2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

A Clustering-based Support Vector Classifier for Dynamic Time-Linkage Optimization

Meng Gao, Xiao-Fang Liu*
Institute of Robotics and Automatic Information Systems
College of Artificial Intelligence
Nankai University
Tianjin, China
liuxiaofang@nankai.edu.cn

Zhi-Hui Zhan
School of Computer Science and Engineering
South China University of Technology
Guangzhou, China
zhanapollo@163.com

Jun Zhang
Hanyang University
Ansan, South Korea
Nankai University
Tianjin, China
junzhang@ieee.org

***Abstract*—Dynamic time-linkage optimization problems (DTPs) bring challenges to existing evolutionary algorithms due to the influence of a current decision in the future. Existing methods usually model the rewards of a current decision in the future for prediction. However, these methods often present low prediction accuracy due to the lack of sufficient training data. In addition, they often require a long computational time. To address these issues, the problem of predicting rewards is converted into a simpler binary classification problem, which evaluates whether a current solution can bring positive or negative influence in the future. This paper proposes a clustering-based support vector classifier for solution evaluation. In the proposed method, the density of the time-linkage property is detected first. Historical data are divided using k-means clustering so as to train a support vector classifier for solution evaluation. Good solutions are selected to generate a final decision solution using a crossover operator. Integrating the clustering-based support vector classifier into particle swarm optimization, a new method named CSVC-PSO is put forward. Multiple instances are constructed using a recent DTP test suite with different types of time-linkage patterns and density. Experimental results demonstrate that the proposed CSVC-PSO outperforms state-of-the-art algorithms on most instances using a shorter time.**

***Index Terms*—evolutionary computation, time-linkage, dynamic optimization, support vector machine**

I. Introduction

Dynamic time-linkage optimization problems (DTPs) [1] widely exist in many real-world applications [2], in which the problem parameters change over time and a current decision solution influences future environments. For example, in a vehicle navigation system, if all vehicles use the same "optimal" route on the same road, the route may no longer be optimal for the entire system in the future [3]. These problems challenge existing evolutionary algorithms due to the change of the optimum and the influence of a current decision in the future.

In the literature, multiple methods are developed to track the changing optima in dynamic environments [4]. The methods can be roughly classified into two types, i.e., diversity enhancement (e.g., [5]) and prediction (e.g., [6]). Particularly, diversity enhancement methods increases population diversity by adding new solutions [7]. In contrast, prediction methods model the changing patterns of the optima for predicting new optima in new environments [6] or shift historical solutions to new environments using transfer learning techniques [8], [9]. These methods have shown good performance on dynamic optimization problems. However, they ignore the time-linkage property and present poor performance on DTPs. Mathematical analysis has demonstrate that it is challenging to solve DTPs using traditional evolutionary algorithms [10]–[12].

To deal with the time-linkage property of DTPs, some methods aim to learn a model for predicting the influence of a current solution in the future [13], [14]. These prediction methods train a surrogate model from the data of historical environments using machine learning techniques. Based on the prediction and the current fitness values, they select a good solution with good performance in both the present and the future. For example, an evolutionary algorithm with a predictor builds a predictor using historical data for optimizing DTPs in [13], but the method presents a large prediction error. The prediction accuracy is further introduced for decision making for improving algorithm performance [14]. To simulate the long-term influence of decision solutions, reinforcement learning is adopted to learn the rewards using surrogate models [15]. However, the method requires a long computational time and may perform poorly due to the lack of sufficient training data.

According to the above discussion, existing methods may present a large prediction errors due to the lack of training data. An alternative method is to build a simpler model for a simple target. Thus, this paper proposes particle swarm optimization with a clustering-based support vector classifier, named CSVC-PSO. Particularly, the clustering-based support vector classifier is developed to evaluate whether a solution has positive or negative influence in the future, rather than the real reward value of each solution. In this way, the evaluation problem is converted into a simpler binary classification problem. In addition, a time-linkage detection method is designed to estimate the density of the dependence of decisions between environments. If the dependence is weak, then the

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62103202, in part by the Natural Science Foundation of Tianjin under Grant 21JCQNJC00140, and in part by the Fundamental Research Funds for the Central Universities, Nankai University (Grant 63231162). (*Corresponding author: Xiao-Fang Liu*)

best solution found by PSO is taken as decision; otherwise, a new solution is constructed by a crossover operator to combine the current fitness value and the future influence. Multiple instances are construed based on the very recent DTP test suite [15], with different types of dependence patterns and density. Experimental results show that the proposed method outperforms state-of-the-art algorithms on most instances. In addition, the proposed method requires a shorter computational time. The contributions of this paper are as follows:

- A new time-linkage detection method is developed to determine whether the dependence between decision solutions is strong. This helps the algorithm to select decision schemes.
- A new clustering-based support vector classifier is proposed to evaluate whether a solution has a positive or negative reward in the future. Compared with existing methods to directly estimate the reward of each solution, the proposed method is able to avoid a large prediction error under limited data.
- A new crossover operator constructs a final solution by considering the fitness value in the current environment and the possible reward in the future.

The rest of this paper is organized as follows. Section II provides the definition of DTP. Section III introduces the proposed method. Section IV presents experimental results. Finally, conclusions are drawn in Section V.

II. Dynamic Time-linkage Optimization

A DTP can be formulated as

$$\begin{aligned} \max \quad & \sum_{t=1}^{t^{end}} f(\mathbf{x}_t, \gamma_t(X_{t-1}^*)) \\ s.t. \quad & \mathbf{x}_t \in \Gamma. \end{aligned} \tag{1}$$

where t^{end} represents the number of time steps, $\mathbf{x}_t$ denotes a solution available at time step t, X_{t-1}^* is the set of decision solutions selected up to time $t-1$, γ_t represents the problem parameters at time step t; and Γ is the search space. The optimization objective of a DTP is to find solutions with the maximum cumulative fitness value from $t=1$ to $t=t^{end}$.

III. Method

In this section, CSVC-PSO is introduced. The flowchart of CSVC-PSO is illustrated in Fig. 1. CSVC-PSO consists of four parts: particle swarm optimization, time-linkage detection, clustering-based support vector classifier, and crossover. Particularly, PSO is employed to search good solutions with the best fitness value in the current environment. During the searching procedure, the solutions found are stored in an external archive P for future utilization. Then the time-linkage detection is performed to determine the density of the dependence between decisions in dynamic environments. If the dependence is weak, then decision is made based on the current fitness values only; otherwise, the future influence of solutions are evaluated using a clustering-based support vector classifier, and a decision is made using a crossover

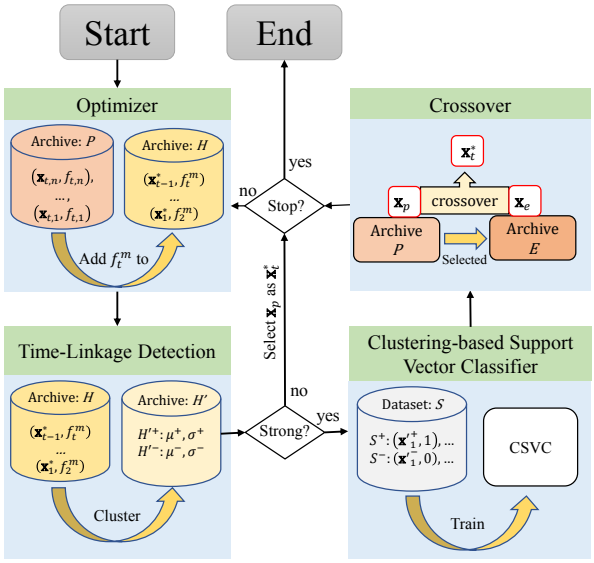


Fig. 1: Flowchart of CSVC-PSO

operator based on both current fitness value and future rewards. Once an environmental change occurs, new solutions are generated to make decisions using the same procedure. The procedure continuous until meeting the terminating conditions. The details of each component in CSVC-PSO are introduced in the following.

A. Particle Swarm Optimization

Denote the current environment index as t. Since PSO [16] has shown good performance in dynamic environments, it is adopted as the optimizer to search for good solutions in each environment. A swarm is randomly initialized first. In every iteration, the velocity $\mathbf{v}_i$ and position $\mathbf{x}_i$ of each particle i are updated as

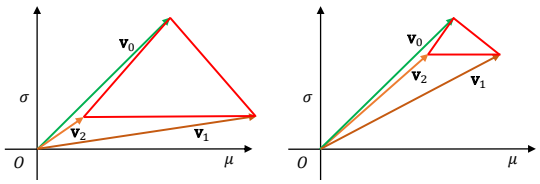
$$\begin{aligned} \mathbf{v}_i &= w\mathbf{v}_i + c_1 r_1 (Pbest_i - \mathbf{x}_i) + c_2 r_2 (Gbest - \mathbf{x}_i), \\ \mathbf{x}_i &= \mathbf{x}_i + \mathbf{v}_i. \end{aligned} \tag{2}$$

where w is the inertia weight, c_1 and c_2 are acceleration coefficients, r_1 and r_2 are random numbers in a range of [0,1], $Pbest_i$ is the historically best position found by particle i, and $Gbest$ is the global best position of the swarm. If the updated position is better than $Pbest_i$, then the $Pbest_i$ is updated. Through multiple iterations, the swarm finds a good solution in the current time step. During the iteration procedure, all the positions found by the swarm are stored in an external archive P. The solutions in P will be selected for decision making. Particularly, the best solution in P is denoted as $\mathbf{x}_p$ and its fitness value is denoted as f_t^m. More details of how to use the archives are given in the following subsections.

In the first four environments ($t < 5$), the best solution found by PSO is taken as the final decision to collect data.

B. Time-Linkage Detection

The time linkage is detected before decision making. If the time linkage is weak, a current decision has little influence in



(a) Large dissimilarity index (b) Small dissimilarity index

Fig. 2: Examples of the dissimilarity index

the future. In such a case, choosing the best solution based on the current fitness value is a good choice. Otherwise, decision making should also consider the future rewards of solutions.

Indeed, we can use the dependence between a decision solution in an environment and the available best fitness value in the next environment to represent the time linkage. Hence, a related sample can be constructed by pairing a decision $\mathbf{x}_j^*$ in j-th environment and the best fitness value f_{j+1}^m found by PSO in the next $(j+1)$-th environment, i.e., $(\mathbf{x}_j^*,\ f_{j+1}^m)$, where $1 \leq j < t$. These samples are stored in an archive H, which is used to learn the dependence between decision solutions of the problem. Particularly, $H = \{(\mathbf{x}_1^*, f_2^m), (\mathbf{x}_2^*, f_3^m), ..., (\mathbf{x}_{t-1}^*, f_t^m)\}$.

If the time linkage is strong, then the difference between fitness values in multiple environments will be large; otherwise, the difference will be small. Inspired by this, we propose to detect the dependence by checking the difference among the best fitness values found in previous environments. To achieve, the samples in H are divided into two subsets using k-means clustering based on the fitness value f_j^m first. The subset with better fitness values are called a positive set and another one is called a negative set. Then, the mean and variance of the archive H, positive set, and negative set are calculated as a two-dimensional vector $\mathbf{v} = (\mu, \sigma)$, in which μ is the mean value of all fitness values in a set and σ represents the variance. The three corresponding vectors are denoted as $\mathbf{v}_0 = (\mu_0, \sigma_0)$, $\mathbf{v}_1 = (\mu_1, \sigma_1)$, and $\mathbf{v}_2 = (\mu_2, \sigma_2)$, respectively.

Since the positive and negative sets are divided from H, one of them has a large mean value than that of H and one has a smaller value. To measure the differences between two subsets, a dissimilarity index is defined as

$$I_d = \frac{\| \mathbf{v}_0 - \mathbf{v}_1 \| + \| \mathbf{v}_0 - \mathbf{v}_2 \| + \| \mathbf{v}_1 - \mathbf{v}_2 \|}{\| \mathbf{v}_0 \| + \| \mathbf{v}_1 \| + \| \mathbf{v}_2 \|} \quad (3)$$

where $\|\mathbf{v}_0\|$ is the modulus length of the vector $\mathbf{v}_0$. An example of the dissimilarity between subsets is illustrated in Fig. 2. The x-axis and y-axis represent the mean value and variance of a set. As shown in Fig. 2, the dissimilarity between two subsets is strongly correlated to the perimeter of the triangle formed by the three vector. Thus, the dissimilarity index can well evaluate the density of the time linkage between environments. If I_d is larger than a predefined threshold δ, then it is considered that the time linkage between environments is strong; otherwise, it is weak.

C. Clustering-Based Support Vector Classifier

To evaluate the future rewards of solutions, existing methods tend to build a model for predicting the accumulated rewards of solutions in a long term. However, these methods present large prediction errors due to the lack of enough training data. Instead of predicting the rewards using regression models, coarse evaluation may be more accurate under limited data. In this paper, we propose a clustering-based support vector classifier (CSVC) to evaluate whether a solution can bring positive influence or negative impacts in the future. That is, solutions are only classified into two clusters. The solution evaluation problem is converted into a binary classification task.

1) Construct Training Data: To train a classifier, training data is constructed first. Since the classifier aims to predict the future influence of solutions, training samples are constructed by pairing the decision solution $\mathbf{x}_j^*$ at j-th environment and the best fitness value obtained in the next $(j+1)$-th environment. That is, the data in H is used for learning. The positive set and negative set obtained during the time-linkage detection indeed are the positive and negative samples for classification. Denote the positive set and the negative one as X^+ and X^-. Particularly, a sample $(\mathbf{x}^*, f) \in X^+$ in the positive set is converted into a new sample $(\mathbf{x}^*,\ 1)$ with a label of 1. In contrast, a sample $(\mathbf{x}^*, f) \in X^-$ in the negative set is converted to a new sample $(\mathbf{x}^*,\ 0)$ with a label of 0. All new samples forms a training set S.

2) Determine Relevant Variables: There are usually some irrelevant variables in the training samples, which increase training difficulties and decrease prediction accuracy [17]. In addition, the number of training samples is small and a small model may be better. Thus, relevant variables, denoted as I_r, are detected and selected to further reduce the number of variables. Denote the variable set as $V = \{x^1, x^2, ..., x^D\}$, here x^i represents the set of the i-th variable of all the solutions in H. In this paper, Pearson correlation coefficient [18] r is used to measure the correlation between a variable and the target in H. Denote the Pearson correlation coefficient of a variable x^i as r_i. Then the variables rank according to the r values in a descending order. To avoid the setting of a threshold, the differences between the r values of any two neighboring variables are calculated. Variables are selected one by one until meeting a variable that has the largest difference to the r value of the next one.

3) Support Vector Classifier: Based on the selected relevant variables, the training samples in S are reduced. The irrelevant variables are removed from each sample. The resultant samples are used to train a support vector classifier [19]. To deal with nonlinear mapping relationships, a nonlinear support vector machine is implemented with the typical radial basis function kernel. The procedure of training a support vector classifier is given in **Algorithm 1**.

D. Crossover for Decision Making

The trained classifier is used to evaluate whether a solution has a negative or positive impact in the next environment.

Algorithm 1 Training a Support Vector Classifier

Input: $H = \{(\mathbf{x}_1^*, f_2^m), (\mathbf{x}_2^*, f_3^m), ..., (\mathbf{x}_{t-1}^*, f_t^m)\}$
Output: a trained classifier

```
X^+, X^- ← perform K-means clustering on H;
S ← ∅;
for each e = (x^*, f) ∈ H do
    if e ∈ X^+ then
        e' = (x^*, 1);
    else
        e' = (x^*, 0);
    end if
    S = S ∪ {e'};
end for
I_r ← Relevant variable selection based on H;
for each (x^*, label) ∈ S do
    x' ← Select the relevant variables from x^* based on I_r;
    Replace (x^*, label) with (x', label);
end for
train a support vector classifier using S;
```

All candidate solutions in the archive P are evaluated using classifier and the ones with positive labels are considered as promising solutions. Among these solutions with positive labels, the one with the highest fitness value is denoted as $\mathbf{x}_e$. Since $\mathbf{x}_e$ is evaluated using relevant variable only, it may have poor values on the irrelevant variables. To address this issue, we utilize the information of the irrelevant variables in the best solution found by the particle swarm optimization (denoted as $\mathbf{x}_p$, the best solution in P). That is, the relevant variables of $\mathbf{x}_e$ and the irrelevant variables in $\mathbf{x}_p$ can be combined to construct a better decision solution. Particularly, both $\mathbf{x}_e$ and $\mathbf{x}_p$ have a relative good fitness value in the current environment, whereas $\mathbf{x}_e$ has also a positive reward in the future. To achieve this, a crossover operator is performed on $\mathbf{x}_e$ and $\mathbf{x}_p$ to generate a final decision solution $\mathbf{x}_t^* = (x_t^1, ..., x_t^D)$ as:

$$x_t^i = \begin{cases} x_e^i, & \text{if } i \in I_r, \\ x_p^i, & \text{otherwise.} \end{cases} \tag{4}$$

where x^i represents the i-th variable of $\mathbf{x}$. The values of relevant variables are set to be the same as $\mathbf{x}_e$, and other variables are set to be the same as those of $\mathbf{x}_p$. At the t-th time step, $\mathbf{x}_t^*$ is adopted as the decision.

IV. Experiment

A. Experimental Setting

1) Benchmark Problems: A recent dynamic time-linkage problem test suite [20] is adopted for test. Multiple instances with different time-linkage patterns are constructed [15]. Three time-linkage patterns are constructed, i.e., linear, sinusoidal and circular dependence, which represent linear or nonlinear relationships between the solutions and the rewards in the future. Particularly, the problems with linear time linkages are formulated as

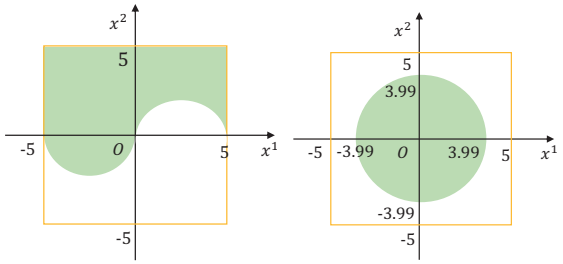
$$\begin{aligned} f(\mathbf{x}, t) &= \max_{i=1}^{m}\{h_i - w_i\|\mathbf{x} - \mathbf{c}_i\|\} + b_t, \\ s.t. \quad b_t &= \begin{cases} b, & x_{t-1}^1 \geq 0, \\ -b, & x_{t-1}^1 < 0, \end{cases} \\ & \mathbf{x} \in [-5, 5]^D. \end{aligned} \tag{5}$$

where h_i is the height of the i-th peak, w_i is the width of the i-th peak, $\mathbf{c}_i$ is the center of the i-th peak, and $f(\mathbf{x}, t)$ is used to find the solution with the highest fitness value. Additionally, b_t is the time-linkage coefficient. In the DTPs, the parameters h_i, w_i, and $\mathbf{c}_i$ vary over time but are not related to the previous decision solution in past environments. In contrast, b_t is only related to the decision solution $\mathbf{x}_{t-1}^*$ in the proceeding environment, resulting in the time-linkage property of the problem. Two other nonlinear dependence types to simulate sinusoidal and circular dependence patterns are constructed by

$$b_t = \begin{cases} b, & 2 * sin(\frac{\pi}{5} * x_{t-1}^1) \leq x_{t-1}^2, \\ -b, & 2 * sin(\frac{\pi}{5} * x_{t-1}^1) > x_{t-1}^2. \end{cases} \tag{6}$$

$$b_t = \begin{cases} b, & \sqrt{(x_{t-1}^1)^2 + (x_{t-1}^2)^2} \leq 3.99, \\ -b, & \sqrt{(x_{t-1}^1)^2 + (x_{t-1}^2)^2} > 3.99. \end{cases} \tag{7}$$

The two nonlinear dependence patterns are illustrated in Fig. 3. The x-axis represents the first relevant variable of solutions and y-axis represents the second variable. As illustrated in Fig. 3a, the dependence pattern of (6) forms a sinusoidal boundary for classification. In contrast, (7) in Fig. 3b forms a circular boundary for classification.



(a) Sinusoidal pattern (b) Circular pattern

Fig. 3: Nonlinear dependence patterns, where the green area is the range of variables that have a positive influence in the future and the rest area is that of negative influence.

The parameters of the problems is set following [20]. The change frequency of the problems is set as 10,000 function evaluations.

2) Competing Algorithms: Two algorithms are taken for comparisons, i.e., the very recent SQL-PSO [15] and PSO [16]. Particularly, SQL-PSO performs the best among existing typical algorithms for DTPs. The parameters of the compared algorithms are set the same as in [15]. In all algorithms, the swarm size is set as 100, $\omega = 0.75$, and $c_1 = c_2 = 1.4$.

TABLE I: Results of dissimilarity index on multiple instances with different b and t^{end} values

b	t^{end}=10	t^{end}=40	t^{end}=70	t^{end}=100
0	0.18 ± 0.05	0.24 ± 0.04	0.26 ± 0.03	0.27 ± 0.03
10	0.32 ± 0.1	0.35 ± 0.07	0.37 ± 0.06	0.36 ± 0.05
20	0.62 ± 0.15	0.57 ± 0.09	0.57 ± 0.08	0.55 ± 0.06
30	0.74 ± 0.17	0.83 ± 0.09	0.8 ± 0.07	0.8 ± 0.06
40	0.96 ± 0.2	1.04 ± 0.12	1.06 ± 0.13	1.06 ± 0.1
50	1.12 ± 0.22	1.19 ± 0.09	1.22 ± 0.07	1.22 ± 0.06
60	1.24 ± 0.15	1.32 ± 0.09	1.34 ± 0.06	1.35 ± 0.05
70	1.29 ± 0.13	1.38 ± 0.07	1.4 ± 0.06	1.4 ± 0.04
80	1.34 ± 0.14	1.4 ± 0.04	1.4 ± 0.03	1.41 ± 0.03
90	1.35 ± 0.12	1.44 ± 0.05	1.44 ± 0.03	1.46 ± 0.03
100	1.36 ± 0.14	1.44 ± 0.06	1.46 ± 0.04	1.47 ± 0.04

3) Performance Metric: To evaluate the performance of the algorithm, the accumulated fitness value of decision solutions in all environments is used as

$$F = \sum_{t=1}^{t^{end}} f(\mathbf{x}_t^*, \gamma_t) \tag{8}$$

where $\mathbf{x}_t^*$ is the decision solution in t-th environment.

Each algorithm independently runs 20 times on each problem instance. The mean values and the standard deviations over 20 runs are reported. Wilcoxon rank-sum test is performed between the proposed method and each competing algorithm at a significance level of 0.05. The signs "+", "≈" and "-" represent that the proposed method is significantly better than, equal to or worse than the competing algorithm. The algorithm with the maximum result is highlighted in **bold**.

B. Effect of Components

This subsection observes the relationship between the dissimilarity index and the time linkage. In addition, the effect of each component in the algorithm is tested.

1) Dissimilarity Index for Time-Linkage Detection: In order to investigate the effect of time-linkage detection, this subsection observes the relationship between the dissimilarity index (denoted as I_d) and time linkage in problems. The problem with a linear dependence pattern in (5) is adopted to construct multiple instances with a different b and t^{end}. Particularly, b is set in a range of [0, 100] with a step size of 10, i.e., $b = 0, 10, 20, ..., 100$. The number of environments t^{end} is set as 10, 40, 70, and 100.

The average and standard deviation of the dissimilarity index I_d over 20 runs are reported in Table I. It can be observed that I_d increases with the growth of the time-linkage coefficient b. When $b > 60$, I_d increases slowly and tends to converge. In the case of $t^{end} = 10$, I_d is lower than that of $t^{end} = 100$ due to the lack of historical data. Nevertheless, the correlation between I_d and b is strong as t^{end} increases. In contrast, when $b \leq 30$, the time-linkage property does not play a significant role in the environments. Therefore, in the subsequent experiments, the threshold $\delta = 0.9$ is chosen for time-linkage detection.

TABLE II: Statistical results of CSVC-PSO and its variants, i.e., CSVC, CSVC+Detection, and CSVC+Cross

b	CSVC	CSVC+Detection	CSVC+Cross	CSVC-PSO
10	4050 ± 760	4967 ± 352	4384 ± 563	**4967 ± 352**
50	6357 ± 922	6382 ± 834	6693 ± 837	**6718 ± 834**
100	8889 ± 937	8781 ± 857	9231 ± 845	**9260 ± 856**

TABLE III: Statistical results of CSVC-PSO, CSVC-PSO(linear), PSO, and SQL-PSO

Pattern	b	CSVC-PSO	CSVC-PSO(linear)	PSO	SQL-PSO
Linear	10	**4967 ± 352**	**4967 ± 352**(≈)	**4967 ± 352**(≈)	3879 ± 644(+)
	30	**5274 ± 662**	**5274 ± 663**(≈)	5108 ± 684(+)	4599 ± 930(+)
	50	6540 ± 966	**6718 ± 834**(−)	4963 ± 968(+)	5348 ± 1199(+)
	80	8094 ± 687	**8200 ± 715**(≈)	5021 ± 1334(+)	7617 ± 1259(≈)
	100	8917 ± 1075	**9260 ± 856**(−)	4903 ± 1680(+)	9117 ± 1625(−)
Sinusoidal	10	**4955 ± 367**	**4955 ± 367**(≈)	**4955 ± 367**(≈)	4087 ± 626(+)
	30	5081 ± 575	**5088 ± 586**(≈)	4994 ± 694(≈)	4178 ± 793(+)
	50	5710 ± 854	**5821 ± 897**(−)	4858 ± 943(+)	5367 ± 910(+)
	80	7295 ± 1033	**7327 ± 1209**(≈)	5485 ± 1364(+)	6557 ± 1473(+)
	100	7262 ± 1377	7652 ± 1478(+)	4533 ± 1772(+)	**8833 ± 1107**(−)
Circular	10	**4978 ± 352**	**4978 ± 352**(≈)	**4978 ± 352**(≈)	4344 ± 591(+)
	30	5004 ± 525	5008 ± 518(≈)	**5081 ± 444**(≈)	4458 ± 442(+)
	50	4999 ± 681	4788 ± 610(+)	5043 ± 534(≈)	**5076 ± 908**(≈)
	80	**5380 ± 1041**	4938 ± 1279(+)	5365 ± 928(≈)	5346 ± 1712(≈)
	100	5368 ± 1849	5369 ± 1854(≈)	5443 ± 1561(≈)	**6410 ± 1879**(−)
	$+/\approx/-$		3/9/3	7/8/0	9/3/3

2) Effects of Components: CSVC-PSO consists of three components: time-linkage detection, classifier, and crossover. Four CSVC-PSO variants are tested, i.e., CSVC that selects the best one with positive influence from the solutions found by PSO, CSVC+Detection that uses the time-linkage detection and CSVC for solution evaluation, and CSVC+Cross that uses CSVC for solution evaluation and a crossover operator to generate the final decision solution. Note that all the three variants use the same optimizer, i.e., PSO, as CSVC-PSO. There instances with a linear dependence pattern $b = 10, 50, 100$ are taken as examples for test. The statistical results of CSVC-PSO and the three variants are reported in Table II. As shown in Table II, CSVC-PSO obtains the best results among four algorithms on all instances. Particularly, the time-linkage detection can improve algorithm performance on instances with $b = 10, 50$, showing that it is better to select solutions based on current fitness value when the time linkage is weak. In contrast, the crossover operator can improve algorithm performance on instances with $b = 100$, showing that the it is better to select irrelevant variables based on current fitness value only. In general, the three components are important in the proposed method.

C. Compared with other Algorithms

In this section, the proposed method is compared with other algorithms. CSVC-PSO configured with a linear kernel is also compared, named CSVC-PSO(linear). Multiple DTP instances with different dependence patterns, i.e., linear, sinusoidal, and circular patterns, are constructed. Particularly, b is set as 10, 30, 50, 80, and 100. A total of 15 instances are tested.

The statistical results of CSVC-PSO, CSVC-PSO(linear), PSO and SQL-PSO are reported in Table III. There is no significance difference between CSVC-PSO and CSVC-PSO(linear) on 9 out of 15 instances. CSVC-PSO performs

TABLE IV: Average running time of each algorithm in an environment

Algorithm	CSVC-PSO	CSVC-PSO (linear)	PSO	SQL-PSO
Time(s)	0.2868	0.2826	**0.0732**	16.9068
Ratio	1	0.98	**0.25**	58.95

"Ratio" is the ratio of the time of an algorithm to that of CSVC-PSO

better than CSVC-PSO(linear) on 3 instances on instances with sinusoidal and circular dependence patterns, whereas worse on 2 instances with linear patterns. This shows that a radius bias function kernel work better on nonlinear dependence patterns. From Table III, CSVC-PSO performs significantly better than PSO and SQL-PSO on 7 and 9 out of 15 instances, whereas worse on 0 and 3 instances only. Particularly, on all instances with $b \leq 30$, CSVC-PSO presents better or similar performance to PSO since the best solution found by PSO is taken as decision in cases of weak time linkage. In contrast, on most instances with $b > 30$, CSVC-PSO performs significantly better than PSO. This shows that the proposed method can well focus on the current fitness value and the future influences in different cases. Compared with SQL-PSO, CSVC-PSO performs significantly better on all instances with $b \leq 30$ and most instances with $b > 30$, especially on sinusoidal dependence patterns. This shows that the proposed method can well evaluate the future influence of solutions in dynamic environments. Specifically, CSVC-PSO performs worse than SQL-PSO on instances with $b = 100$. This is because that the state predictor in SQL-PSO can well predict rewards when the dependence density is very strong. In general, the proposed method can evaluate the positive or negative influence on different types of dependence patterns and density using a simple classification rather than using reinforcing learning.

Table IV lists the average running time of all algorithms on one instance with a linear dependence pattern and $b = 100$. As shown in Table IV, CSVC-PSO and CSVC-PSO (linear) have similar running time. PSO is the quickest. CSVC-PSO require less than 0.3 seconds due to their simple and efficient design. In contrast, SQL-PSO requires the longest computation time due to its extensive use of proxy models, which is nearly 59 times that of CSVC-PSO. Thus, CSVC-PSO can obtain better results than SQL-PSO using a shorter time.

V. Conclusion

This paper proposes particle swarm optimization with a clustering-based support vector classifier, named CSVC-PSO, to solve DTPs. In the proposed method, a time-linkage detection method is developed to evaluate the density of the dependence of decisions between environments. A clustering-based support vector classifier is trained using historical data to evaluate the influence of solutions in the future. A crossover operator is performed to combine the current fitness values and future influence for decision making. Experimental results show that, on multiple instances with different types of dependence patterns and density, the proposed CSVC-PSO outperforms than state-of-the-art algorithms on most instances. In addition, CSVC-PSO requires a shorter time than the state-of-the-art algorithm.

References

[1] T. T. Nguyen, Z. Yang, and S. Bonsall, "Dynamic time-linkage problems-the challenges," in *2012 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future*. IEEE, 2012, pp. 1–6.

[2] B. Werth, E. Pitzer, J. Karder, S. Wagner, and M. Affenzeller, "Dynamic vehicle routing with time-linkage: from problem states to algorithm performance," in *International Conference on Computer Aided Systems Theory*. Springer, 2022, pp. 69–77.

[3] T. T. Nguyen and X. Yao, "Dynamic time-linkage evolutionary optimization: definitions and potential solutions." *Metaheuristics for Dynamic Optimization*, vol. 433, pp. 371–395, 2013.

[4] Z.-H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artificial Intelligence Review*, pp. 1–52, 2022.

[5] J. Luo, F. He, H. Li, X.-T. Zeng, and Y. Liang, "A novel whale optimisation algorithm with filtering disturbance and nonlinear step," *International Journal of Bio-Inspired Computation*, vol. 20, no. 2, pp. 71–81, 2022.

[6] A. Ahrari, S. Elsayed, R. Sarker, D. Essam, and C. A. C. Coello, "Adaptive multilevel prediction method for dynamic multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 463–477, 2021.

[7] J. Karimi, H. Nobahari, and S. H. Pourtakdoust, "A new hybrid approach for dynamic continuous optimization problems," *Applied Soft Computing*, vol. 12, no. 3, pp. 1158–1167, 2012.

[8] M. Jiang, Z. Wang, S. Guo, X. Gao, and K. C. Tan, "Individual-based transfer learning for dynamic multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 10, pp. 4968–4981, 2021.

[9] X.-F. Liu, Z.-H. Zhan, T.-L. Gu, S. Kwong, Z. Lu, H. B.-L. Duh, and J. Zhang, "Neural network-based information transfer for dynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1557–1570, 2019.

[10] W. Zheng, H. Chen, and X. Yao, "Analysis of evolutionary algorithms on fitness function with time-linkage property," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 696–709, 2021.

[11] W. Zheng, Q. Zhang, H. Chen, and X. Yao, "When non-elitism meets time-linkage problems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 741–749.

[12] W. Zheng and X. Yao, "Theoretical analyses of evolutionary algorithms on time-linkage onemax with general weights," *arXiv preprint arXiv:2305.07098*, 2023.

[13] P. A. Bosman, "Learning, anticipation and time-deception in evolutionary online dynamic optimization," in *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*, 2005, pp. 39–47.

[14] C. Bu, W. Luo, T. Zhu, and L. Yue, "Solving online dynamic time-linkage problems under unreliable prediction," *Applied Soft Computing*, vol. 56, pp. 702–716, 2017.

[15] T. Zhang, H. Wang, B. Yuan, Y. Jin, and X. Yao, "Surrogate-assisted evolutionary q-learning for black-box dynamic time-linkage optimization problems," *IEEE Transactions on Evolutionary Computation*, 2022.

[16] X.-F. Liu, Y. Fang, Z.-H. Zhan, and J. Zhang, "Strength learning particle swarm optimization for multiobjective multirobot task scheduling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 7, pp. 4052–4063, 2023.

[17] X.-F. Liu, X.-X. Xu, Z.-H. Zhan, Y. Fang, and J. Zhang, "Interaction-based prediction for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023.

[18] S. Tutorials, "Pearson correlation," *Retrieved on February*, vol. 4, 2014.

[19] S. Ghosh, A. Dasgupta, and A. Swetapadma, "A study on support vector machine based linear and non-linear pattern classification," in *2019 International Conference on Intelligent Sustainable Systems*, 2019, pp. 24–28.

[20] H. Fu, P. R. Lewis, B. Sendhoff, K. Tang, and X. Yao, "What are dynamic optimization problems?" in *2014 IEEE Congress on Evolutionary Computation*. IEEE, 2014, pp. 1550–1557.