates the new generation by comparing the fitnesses of weight vectors and their corresponding trail vectors, and GP updates the top m individuals according to the survival weight vectors.

E. Evaluation in ITGP

The evaluation process in ITGP consists of two parts, that is, the evaluation on the source data and the re-evaluation on the target training data. The evaluation is to measure the performance of GP individuals on the weighted source instances while the re-evaluation measures how the top GP individuals perform on the target domain. During this process, two fitness functions are employed.

1) Evaluation on the Source Data: When evaluating GP models in ITGP, the weighted error on the selected source-domain data are measured by the weighted mean-squared error (WMSE), as shown in

$$fv^S_i = \text{WMSE} = \sum_{j=1}^{n} w_j \cdot \left(f_j - y_j\right)^2, (i = \{1, 2, \ldots, m\}) \quad (4)$$

where n is the number of selected source-domain instances, w_j is the weight of the jth selected instance, f_j is the output of the model, and y_j is the target value for the jth selected instance. Each fitness vector has m components, that is, $fv^S_1, fv^S_2, \ldots, fv^S_m$. Each of these fv^S values are obtained by the models on a set of selected source-domain instances with their corresponding weights.

2) Re-Evaluation on the Target Data: To measure the performance of the models learned from the weighted source instances on the target domain and the performance of the weight vectors/DE individuals, ITGP will select m models for re-evaluation. These m models have the smallest fv^S_i, respectively. They are the best GP individuals under each weight vector, and their learning performance on the target training data highly depends on how much difference the weight vector can correct the difference between the distributions of the source and target domains. Thus, the training error of these models on the target data can be considered as a good indicator of the fitness of the weight vectors. In the re-evaluation process, the commonly used fitness measure—mean-square error

(MSE) is employed as

$$fv^T = \text{MSE} = \frac{1}{n_t}\sum_{i=1}^{n_t}(f_i - y_i)^2 \tag{5}$$

where n_t is the number of target training instances, f_i is the predict value of the model on the ith target training instance, and y_i is the corresponding target value.

F. Breeding in ITGP

At every generation, ITGP maintains two populations of individuals, that is, GP individuals/regression models and DE population representing a set of weight vectors. Compared with standard GP, the breeding of regression models in ITGP uses two new mechanisms, that is, how to determine elitism and how to select the parents for breeding. The elites in ITGP are from the top tp models within the m winners ($tp \leq m$) that have been exposed to the re-evaluation process. On the other hand, for generating the remaining individuals in the new generation, that is, $(p - tp)$ new individuals, parent(s) are selected using a new tournament selection operator.

As we mentioned in the previous section, instead of a real number, the fitness of a model is represented in a fitness vector containing m dimensions, that is, $fv = fv_1^S, fv_2^S, \ldots, fv_m^S$. Due to the new representation of the fitness in ITGP, we extend the original tournament selection operator to compare the fitness vectors. The new selection operator compares the fitness vectors of candidate individuals in a number of dimensions, which is denoted as $D(D \leq m)$. In our preliminary work [11], the D dimensions are randomly sampled, where each of the original m dimensions has the same opportunity to determine the winner of the tournament. Actually, the importance of each dimension, which is determined by the quality of the weight vector is not equal. Treating them equally might miss good candidate parents. To compare the candidate parents in a more precise way, in ITGP, the D dimensions are selected by the *roulette wheel selection*. Specifically, the selection process is performed in the following steps.

1) *Step 1:* T GP individuals are randomly sampled according to the tournament size.
2) *Step 2:* D dimensions are selected using the *roulette wheel selection*, that is, the fitnesses of weight vectors is used for determining the probability of the corresponding dimension is chosen for comparison.
3) *Step 3:* Compare with the fitness values of the T individuals in pairs in these D dimensions. The one which has not worse (equal or better) than other's fitness values in all the D dimensions is the winner and will be selected as a parent.

The value of D is related to the total number of dimensions of the fitness in ITGP, that is, m. Usually, a larger m needs a larger D. The value of D also influences the selection pressure to some degree. The larger D leads to higher selection pressure and makes the selection process focus mainly on the best individuals. This will decrease/limit the diversity of the GP population and might lead to premature converge. Too small D will increase the randomness of the selection process and miss good solutions. In this article, D is set to 3 according to the relatively small value of m ($m = 50$).

Meanwhile, for generating new weight vectors in SaDE, four commonly referred and effective trail vector generation strategies are used, which are shown as

$$V_i = W_{r1} + F \cdot (W_{r2} - W_{r3}) \tag{6}$$
$$V_i = W_{\text{best}} + F \cdot (W_{r2} - W_{r3}) \tag{7}$$
$$V_i = W_i + K \cdot (W_{r3} - W_i) + F \cdot (W_{r1} - W_{r2}) \tag{8}$$
$$V_i = W_i + F \cdot (W_{\text{best}} - W_i) + F \cdot (W_{r1} - W_{r2}). \tag{9}$$

Here, the indices r_1, r_2, and r_3 are the index of the randomly selected individuals. K and F are scaling factors, which usually have a positive value to scale the difference vector. The adjust of parameters, such as F and CR follows the recommended way in [27]. The selection process for generating the new generation in SaDE is determined by the comparison between the fitnesses of the trail vector and the target vector, which are determined by the performance of the corresponding top GP individuals on the target training data.

In summary, ITGP utilizes a new instance weighting framework to guide the evolutionary process by evolving weight vectors. Compared with our preliminary work [11], a new instance selection and weight vector initialization method is employed, along with a new selection operator which selects parents from a more precise comparison. In this way, the regression models and the weight vectors are learned together with good interactions during several stages in the evolutionary process. With these new developments, ITGP is expected to be more effective and efficient to extract much useful information from the source domain to enhance the performance on the target domain.

IV. Experiment Design

A. Benchmark Methods

To investigate the effectiveness of the proposed method, in the experiments, it is compared with *seven* benchmark methods. Four of them are GP methods, and the other three are support vector regression (SVR) methods [28]. These benchmark methods are described as follows.

1) *GP-Tar*, which refers to standard GP using the target training data only.
2) *GP-Comb*, which is standard GP method learning from a combining set of the source data and the target training data.
3) *TLGP*, which was proposed in our preliminary work [11] and is the basis of ITGP. Compared with ITGP, TLGP performs implicitly instance selection on the source domain and randomly initializes the population for DE. The comparison between TLGP and ITGP is to confirm the effort of the instance selection and initialization method for generating the population of weight vectors, as well as the new selection operator based on both the fitness of the weight vectors and the fitness of GP individuals.
4) *TLGP-NS*, which is a TLGP method using the new selection operator. The comparison among TLGP, TLGP-NS,

and ITGP is to confirm the effort of the new selection operator.

5) *SVR-Tar* and *SVR-Comb*, which stand for *SVR* with the two kinds of training data. The performance of SVR with the radial RBF learning from the target data only and a combination of the source- and target-domains data are both used for comparison.
6) *SVR-W*, which refers to SVR learning from a combination of the source- and target-domain data using the weights obtained from KMM. The weights are also used in ITGP.

We also considered comparing with TLGP using KMM. Due to the page limit, we have shown the additional comparison in the online supplementary material[1] of this article. All the GP methods are implemented under the ECJ GP framework [29]. SVR methods are implemented under the *R* packages "e1071" [30] using the RBF kernel function and other default settings.

B. Datasets

Due to the lack of the transfer learning benchmark problem in symbolic regression, some recent related work modified traditional regression datasets for testing transfer learning methods [17], [31]. In this article, we follow these works and modify six datasets, including five real-world problems from the Delve dataset repositories[2] and the UCI [32], and one synthetic dataset generated by the well-known Friedman-1 (Friedman) [33].

In the original Friedman problem, each instance $X = \{x_1, x_2, \ldots, x_{10}\}$ has ten variables where each variable $x_i, i \in [1, 10]$ follows the uniform distribution $[0, 1]$. But only the first five input variables are related to the output variable y, that is

$$\begin{aligned} y = a_1 \times 10 \sin(\pi(b_1x_1 + c_1) \times (b_2x_2 + c_2)) \\ + a_2 \times 20(b_3x_3 + c_3 - 0.5)^2 + a_3 \times 10(b_4x_4 + c_4) \\ + a_4 \times 5(b_5x_5 + c_5) + N(0, 1) \end{aligned}$$

where a_i, b_i, and c_i, $i \in [1, 5]$ are parameters, and N refers to the normal distribution. Different values are set to these parameters when generating the source- and target-domain data. For the source data, $a_i = b_i = 1$ and $c_i = 0$, while a_i and b_i are drawn from $N(1, 0.1)$ and c_i is drawn from $N(0, 0.05)$ for the target-domain dataset [17].

In two real-world benchmark problems, the source data and the target data are two datasets having the same feature space and similar tasks. For the other three real-world datasets, according to one specific feature, the different values of which can typically distinguish the distribution of the data, the datasets are split into the source data and target data. Then, the same features in the two domains are perturbed using random numbers following different normal distributions. This can make sure the distribution of the two domain data is different but not too dissimilar, which is a key assumption of transfer learning methods considered in this article. The detailed introduction of the real-world datasets is as follows.

1) The wine quality dataset (Wine) which aims to predict the quality of the wine, including the red and the white wine samples. We use the quality prediction problem for the white wine as the source-domain problem and the quality prediction for the red wine as the target-domain task.
2) The kin dataset (Kin) is a family of datasets. This article uses the "n" datasets (n for nonlinear) where "nm" (nonlinear medium variance) dataset is used as the source-domain data while the "nh" (nonlinear high variance) dataset is the target data. Features in the target data were perturbed by random numbers following the distribution of $N(0.5, 0.1)$.
3) The student performance dataset (Student) which approaches student achievement in secondary education of two Portuguese schools. The original two datasets provide the performance in two distinct subjects mathematics and Portuguese language. The dataset containing mathematical performance is treated as the source data while the dataset for predicting the Portuguese language is used as the target-domain dataset.
4) The housing dataset (House) which aims to predict the "MEDV." The dataset is split into two datasets according to the feature "TAX." The source data are instances with TAX <= 600, while the target data are instances with TAX > 600. Meanwhile, three features, "RM," "AGE," and "B" are perturbed where the source data are perturbed by random numbers in $N(0.1, 0.1)$, while in the target data, they are perturbed by random numbers drawn from the distributions $N(7, 1)$, $N(5, 1)$, and $N(8, 1)$, respectively.
5) The abalone dataset (Abalone) which predicts the age of abalone from physical measurements. The original dataset has both the information of female and male abalone. We treat the male data as the source-domain data and the female data as the target data.

The number of features and instance in all the datasets are summarized in Table I. Note that the difference between the distributions in the two domains in each problem has been verified by the multivariable Kolmogorov–Smirnov (KS) test [34]. The relative square error (RSE) on the target data are reported for easier comparisons. The definition of RSE is as follows:

$$\mathrm{RSE} = \sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{(\bar{y} - y_i)^2} \tag{10}$$

where y_i is the ith target output, $\hat{y}_i$ is the ith output of model, and $\bar{y}$ is the mean outputs. The relative error shows how the performance of the candidate model ($\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$) is when compared with a simple predictor using the average of y values ($\sum_{i=1}^{n}(\bar{y} - y_i)^2$). In each of the five GP methods, 50 independent runs have been conducted on each of the six problems. The parameter settings for all the GP methods are summarised in Table II and most of them are common setting for GP.

V. Results and Analysis

A. Performance on the Target Domain

The effectiveness of the eight methods is compared using their performance on the target domain, which is the focus

[1]http://homepages.ecs.vuw.ac.nz/~chenqi1/Mywork/OLS-GPTL.pdf
[2]https://www.cs.toronto.edu/~delve/data/datasets.html

TABLE I
DATASETS

Dataset	#Features	#Source Data	#Target-Training	#Target-Test
Wine	11	4898	160	1439
Kin	8	8192	819	7373
Student	45	689	40	355
House	13	368	14	124
Abalone	7	1528	130	1177
Friedman	10	1000	50	200

TABLE II
PARAMETER SETTINGS FOR GP METHODS

Parameter	Value
Population Size	512
Maximal #Generations	100
Crossover & Mutation Rates	0.9 & 0.1
Elitism	1
Maximum Tree Depth	10
Initialisation	Ramped half-and-half
Initial Tree Depth	2-6
Function Set	$+, -, *, \%$ protected (a%0 = 1)
D in selection	3
Population size W in SaDE	50
F and K for SaDE	$N(0.5, 0.3)$
CR_m for SaDE	$\{0.5, 0.5, 0.5, 0.5\}$
t for instance selection	2 for Wind, Kin, and Abalone; 3 for the other three

TABLE III
RSES ON THE TARGET DOMAIN

	Method	Training RSE Median(Std)	Test RSE Median(Std)	VS. GP-Tar (Training, Test)	VS. ITGP (Training, Test)
Wine	SVR-Tar	0.23	0.98	(−,+)	(−,+)
	SVR-Comb	**0.025**	1.5	(−,+)	(−,+)
	SVR-W	0.81	0.87	(=,−)	(−,+)
	GP-Tar	0.78 ± 0.05	0.91 ± 0.06	NA	(−,+)
	GP-Comb	1.32 ± 0.16	0.9 ± 0.06	(+,=)	(+,+)
	TLGP	1.12 ± 0.05	0.71 ± 0.21	(+,−)	(=,+)
	TLGP-NS	1.1 ± 0.06	0.74 ± 0.21	(+,−)	(=,+)
	ITGP	1.07 ± 0.05	**0.69 ± 0.23**	(+,−)	NA
Kin	SVR-Tar	**0.39**	1.53	(−,+)	(−,+)
	SVR-Comb	0.94	0.95	(−,+)	(−,+)
	SVR-W	0.41	0.83	(−,+)	(−,+)
	GP-Tar	0.62 ± 0.08	0.82 ± 0.04	NA	(−,+)
	GP-Comb	0.62 ± 0.11	0.79 ± 0.06	(=,=)	(−,+)
	TLGP	2.49 ± 0.2	0.71 ± 0.09	(+,−)	(=,+)
	TLGP-NS	2.44 ± 0.19	**0.7± 0.09**	(+,−)	(=,=)
	ITGP	2.36 ± 0.11	**0.7 ± 0.05**	(+,−)	NA
Student	SVR-Tar	0.58	1.0	(+,+)	(+,+)
	SVR-Comb	0.08	0.78	(+,+)	(+,+)
	SVR-W	0.43	0.82	(+,+)	(+,+)
	GP-Tar	**0.03 ± 0.01**	0.47 ± 0.15	NA	(−,+)
	GP-Comb	0.14 ± 0.01	0.56 ± 0.02	(+,+)	(+,+)
	TLGP	0.05 ± 0.01	0.2 ± 0.1	(=,−)	(=,+)
	TLGP-NS	0.05 ± 0.01	0.19 ± 0.12	(=,−)	(=,+)
	ITGP	0.04 ± 0.01	**0.18 ± 0.05**	(−,+)	NA
House	SVR-Tar	0.24	1.24	(+,+)	(−,+)
	SVR-Comb	0.27	1.89	(+,+)	(−,+)
	SVR-W	0.79	1.96	(+,+)	(−,+)
	GP-Tar	**0.08 ± 0.1**	0.67 ± 0.15	NA	(−,+)
	GP-Comb	0.49 ± 0.18	0.6 ± 0.3	(+,=)	(−,+)
	TLGP	1.97 ± 0.24	0.38 ± 0.26	(+,−)	(=,+)
	TLGP-NS	1.94 ± 0.15	0.33 ± 0.26	(+,−)	(=,+)
	ITGP	1.87 ± 0.13	**0.32 ± 0.14**	(−,+)	NA
Abalone	SVR-Tar	0.7	1.34	(+,+)	(−,+)
	SVR-Com	0.78	1.08	(+,+)	(−,+)
	SVR-W	0.78	1.56	(−,+)	(−,+)
	GP-Tar	**0.54 ± 0.01**	0.99 ± 0.06	NA	(−,+)
	GP-Com	0.57 ± 0.02	0.84 ± 0.04	(=,−)	(−,+)
	TLGP	4.13 ± 0.11	0.96 ± 0.13	(+,=)	(+,+)
	TLGP-NS	4.11 ± 0.07	0.95 ± 0.15	(+,=)	(+,+)
	ITGP	3.9 ± 0.08	**0.79 ± 0.16**	(+,−)	NA
Friedman	SVR-Tar	0.79	0.86	(+,+)	(+,+)
	SVR-Comb	215.34	1.03	(+,+)	(+,+)
	SVR-W	0.57	0.76	(+,+)	(+,+)
	GP-Tar	0.18 ± 0.03	0.65 ± 0.04	NA	(+,+)
	GP-Comb	235.71 ± 2.96	14.5 ± 0.07	(+,+)	(+,+)
	TLGP	0.12 ± 0.01	0.63 ± 0.16	(−,−)	(=,+)
	TLGP-NS	0.12 ± 0.008	0.68 ± 0.25	(−,=)	(=,+)
	ITGP	**0.1 ± 0.01**	**0.32 ± 0.16**	(−,−)	NA

of transfer learning approaches. Table III provides a summary of the RSEs obtained by the learned models on the target-domain data. While the median values and the standard deviations of the RSEs obtained by the 50 best-of-run models on the target training data and the target test data have been presented for the five GP methods, the RSE obtained by SVR models is also presented. Since SVR is a deterministic learning method, only one RSE has been obtained on each dataset. Note that the training RSEs are calculated on the target training data only, which is not the whole training set for GP-Comb, SVR-Comb, and SVR-W. Two sets of nonparametric statistical significance tests, the Wilcoxon test with a significance level of 0.05, are performed to compare the training RSEs and test RSEs between the best-of-run models in ITGP (and GP-Tar) and the other GP methods. Z-test is used to test whether there is a significant difference between GP methods (with 50 groups of results) and SVR methods (one group of results). In Table III, the two sets of statistical test results are presented, which show clearly whether one method can significantly outperform GP-Tar and ITGP. By comparing with GP-Tar, it is easy to know whether a GP method performs a positive transfer learning. While "−" stands for ITGP (GP-Tar) performs significantly worse than the compared method, "+" indicates ITGP (GP-Tar) is significantly better, and "=" means no significant difference. "N/A" means no comparison when the compared method is ITGP (or GP-Tar) itself.

1) Comparisons on the Target Training Data: It is clearly shown in Table III that GP methods equipped with an instance weighting framework have a similar pattern. They usually have a much higher training error than GP-Tar and SVR methods on the examined problems. Specifically, TLGP, TLGP-NS, and ITGP have worse training performance than GP-Tar on five of the six target training datasets except for Friedman, while their training performance is worse than those of SVR methods on four of the six datasets except for Student and Friedman. When comparing with GP-Comb, on Wine, Student, and Friedman, they have smaller training errors, but not the case on the other three datasets. According to the statistical significance test, all the differences on the training performance are significant. The results on the target training data are not unexpected. The reason is that the source- and target-domain data follow a different distribution, and in ITGP (the same for TLGP and TLGP-NS), the target training instances involved in the training process by selecting models and evaluating the weights vectors only. Compared with methods where these instances are directly used in learning the regression models, it is very likely for ITGP (and the other two GP methods) to have less effective training fitting to the target-domain data.

Among GP Methods: Considering the training performance of the three GP with instance weighting methods, TLGP-NS has slightly better training performance than TLGP, which indicates the new selection operator contributes to guiding the training process toward regression models that learn better on the target domain by assigning a higher probability for considering fitter weight vectors. The new proposed method ITGP can have better training performance than TLGP and TLGP-NS in most of the examined cases. On five of the six datasets, ITGP has slightly better training performance than TLGP and TLGP-NS, while on Abalone, ITGP has a significantly better training performance than the other two methods. This indicates that incorporating instance selection, which aims to keep the more important source-domain instances while eliminating the less helpful ones, helps to release the dominance of the source-domain data during the learning process. However, the training error in ITGP still much higher than GP-Tar in most cases.

GP Versus SVR: The comparisons between GP methods and SVR methods where the same training data are provided, that is, the comparisons between GP-Tar and SVR-Tar, and between GP-Comb and SVR-Comb, show that on Wine and Kin, and Abalone, where there are a large number of target-domain instances available, GP-Tar has a higher training error then SVR-Tar. However, on the other three datasets, where there are a relatively smaller number of training instances available, GP-Tar outperforms SVR-Tar. When exposed to the combination of the source- and the target-domain data, GP-Comb has better training performance than SVR-Comb on Kin and Abalone, but not the case on the other four datasets. All the findings indicate that compared with SVR, GP is good at learning from a smaller number of instances.

With Versus Without the Source-Domain Data: Comparing the methods learning directly from both the source-domain and target-domain data with the corresponding method that learns from the target-domain data only, that is, GP-Comb versus GP-Tar and SVR-Comb versus SVR-Tar, GP-Comb and SVR-Comb generally have worse training performance than their counterpart learning from the target-domain directly. This is particularly the case on Friedman where the distribution difference between the source and the target domains is large. The worse performance of GP-Comb and SVR-Comb, that is, when learning directly from the source-domain data, is mainly due to the difficulty in detecting transferable knowledge without any transfer learning strategy. Meanwhile, the dominance of the source-domain instances over the learning process is also a potential reason. In this case, SVR-W with weighted source-domain instances can effectively improve the training performance in some cases, for example, on Kin and Friedman, but not all the cases.

2) Comparing Results on the Target Test Data: Considering the generalization performance on the target test sets, which is a more important criterion to measure the transfer learning performance, the proposed method ITGP is definitely the winner among the eight methods. On all the six test sets, ITGP has much smaller RSEs than the other seven methods, which are all significant except for comparing with TLGP-NS on Kin. This pattern can be easily found in Fig. 4, which shows the distribution of the test RSEs in GP methods. The boxplots in Fig. 4 are drawn using the test RSEs obtained by

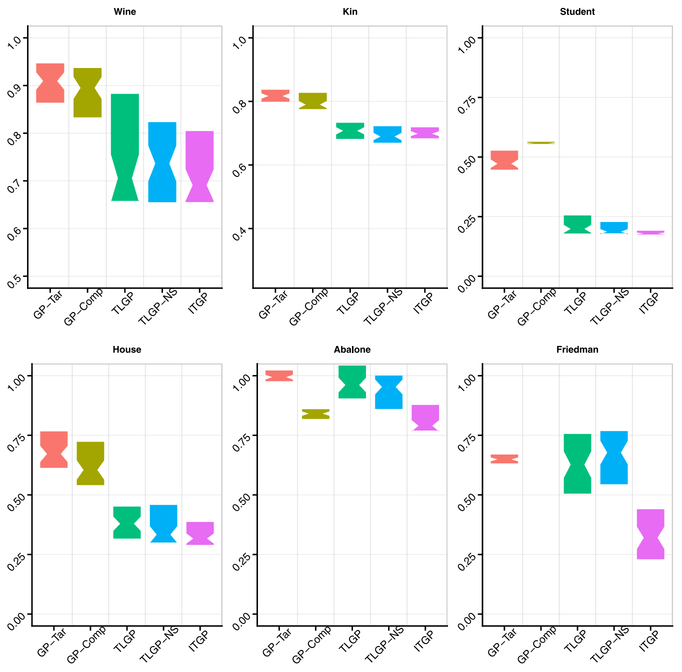

Fig. 4. Distribution of test RSEs in GP methods on the *Target* domain.

the evolved models of the 50 GP runs in each method. Each box shows the distribution of the 50 test RSEs for each GP method. Specifically, ITGP has notable improvement over GP-Tar on all the test sets, which is indicated by the large distance between the boxes of the two methods on Student, House, and Friedman, and the nonoverlap of the boxes on the other three test sets in Fig. 4.

Comparisons Among GP Methods: On all six test sets, the other two GP methods equipped with the instance weighting framework, that is, TLGP and TLGP-NS, also have a much smaller RSE value than GP-Tar, which indicates that the instance weighting framework is able to transfer useful knowledge to improve the generalization performance of GP. On Friedman, TLGP, and TLGP-NS can have a smaller test error than GP-Tar, but they are not very stable, which can be shown by a much longer box than in GP-Tar. On Abalone, TLGP, and TLGP-NS outperform GP-Tar slightly but no significant difference has been found between them. On Wine, Kin, Student, and House, TLGP and TLGP-NS have a notable advance over GP-Tar on the test performance, which is shown by the much obvious distance between their boxes. The differences are all significant. On five of the six test sets, TLGP-NS which differs from TLGP only on the selection operators, can improve the test performance of TLGP to some extent with regard to the smaller median test RSE and the distribution of the test errors.

On the other hand, compared with GP-Tar, GP-Comb can have better test performance on four datasets, that is, Wine, Kin, House, and Abalone. On Abalone, GP-Comb has an obvious advantage over GP-Tar, which is significant. On the other three datasets, GP-Comb only outperforms GP-Tar slightly but not significantly. On Student, GP-Comb has a slightly worse test performance than GP-Tar, while on Friedman, the test error of GP-Comb is much higher than that in GP-Tar, which is significant. The comparison between the test performance of GP-Tar and GP-Comb indicates that without any transfer learning component, it is difficult for learning methods to learn useful knowledge from the source-domain data directly to help learning in the target-domain data. On all the six test sets, the proposed method ITGP can improve the performance of GP-Comb significantly. The advantage is definitely brought by the instance weighting and selection component, which help to identify the more useful source-domain instances and lead to a better cross-domain generalization performance. Note that this article considers the scenarios when the distributions of the source data and the target data are different but similar where the similarity of the distributions could be estimated by various quantitative measures. The examined transfer learning methods are not designed to transfer well when the source domain is very different from the target domain.

GP Versus SVR: It can also be seen from Table III that on all the test sets, comparing with SVR, GP methods consistently outperform the corresponding SVR methods on all the test sets. The pattern is different from that on the training sets. The reason is the overfitting occurs in SVR which is indicated by the huge difference between the performance on the test sets and the training sets. This is particularly the case in SVR-Comb when the regression model is learned from a combination of the source- and target-domain data. With a set of weighted source-domain instances, SVR-W can generalize better. SVR-W has a worse test performance than SVR-Tar on House and Abalone. However, on the other four of the six datasets, it can improve the generalization performance of SVR on the target domain. This also confirms that the weights provided by KMM could benefit the learning on the target domain. However, due to the difficulty in tuning the parameters (e.g., the bandwidth in KMM) for each problem, the negative effort might also happen. That is why KMM can provide a good start point in the proposed instance weighting framework but a further search (such as by DE in the new instance weighting framework) for the optimal weights is necessary.

The Influence of Number of Instances: On datasets where there is an insufficient number of instances in the target domain, for example, in Student, House, and Friedman, it is extremely difficult for GP-Tar to generalize well. In this case, ITGP has a notable advantage over GP-Tar, and the difference between their test performance is much more obvious than on the other datasets. Meanwhile, in this case, directly using the source-domain data might even increase the test error, that is, both GP-Comb and SVR-Comb have a worse generalization error than GP-Tar and SVR-Tar. The case is even worse on Friedman where the distribution on the source and the target domains is more different. This indicates the distribution similarity is also a key component to determine the effectiveness of the source-domain instances and also confirms the effective transfer learning ability of the new instance weighting framework.

Another important pattern is that, on the datasets which have the largest number of source-domain instances, that is, Wine and Kin, where there are over thousand source-domain instances, the new method ITGP equipped with an instance selection method could also notably outperform GP-Tar. This is an advance over our previous method TLGP. Comparing TLGP with GP-Tar, there is a smaller difference between their performance on the test performance. This is due to an over-large number of source-domain instances that might make it difficult to detect and transfer the source-domain knowledge due to the larger search space, which limits the searchability of DE and leads to a less effective transfer of knowledge in TLGP. Meanwhile, it also makes the proportion of the target training instances too small, so that the learning is dominated by the source-domain data. The new method ITGP performs instance selection base on the initial weights provided by the density ratio estimation, which is able to shrink the search space and reduce the proportion of source-domain data. Meanwhile, the initial weights provide good start points for the search of DE, which could make it more effective and efficient.

3) Evolutionary Plots on the Test Sets: To better examine the generalization capacity of GP methods, the evolutionary plots of the models in the five GP methods on the *test* data are shown in Fig. 5. At each generation, the test RSE of the best-of-generation solution is recorded and these plots are drawn using these best-of-generation RSEs. Note that the evolutionary process does not consider these errors. They are only used for the analysis purpose.

Fig. 5 clearly shows that ITGP has better generalization capability than the other four GP methods regarding reducing the trend of overfitting (on House, Abalone, and Friedman) and having more stable test performance among different GP runs. The stableness is shown in the much narrower band and the disappearance of outliers other GP methods (e.g., the final two/three generations in TLGP and TLGP-NS on House). ITGP advances most of the GP methods from an early stage of the evolutionary process, which is usually within the first several generations except for Student. On Kin, Abalone, and Friedman, ITGP outperforms TLGP and TLGP-NS from the early stage, which confirms the initial weight vectors in ITGP are better than the randomly generated ones in TLGP and TLGP-NS.

In the synthetic dataset, that is, Friedman, which has a notable difference between the distributions of the source and target domains, a notable pattern can be found. While GP-Comb does not generalize at all (note that it has an increasing generalization error after one or two generations' decreasing). During the evolutionary process, the test RSEs of GP-Comb on Friedman increased from the very beginning generations, ITGP can still generalize well. Comparing with the other two GP methods, ITGP also has better generalization performance over generations, which confirms the advance of the new instance weighting framework on acquiring more useful knowledge from the source data.

For the other GP methods, GP-Tar usually converges early, thus it is difficult for GP-Tar to have a decreasing test error after a few generations. It also suffers from overfitting on House. Compared with the other datasets, House has a much smaller number of target training instances, which might not enough to represent the true pattern in the target domain. In these cases, instances from some similar domains are more effective to help. GP-Comb, which uses the source-domain instances directly, can also easily improve the generalization

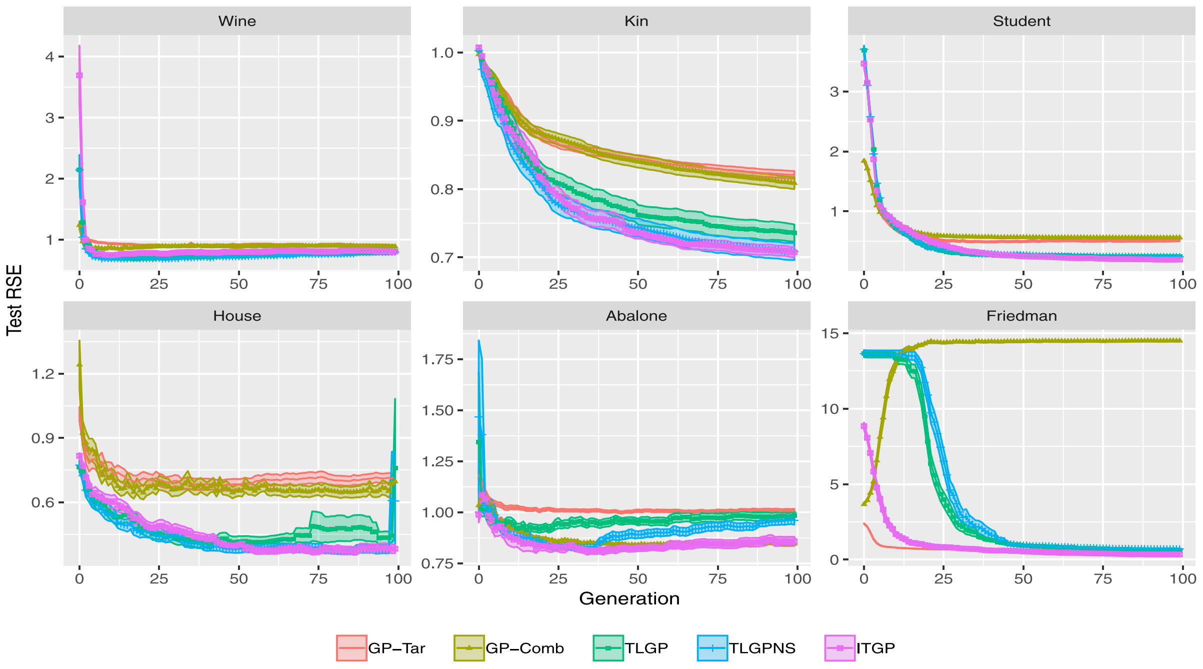

Fig. 5. Evolutionary plots (with 95% confidence interval) on the test sets.

performance of GP on the target domain without any transfer learning component.

B. Analysis of Evolved Models

A further analysis was taken on examining the models evolved by the five GP methods. We randomly picked one from the 50 GP runs of each method (i.e., the same run for each method) on the synthetic dataset, that is, Friedman, where the target function is known. The evolved models are presented as well as their mathematically simplified form for easy comparing their behavior in Table IV. As it shows, none of the GP methods can evolve models having the same structure as the target function for Friedman, which is a difficult regression problem. Among the five GP methods, the model evolved by ITGP has a similar number of features with the target model and contains some large values for the parameters, which is similar to the distribution of the target-domain data. Moreover, the models evolved by ITGP are much smoother in both the structure and the behavior. All these observations confirm and explain why models evolved by ITGP can have a better prediction/generalization performance on the target test data to some extent.

C. Further Examination on the Program Size and Computation Cost of GP Methods

To understand the influence of the new transfer learning method on the size of the learned model and the computational cost of GP, Table V shows the mean and smallest program size counting the number of nodes in the best-of-run trees in the five GP methods. Note that in the column of "Size," "(Best)" stands for the smallest programs. The overall computational cost of the GP runs is also examined in terms of the average and the shortest running time for one run. Here, (Best) in the "Time" column refers to the shortest GP run.

It is clear that GP with instance weighting frameworks typically evolves the smallest trees. These trees are much smaller than those in the other two GP methods in all the examined problems. The smaller/simpler models in the two TLGP methods are because that DE performed an implicit instance selection when the search for the optimal weights. The evolutionary process in TLGP generally learns from a dynamic and smaller training sets. The explicit instance selection process in ITGP helps further reduce the size of the evolved models. Models in ITGP have the smallest size among the five methods.

Considering the computational cost, compared with the other three GP methods, which also learn from a combination of the source and the target data, ITGP spends notably less computational cost than GP-Comb on three of the six datasets, while on the other three datasets, it spends long time to evolve models than GP-Comb. Compared with TLGP and TLGP-NS, ITGP consistently spends much shorter computational time on all the six datasets. TLGP and TLGP-NS usually spend a much more computational time. This is due to the additional effort on searching for the weight vector, the more complex evaluations and selection processes. However, with instance selection, the cost of the instance weight framework is decreased hugely. Thus, the computational time of ITGP is much less than its two counterparts.

Comparing with GP-Tar, which only learns from the target data, in general, ITGP spends more computational time than GP-Tar. However, it is affordable for GP to use the new instance weighting framework, which generally takes less than 1 min for one GP run. Clearly, comparing with the notable cross-domain generalization gain and the much

TABLE IV
EXAMPLE OF BEST-OF-THE-RUN MODELS FOR PROBLEM
$Y = a_1 \times 10\sin(\pi(b_1x_1 + c_1) \times (b_2x_2 + c_2)) + a_2 \times 20(b_3x_3 + c_3 - 0.5)^2 + a_3 \times 10(b_4x_4 + c_4) + a_4 \times 5(b_5x_5 + c_5) + N(0, 1)$

Method	Evolved Model	Simplified Model
GP-Comb	$((((((0.76--0.96)+((X5+X3)+(0.69+X1)))/((X8+X9)/((X8+X4)+((X8+X9)/(-0.86-X10)))))*((((-0.86-X10)/(X8+X9))/((X5/X4)*-0.98))+((-0.62+(X8*-0.62))-((0.76--0.96)/((X8*-0.62)+(X8*-0.62))))))-X4)+((-1.0-X9)+-0.19))*(((((0.87-X1)*((0.76--0.96)/(X5/X4)))+((((X8+X9)/(-0.86-X10))+X3)+(0.69+X1)))+(-0.19--0.87))+0.69)$	$(X5*X8*(X10+0.86)*(X8+X9)**2*(X4+X9+1.19)+(X8+X9-(X10+0.86)*(X4+X8))*(1.02*X4*X8*(X10+0.86)-0.62*X5*X8*(X8+1)*(X8+X9)+1.38*X5*(X8+X9))*(X1+X3+X5+2.41))*(1.72*X4*(X1-0.87)*(X10+0.86)-X5*(X10+0.86)*(X1+X3+2.06)+X5*(X8+X9))/(X5**2*X8*(X10+0.86)**2*(X8+X9)**2)$
GP-Tar	$(((((0.49+X1)+X4)*((0.49+X1)+((0.49+X8)+X4)))+((-0.58/(((X1*0.88)+(X3/-0.22))-(((-0.27-X10)--0.07)+((0.49+X1)+(0.84+0.09)))))+(((X7*X4)+(X7*(X7*(-0.58/X5))))+(X8*X2))))+(((0.84+(0.37+X5))+((0.6+(((X10*X8)+X7)*X2))*(0.49+(0.84+(0.84+(0.49+X1))))))+((((X7*X4)+(X7*(-0.58/X5)))+((0.4+X7)*X2))*(((0.49+X8)+(0.49+X8))+(0.84+((0.49+X1)+X5))))))+(((0.82+X5)+X4)*((0.49+X1)+(0.84+0.09)))$	$(X5*(0.12*X1-X10+4.54*X3+1.22)*(X2*X8+X4*X7+X5+(X1+1.42)*(X4+X5+0.82)+(X1+2.66)*(X2*(X10*X8+X7)+0.6)+(X1+X4+0.49)*(X1+X4+X8+0.98)+1.21)+0.58*X5+(-0.58*X7**2+(X5*(X2*(X7+0.4)+X4*X7)-0.58*X7)*(X1+X5+2*X8+2.31))*(0.12*X1-X10+4.54*X3+1.22))/(X5*(0.12*X1-X10+4.54*X3+1.22))$
TLGP	$(((X4+((0.6+(((0.27+X10)--0.97)+(X4+X4)))+((((X1--0.97)/0.11)+-0.25)--0.75)))+(-0.79-X7))*((X7+X1)+((X8+0.36)+0.25)))+(((((-0.25/-0.98)+(((X7--0.83)/(X1+X2))+(X10/X3)))+-0.89)-((-0.57+(-0.41-0.74))-(X8*0.72)))+((X4+(X2--0.43))*(X7+((0.6+X3)+(-0.43+(X2*0.31))))))$	$(X10*(X1+X2)+X3*(X1+X2)*(0.72*X8+(X2+X4+0.43)*(0.31*X2+X3+X7+0.17)+(X1+X7+X8+0.61)*(9.09*X1+X10+3*X4-X7+10.37)+1.08)+X3*(X7+0.83))/(X3*(X1+X2))$
TLGP-NS	$(((X4+X4)+((X1/-0.81)+((-0.36+((0.98+X1)/(X2--0.46)))+X4)))*((((X5-(-0.08-(0.93+X2)))+(0.68+0.18))/((X4+(-0.23*X8))-(-0.67+X8)))-(-0.83+(((0.18*X1)-((0.93+X2)+(X5*X1)))+0.2))))+(((((0.93+X2)+(X5*X1))+((0.93+X2)+(-0.61-X6)))-(X4+(-0.08-(0.93+X2))))-(((-0.91-0.25)+X3)+(X4+X1)))$	$((X2+0.46)*(X4-1.23*X8+0.67)*(X1*X5-X1+3*X2-X3-2*X4-X6+3.42)-(-X1+(X2+0.46)*(1.237*X1-3*X4+0.36)-0.98)*(X2+X5+(X4-1.23*X8+0.67)*(X1*X5-0.18*X1+X2+1.56)+1.87))/((X2+0.46)*(X4-1.23*X8+0.67))$
ITGP	$(((((X5--0.98)-(((X6-X7)*X5)+(X9-X2)))+((X3/-0.9)/(X2+((X5--0.54)+X3))))+(X1+((X2+((X3/-0.9)/0.62))+X2)))+((X3+X2)+((((1.0+X4)+(0.97+0.8))+((-0.71*X3)+(X5--0.98)))+X1)))*(1.0+X4)$	$(-1.11*X3+(X2+X3+X5+0.54)*(2*X1+4*X2-1.50*X3+X4-X5*(X6-X7)+2*X5-X9+4.73))*(X4+1.0)/(X2+X3+X5+0.54)$

simpler models with a better interpretability in ITGP, it is worth to pay a small price on the increase of training time.

VI. FURTHER COMPARISONS WITH TRANSFER LEARNING METHODS

To further investigate the transfer learning ability of the proposed method, a set of experiments has been conducted to compare ITGP with two state-of-the-art boosting-based transfer learning methods for regression. The two methods are TrAdaBoost and two-stage TrAdaBoost using a simple neural network as the basic learner. The key idea of TrAdaBoost is introduced in Section II-B. For more details, readers are referred to the original paper [17].

TrAdaBoost, two-stage TrAdaBoost, and ITGP are compared on two datasets *SARCOS*[3] and *UjiIndoorLoc* [35], which are commonly used in multitask learning. The information of the two datasets is summarized in Table VI. Eight tasks are generated from the two datasets. A detailed description of the generated tasks is presented in the online supplementary material[4] of this article. Each of the three methods has conducted 30 runs on each task. The Wilcoxon test with a significance level of 0.05 is conducted to compare the test RSEs of ITGP and the two TrAdaBoost methods in pairs.

The median and the standard deviation of the RSEs on the target test sets are summarized in Table VII. It is clearly shown that ITGP has a notably smaller test RSEs than the two TrAdaBoost methods on seven of the eight tasks except on UjiIndoor-Lo. The difference between ITGP and the two TrAdaBoost methods are all significant on these seven tasks. On UjiIndoor-Lo, ITGP has a significantly higher RSE than the two-stage TrAdaBoost method. There is no significant difference between TrAdaBoost and ITGP. Compared with the two TrAdaBoost methods, which are ensemble transfer learning methods using an advance instance weighting framework, ITGP still has an impressive advantage. Meanwhile, it also needs to point out that the advantage might partly be brought by GP, which is much stronger than the simple neural network used in the two TrAdaBoost methods. The results also confirm the effectiveness of the proposed instance weight framework on large-scale transfer learning tasks. Considering the computational cost, the new method spent a little longer time on training but the test time is shorter.

[3]http://www.gaussianprocess.org/gpml/data/
[4]http://homepages.ecs.vuw.ac.nz/∼chenqi1/Mywork/OLS-GPTL.pdf

TABLE V
NUMBER OF NODES IN PROGRAMS AND COMPUTATIONAL TIME

	Method	Size (#Nodes) Mean(Best)	Time (Second) Mean(Best)
Wine	GP-Tar	104(41)	4.35(1.86)
	GP-Com	94(31)	118.62(36.18)
	TLGP	66(5)	138.54(80.55)
	TLGP-NS	65(19)	120.81(79.47)
	ITGP	52(7)	20.98(9.43)
Kin	GP-Tar	87(10)	14.12(0.88)
	GP-Com	88(10)	151.79(8.39)
	TLGP	70(10)	189.01(65.16)
	TLGP-NS	68(10)	169.59(59.1)
	ITGP	61(9)	94.46(37.59)
Student	GP-Tar	119(49)	1.5(0.8)
	GP-Com	97(39)	8.52(4.53)
	TLGP	85(43)	65.65(51.82)
	TLGP-NS	81(33)	65.53(49.89)
	ITGP	88(41)	57.77(32.82)
House	GP-Tar	115(3)	0.59(0.25)
	GP-Com	103(19)	4.18(1.14)
	TLGP	59(19)	27.28(22.96)
	TLGP-NS	44(9)	31.65(19.74)
	ITGP	41(9)	19.52(11.56)
Abalone	GP-Tar	101(59)	1.78(1.14)
	GP-Com	96(35)	17.29(7.33)
	TLGP	60(15)	40.6(22.56)
	TLGP-NS	69(17)	45.2(23.21)
	ITGP	48(15)	37.95(28.6)
Friedman	GP-Tar	161(73)	1.21(0.66)
	GP-Com	152(71)	19.66(9.58)
	TLGP	82(43)	121.7(93.86)
	TLGP-NS	85(37)	123.5(32.55)
	ITGP	79(37)	14.02(12.02)

TABLE VI
SARCOS AND UJIINDOOR DATASETS

Name	#Features	#Source	#Target	
			Training	Test
SARCOS	21	44484	449	4000
UjiIndoor	520	19937	200	1798

TABLE VII
TEST RSES OF TRADABOOST, TWO-STAGE TRADABOOST, AND ITGP ON THE TARGET DOMAIN OF SARCOS AND UJIINDOORLOC

	TrAdaBoost Median(Std)	2-stage TrAdaBoost Median(Std)	ITGP Median(Std)
UjiIndoor-Lo	1.05(0.001)	0.993(0.0001)	1.022(0.13)
UjiIndoor-La	0.97(0.06)	0.237(0.005)	0.1(0.06)
$\text{SARCOS}_{1,2}$	0.505(0.01)	0.48(0.02)	0.17(0.14)
$\text{SARCOS}_{1,3}$	2.15(0.36)	2.05(0.23)	0.4(0.13)
$\text{SARCOS}_{2,1}$	0.32(0.09)	0.31(0.02)	0.14(0.019)
$\text{SARCOS}_{2,3}$	0.198(0.08)	0.17(0.04)	0.05(0.01)
$\text{SARCOS}_{3,1}$	1.42(0.26)	1.23(0.14)	0.57(0.11)
$\text{SARCOS}_{3,2}$	0.54(0.06)	0.53(0.02)	0.14(0.092)

VII. CONCLUSION

This article aimed to extend our recently proposed instance weighting framework (TLGP) for GPSR for transfer learning to achieve a more effective and efficient correction of the distribution difference between the source domain and the target domain. The goal has been achieved by introducing a density estimation method for the source-domain instance selection and the weight vectors initialization in DE. Thus, it helps shrink the search space and provide good start points for the search of the optimal weights for the source-domain instances. The proposed method ITGP achieved a significantly better cross-domain generalization performance than GP-Tar, GP-Comb, a widely used regression method SVR under various settings as well as TLGP. A further examination of the evolutionary process and the comparison with the other GP methods also show the effort of ITGP on reducing the overfitting trend in the learning process and achieving stable generalization performance among different runs. All these advantages confirm that compared with learning from an insufficient number of target-domain instances (GP-Tar and SVR-Tar) and utilizing the source-domain instances directly but without any transfer learning component (GP-Comb and SVR-Comp), and without the instance selection process (TLGP and TLGP-NS), the new framework which utilizes KMM and DE with a good optimization ability can guide GP to effectively utilize more useful source-domain instances and learn models with good cross-domain generalization performance. Furthermore, these more generalizable models typically are much simpler/smaller than their counterparts in GP-Tar, GP-Comb, and TLGP/TLGP-NS.

Apart from the very promising transfer learning ability achieved by ITGP, we focus mainly on instance selection and weight optimization in this article. We would like to extend ITGP to correct the distribution difference between the source and the target domains by mapping the feature space. Meanwhile, like many existing transfer learning methods, the transfer learning ability of ITGP might limit when the source and the target domains are too dissimilar. Leveraging the transfer learning ability of ITGP in this scenario is part of our future work. Moreover, this article only considers the transfer learning scenario where there is only one source domain, we will also further investigate the proposed instance weighting framework when the available source data comes from several domains. Transfer learning in this scenario seems more difficult and challenging, so further investigation is more interesting.

REFERENCES

[1] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in NLP," in *Proc. ACL*, 2007, p. 264.

[2] S. P. K. Karri, D. Chakraborty, and J. Chatterjee, "Transfer learning based classification of optical coherence tomography images with diabetic macular EDEMA and dry age-related macular degeneration," *Biomed. Opt. Exp.*, vol. 8, no. 2, pp. 579–592, 2017.

[3] H. Lei *et al.*, "A deeply supervised residual network for HEP-2 cell classification via cross-modal transfer learning," *Pattern Recognit.*, vol. 79, pp. 290–302, Jul. 2018.

[4] M. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jul. 2009.

[5] U. Côté-Allard *et al.*, "Deep learning for electromyographic hand gesture signal classification using transfer learning," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 4, pp. 760–771, Jul. 2019.

[6] F. Knoll, K. Hammernik, E. Kobler, T. Pock, M. P. Recht, and D. K. Sodickson, "Assessment of the generalization of learned image reconstruction and the potential for transfer learning," *Magn. Reson. Med.*, vol. 81, no. 1, pp. 116–128, 2019.

[7] T. T. H. Dinh, T. H. Chu, and Q. U. Nguyen, "Transfer learning in genetic programming," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2015, pp. 1145–1151.

[8] E. Haslam, B. Xue, and M. Zhang, "Further investigation on genetic programming with transfer learning for symbolic regression," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2016, pp. 3598–3605.

[9] D. O'Neill, H. Al-Sahaf, B. Xue, and M. Zhang, "Common subtrees in related problems: A novel transfer learning approach for genetic programming," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2017, pp. 1287–1294.

[10] L. L. Minku, "Transfer learning in non-stationary environments," in *Learning from Data Streams in Evolving Environments*. Cham, Switzerland: Springer, 2019, pp. 13–37.
[11] Q. Chen, B. Xue, and M. Zhang, "Differential evolution for instance based transfer learning in genetic programming for symbolic regression," in *Proc. Genet. Evol. Comput. Conf. Companion (GECCO)*, 2019, pp. 161–162.
[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[13] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowl. Based Syst.*, vol. 80, pp. 14–23, May 2015.
[14] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 601–608.
[15] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1433–1440.
[16] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. ACM 24th Int. Conf. Mach. Learn.*, 2007, pp. 193–200.
[17] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 863–870.
[18] N. D. Lawrence and J. C. Platt, "Learning to learn with the informative vector machine," in *Proc. ACM 21st Int. Conf. Mach. Learn.*, 2004, p. 65.
[19] A. T. W. Min, R. Sagarna, A. Gupta, Y.-S. Ong, and C. K. Goh, "Knowledge transfer through machine learning in aircraft design," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 48–60, Nov. 2017.
[20] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, Jun. 2016.
[21] L. Feng *et al.*, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3457–3470, Sep. 2019.
[22] R. Chandra and S. Cripps, "Coevolutionary multi-task learning for feature-based modular pattern classification," *Neurocomputing*, vol. 319, pp. 164–175, Nov. 2018.
[23] R. Chandra, Y.-S. Ong, and C. Goh, "Co-evolutionary multi-task learning for dynamic time series prediction," *Appl. Soft Comput.*, vol. 70, pp. 576–589, Sep. 2018.
[24] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, doi: 10.1109/TSMC.2018.2853719.
[25] K. Price and R. Storn, "Differential evolution: A simple evolution strategy for fast optimization," *Dr. Dobb's J.*, vol. 22, no. 4, pp. 18–24, 1997.
[26] V. Kachitvichyanukul, "Comparison of three evolutionary algorithms: GA, PSO, and DE," *Ind. Eng. Manag. Syst.*, vol. 11, no. 3, pp. 215–223, 2012.
[27] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[28] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.
[29] S. Luke *et al.* (2004). *A Java-Based Evolutionary Computation Research System*. Accessed: Mar. 2004. [Online]. Available: http://cs.gmu.edu/˜eclab/projects/ecj
[30] D. Meyer and F. Wien, "Support vector machines," *R News*, vol. 1, no. 3, pp. 23–26, 2001.
[31] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Fuzzy regression transfer learning in Takagi–Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1795–1807, Dec. 2017.
[32] M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[33] J. Friedman, "Multivariate adaptive regression splines," *Ann. Stat.*, vol. 19, no. 1, pp. 1–67, 1991.
[34] Y. Rahmatallah, B. Zybailov, F. Emmert-Streib, and G. Glazko, "GSAR: Bioconductor package for gene set analysis in R," *BMC Bioinformatics*, vol. 18, no. 1, p. 61, 2017.
[35] J. Torres-Sospedra *et al.*, "UjiindoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," in *Proc. IEEE Int. Conf. Indoor Position. Indoor Navig. (IPIN)*, 2014, pp. 261–270.

Qi Chen (Member, IEEE) received the B.E. degree in automation from the University of South China, Hengyang, China, in 2005, the M.E. degree in software engineering from the Beijing Institute of Technology, Beijing, China, in 2007, and the Ph.D. degree in computer science from the Victoria University of Wellington, Wellington, New Zealand, in 2018.

Since 2014, she has been with the Evolutionary Computation Research Group, Victoria University of Wellington, where she is currently a Postdoctoral Research Fellow with the School of Engineering and Computer Science. Her current research mainly focuses on genetic programming for symbolic regression. Her research interests, include machine learning, evolutionary computation, feature selection, feature construction, transfer learning, domain adaptation, and statistical learning theory.

Dr. Chen serves as a Reviewer for international conferences, including the IEEE Congress on Evolutionary Computation, and international journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.

Bing Xue (Member, IEEE) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science from the Victoria University of Wellington, Wellington, New Zealand, in 2014.

She is currently an Associate Professor with the School of Engineering and Computer Science, Victoria University of Wellington. She has over 200 papers published in fully refereed international journals and conferences. Her research focuses mainly on evolutionary computation, feature selection, feature construction, image analysis, and transfer learning.

Dr. Xue is currently the Chair of the IEEE Computational Intelligence Society (CIS) Data Mining and Big Data Analytics Technical Committee and the IEEE Task Force on Evolutionary Feature Selection and Construction, and the Vice-Chair of the IEEE CIS Task Force on Transfer Learning and Transfer Optimization and the IEEE CIS Task Force on Evolutionary Deep Learning and Applications.

Mengjie Zhang (Fellow, IEEE) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from the Royal Melbourne Institute of Technology, Melbourne, VIC, Australia, in 2000.

He is currently a Professor of computer science, the Head of the Evolutionary Computation Research Group, and an Associate Dean (Research and Innovation) with the Faculty of Engineering, Victoria University of Wellington, Wellington, New Zealand. He has published over 500 research papers in refereed international journals and conferences. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job shop scheduling, and transfer learning.

Prof. Zhang was the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, the IEEE CIS Emergent Technologies Technical Committee, and the Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is the Vice-Chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction and the Task Force on Evolutionary Computer Vision and Image Processing, and the Founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a Committee member of the IEEE NZ Central Section. He is a fellow of the Royal Society of New Zealand and have been a Panel Member of the Marsden Fund (New Zealand Government Funding). He is a member of ACM.