Speciated Evolutionary Algorithm for Dynamic Constrained Optimisation

Xiaofen $Lu^{1,2(\boxtimes)}$, Ke Tang², and Xin Yao^{1,2}

¹ CERCIA, School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK {xx1332,x.yao}@cs.bham.ac.uk
² UBRI, School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei 230027, Anhui, China ketang@ustc.edu.cn

Abstract. Dynamic constrained optimisation problems (DCOPs) have specific characteristics that do not exist in dynamic optimisation problems with bounded constraints or without constraints. This poses difficulties for some existing dynamic optimisation strategies. The maintaining/introducing diversity approaches might become less effective due to the presence of infeasible areas, and thus might not well handle with the switch of global optima between disconnected feasible regions. In this paper, a speciation-based approach was firstly proposed to overcome this, which utilizes deterministic crowding to maintain diversity, assortative mating and local search to promote exploitation, as well as feasibility rules to deal with constraints. The experimental studies demonstrate that the newly proposed method generally outperforms the state-of-theart algorithms on a benchmark set of DCOPs.

Keywords: Evolutionary algorithm \cdot Speciation \cdot Deterministic crowding \cdot Local search \cdot Dynamic constrained optimisation problem

1 Introduction

In the real world, many optimisation problems are changing over time due to the dynamic environments [6, 19]. These problems require an optimisation algorithm to quickly find the new optimum once the problem changes [20]. As a class of nature-inspired optimisation methods, evolutionary algorithms (EAs) can have good adaptation to the changing environments, and thus have been widely studied in the field of dynamic optimisation (DO). Many evolutionary DO approaches have been developed, which include maintaining/introducing diversity strategies, memory approaches, prediction approaches, multi-population approaches and so on [20]. As revealed in [20,23], most existing studies of DO focus on unconstrained or bounded constrained dynamic optimisation problems (DCOPs), and few consider dynamic constrained optimisation problems (DCOPs) despite their high popularity in real-world applications. In a DCOP, a change may occur

© Springer International Publishing AG 2016

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 203–213, 2016. DOI: 10.1007/978-3-319-45823-6_19

in either constraints or objective functions or both. Therefore, DCOPs have some specific characteristics compared to unconstrained or bounded constrained DOPs. For a DCOP, the distribution of infeasible/feasible solution might change, and the global optima might move to another disconnected feasible region, or appear in a new region without changing the current optima due to the dynamics of environments [19,23].

Addressing DCOPs poses difficulties for some existing DO strategies and constraint handling (CH) techniques due to their characteristics. As discussed in [19,23], maintaining/introducing diversity methods such as random-immigrants (RI) [12] and hyper-mutation (HyperM) [5] become less effective on DCOPs than on DOPs, when combined with penalty function that prefers feasible solutions to infeasible ones. This is because that they cannot maintain enough diversity to adapt to the new problem as the introduced random solutions are likely to be rejected by the penalty function if they are infeasible. Furthermore, without enough diversity, they may not deal well with the switch of the global optimum between disconnected feasible regions, as it needs to go through an infeasible path from the previous optimum to the current optimum. Moreover, some adaptive/self-adaptive CH techniques also face challenges in solving DCOPs as they need the knowledge of problem that is unavailable in a dynamic environment or historical information that might be outdated once the problem changes [23].

Researchers have recently carried out some studies trying to solve the challenges of DCOPs. To allow diversified infeasible solutions distributed in the whole search space, the authors in [19,25], and [2] applied the repair method [27] to handle constraints along with RI/HyperM to deal with dynamics. However, these methods need a lot of feasibility checkings, and thus cannot be applied to DCOPs in which the ratio of feasible solutions is very low and a feasibility checking is computationally costly. The authors in [1] employed simple feasibility rules [17] as the CH strategy along with RI and combined DE variants to introduce diversity after each change. However, this method may not maintain diversity well during the run as infeasible solutions are still likely abandoned by feasibility rules. Furthermore, it might be ineffective to make the partially converged population to re-diversify to track the switched global optima. Similarly, the proposed approach in [4] to deal with DCOPs might not quickly find the switched optima as the population tends to converge during the run.

In this paper, a speciation-based method, called speciated evolution with local search (SELS), is suggested to address the challenges of DCOPs. Speciation allows an EA to find multiple optima through making comparisons among similar individuals [7]. Thus, newly generated and promising infeasible solutions can be accepted in the new method. Furthermore, good solutions can be maintained in different feasible regions, and thus SELS should react quickly when the global optimal solution switches to another feasible region. In the literature, speciation has been utilized to solve dynamic unconstrained or bounded constrained optimisation problems [14, 15], but no studies apply them to DCOPs. In addition to speciation, a local search strategy is employed in SELS to promote exploitation of the promising regions to quickly find the changed optimum. SELS also uses a change detection method, and adds some random immigrants into the population to introduce diversity once a change is detected. Finally, to deal with constraints, the simple and parameter-free feasibility rules [17] are employed.

The remaining part of this paper is organized as follows. Section 2 summarizes the existing related work, and Sect. 3 details the new method. In Sect. 4, experimental results are presented, and conclusions and future work will be given in Sect. 5.

2 Related Work

Solving DCOPs does not only need DO strategies to deal with dynamics but also requires CH techniques to handle constraints. This section will introduce the efforts that researchers have done in combining DO strategies and CH techniques to solve DCOPs.

The RI and HyperM methods were combined in [23] with a penalty function proposed in [18]. When using RI, a fraction of the population is replaced by random solutions at every generation to maintain diversity. In hybrid with HyperM, the original mutation rate will change to a higher one to introduce diversity once change is detected. However, as most of the added random solutions are infeasible and they are likely rejected by the used penalty function, these combination methods can not maintain enough diversity to adapt to the new problem. Therefore, the authors suggested that a CH technique that can maintain diversified infeasible solutions is needed, when combined with RI and HyperM to solve DCOPs.

To maintain diversified infeasible solutions in solving DCOPs, the authors in [22] combined the repair method with RI and HyperM, respectively. By using the repair method, an infeasible solution was evaluated by the repaired feasible solution. Thus, the infeasible solutions that can make good feasible solutions are reserved. Other studies using the repair methods exist in [2,24]. The former work used an improved repair method, and the latter applied the repair method and RI together in a DE context. However, using the repair method has a big disadvantage. That is, it requires a considerable number of feasibility checking during the repair process, and thus not suitable for problems with very small feasible area or expensive feasibility checking.

A simple ranking scheme in [13] was applied to handling constraints in [3]. To maintain population diversity, the method monitored the population diversity and switched between a global search and a local search operator according to whether the diversity degree is larger than a threshold. As a result, this method will highly depends on the setting of the threshold. To avoid the setting of the threshold, the authors in [4] used the Shannon's index of diversity as a factor to balance the influence of the global-best and local-best search directions. However, the population tends to converge in this method, and it might be ineffective to make the partially converged population to re-diversify to track the switched global optimum.

The authors in [1] employed simple feasibility rules in [17] as the CH strategy, and considered RI and combined DE variants to maintain and introduce diversity, respectively. However, this method may not well maintain diversity as infeasible solutions are likely abandoned according to feasibility rules, and thus might not effectively solve the switching global optima between disconnected feasible regions.

Except DO strategies to introduce/maintain diversity were considered, other DO strategies such as memory and prediction methods were also combined with CH techniques to deal with DCOPs. The study in [26] adapted abstract memory method, and the infeasibility driven evolutionary algorithm (IDEA) was combined with prediction method (to predict the future optima) in [10] and [11] to solve DCOPs. However, both memory and prediction methods are only applicable to particular dynamic problems (i.e., cyclic and predictable dynamic problems, respectively).

3 The Proposed Method

The proposed SELS method considers simple feasibility rules to handle constraints, which do not need to repair infeasible solutions. However, as feasibility rules prefer feasible solutions to infeasible ones, the diversity will decrease quickly. To avoid this, the speciation is employed, which makes comparison among similar individuals. Thus, SELS should maintain good diversity and respond quickly once the change happens. As speciation focuses on exploration, the new optima might not be found quickly as promising regions are not exploited sufficiently. Therefore, SELS uses a local search strategy to promote exploitation of the promising regions. In the following part of this section, the elements of SELS will be first described, and then the pseudo-code is given.

Speciation Method. This work uses deterministic crowding (DC) as the speciation method. Algorithm 1 gives the pseudo-code of DC. The DC method pairs all population elements randomly and generates two offspring for each pair based on EA operators. Selection is then operated on these four individuals, and a similarity measure is used to decide which offspring competes against which parent. The offspring will replace the compared parent and enter next generation if it is fitter.

In addition to DC, SELS employs an assortative mating (AM) [8] to induce speciation in the population. As DC can maintain good solutions on different peaks or in different feasible regions, intuitively, we would not like to operate crossover between solutions on different peaks or in different feasible regions as doing this will likely generate solutions in the valley or infeasible regions. To avoid this, assortative mating is used, which mates individuals with the most similar non-identical partner in the population. Through doing crossover between individuals in proximity, species will be automatically generated, and the exploitation of the corresponding search area will also be enhanced. In this work, Euclidean distance is used as the similarity measure.

Algorithm 1. Deterministic Crowding [16]

1: Randomly pair all individuals in the population 2: for each pair of individuals, p_1 and p_2 , do 3: Generate two offspring, o_1 and o_2 , based on EA operators 4: if $dist(p_1, o_1) + dist(p_2, o_2) \le dist(p_1, o_2) + dist(p_2, o_1)$ then 5: $p_1 = \operatorname{fitter}(p_1, o_1)$ 6: $p_2 = \operatorname{fitter}(p_2, o_2)$ 7: else 8: $p_1 = \operatorname{fitter}(p_1, o_2)$ 9: $p_2 = \operatorname{fitter}(p_2, o_1)$ 10: end if 11: end for

Feasibility Rules. Feasibility rules have been commonly used to solve constrained optimisation problems due to its simple and parameter-free characteristics, which make comparison between individuals through the following three selection criteria:

- 1. If both are feasible solutions, the one with the highest fitness value is selected.
- 2. Between a feasible solution and an infeasible solution, the feasible one is preferred.
- 3. If both are infeasible, the one with the lowest sum of constraint violation wins.

In the proposed SELS, feasibility rules are employed to determine the fitter one in each pair of parent and offspring in the deterministic crowding.

Local Search (LS). In this work, the local evolutionary search enhancement by random memorizing (LESRM) [28] is applied to the best solution (x_{best}) at each generation. The LESRM uses an EA that has a step size control and adjusts the search direction based on individuals encountered before, and thus can do efficient exploitation in the area that the best solution is located in. In this paper, the EA used in LESRM is given in Algorithm 2, and the random memorizing part of LESRM is the same as in [28]. Here, ls_{num} denotes the maximum number function evaluations (FEs) permitted for LS at every generation, and D denotes dimension of the problem. The success ratio of δ denotes the ratio that x_{new} (generated by δ) is better than x_{best} .

Change Detection and Diversity Introduction. To detect the change, assume k is the number of individuals for detection and NP is the size of population, the (NP/k)-th, (2*NP/k)-th, ((k-1)*NP/k)-th, ..., (NP)-th (denoted as detection index) individual in the population are reevaluated at every generation to detect changes in time. Once a change is detected, the whole population will be re-evaluated, and NI random individuals will be generated to randomly replace the individuals of the population except the best individual. In our work, we

Algorithm 2. Evolutionary Search in LESRM in SELS (x_{best})

1: Find the closest non-identical solution x_{near} to x_{best} 2: Set $\delta = \text{dist}(x_{near}, x_{best})$, and LS generation counter $ls_g = 0$ 3: repeat 4: repeat 5:Set $ls_g = ls_g + 1$, and $x_{new} = x_{best} + \delta^* \operatorname{randn}(1, D)$; if $mod(ls_q,2) == 0$ then 6: 7: Calculate the success ratio of δ Set $\delta = \delta/2$ if the success ratio of δ is less than 0.5 8: Set $\delta = \delta * 2$ if the success ratio of δ is larger than 0.5 9: 10:end if **until** x_{new} is fitter than x_{best} 11: 12:Set $x_{best} = x_{new}$ 13:Do random memorizing as in [28]14: **until** ls_{num} function evaluations are used up

first give each individual in the population a rank based on the feasibility rules, and estimate the degree of change severity as the ratio of the number of reverse order after re-evaluation. We then set NI to max(([reverse_ratio * NP], 2).

The Framework of the Proposed Method. Algorithm 3 gives the whole process of SELS, which begins with a randomly generated population of candidate solutions and utilizes genetic algorithm (GA) to evolve this population.

Al	gorithm 3. The Framework of SELS
1:	Evaluate a randomly generated population $\mathbf{P} = \{x_i i = 1, 2,, NP\}$
2:	while computational resources are not used up do
3:	repeat
4:	Randomly select an unpaired individual x from the population
5:	Calculate Euclidean distance between x and each other unpaired individual
6:	Pair x with the individual that has smallest positive Euclidean distance to x
7:	until all individuals in the population are paired
8:	for $i \leftarrow 1, NP$ do
9:	Re-evaluate x_i or x_{i+1} if i or $i+1$ is one detection index
10:	if the change is detected then
11:	Re-evaluate solutions in ${f P}$ and introduce diversity, go to Step 17
12:	else
13:	Generate two offspring o_1, o_2 from x_i and x_{i+1} with GA crossover and
	mutation
14:	Do deterministic crowding (x_i, x_{i+1}, o_1, o_2) , and set $i = i + 1$
15:	end if
16:	end for
17:	Do local search to the best solution in \mathbf{P}
18:	end while

4 Experimental Studies

4.1 Experimental Setup

To assess the efficacy of SELS, we conducted experiments on 11 DCOP benchmark test functions proposed in [21]. They are: G24-l(dF,fC), G24-2(dF,fC), G24-3(dF,dC), G24-4(dF,dC), G24-5(dF,dC), G24-6a(2DR,hard), G24-6c(2DR,easy), G24-7(fF,dC), G24-6d(2DR,hard), G24-8b(fC,OICB). We recorded the performance of SELS on each test function using the modified offline error [22] as the performance metric, then compared its performance to that of 6 state-of-the-art algorithms. They are, dGArepairRIGA [22], dGArepairHyperM [22], GSA + Repair [24], DDECV + Repair [2], DDECV [1] and EBBPSO-T [4]. In the experiments, the number of changes is set to 12, the change frequency is 1000 objective function evaluations, and the change severity is medium (i.e., k = 0.5, and s = 20). We use this setting as all of the 6 compared algorithms have presented complete experimental results only on this setting in their original papers.

Note that the first 4 of the compared algorithms use a repair scheme, so they need a lot of feasibility checking but they ignore the cost. To make a fair comparison, we run SELS only evaluating the feasibility for an infeasible solution and do not count in the number of used fitness evaluations. The resulted algorithm is denoted as eSELS and compared to the 4 repair algorithms. When compared to DDECV and EBBPSO, SELS evaluates both the feasibility and objective function value for every individual, no matter feasible or infeasible, which is the same to what DDECV and EBBPSO do.

In the experiments, for both SELS and eSELS, intermediate crossover with $p_c = 1.0$, and Guassian mutation with $p_m = 1/D$ and scale = 0.1 are used, respectively. In the mutation, at least one variable is mutated every time. The number of LS objective function evaluations (ls_{num}) is set to 16, and 4 individuals are used for change detection every generation. Each algorithm is run 50 times on each test function.

4.2 Comparison Results with Existing Algorithms

Table 1 summarizes the mean and standard deviation of the modified offline error over 50 runs obtained by DDECV, EBBPSO-T and SELS as well as the performance rank of each algorithm on each test function (in case of ties, average ranks are assigned) based on Z-test with a level of 0.05. We applied the Friedman test and further a Holm's post-hoc procedure [9], which was used for multiple comparison of algorithms, to investigate whether SELS performed best on the set of test functions. The analysis shows that SELS has significant improvement than DDECV and EBBPSO-T at a level of 0.05.

Table 2 gives the performance rank of eSELS and the other 4 repair algorithms on each test function. We also applied the Friedman test and further a Holm's post-hoc procedure [9] to do a multiple-problem comparison among the 5 algorithms. The statistical test results show that eSELS performed significantly better than each other algorithm on the test function set at a level of 0.05.

Func	DDECV[rank]	EBBPSO-T[rank]	SELS[rank]
G24-l	$0.109 \pm 0.033 [3]$	$0.084 \pm 0.041 [2]$	$\bf 0.025 \pm 0.008 [1]$
G24-2	$0.126 \pm 0.030 [2]$	$0.136 \pm 0.013 [3]$	${\bf 0.050 \pm 0.015} [1]$
G24-3	$0.057 \pm 0.018 [3]$	${\bf 0.032 \pm 0.005} [1]$	$0.044 \pm 0.022 [2]$
G24-3b	$0.134 \pm 0.033 [3]$	$0.104 \pm 0.015 [2]$	${\bf 0.052 \pm 0.018} [1]$
G24-4	$0.131 \pm 0.032 [2.5]$	$0.138 \pm 0.022 [2.5]$	${\bf 0.082 \pm 0.021} [1]$
G24-5	$0.126 \pm 0.030 [2.5]$	$0.126 \pm 0.019 [2.5]$	${\bf 0.054 \pm 0.014} [1]$
G24-6a	$0.215 \pm 0.067 [3]$	$0.116 \pm 0.099 [2]$	${\bf 0.055 \pm 0.009} [1]$
G24-6c	$0.128 \pm 0.025 [2]$	$0.251 \pm 0.061 [3]$	${\bf 0.052 \pm 0.008} [1]$
G24-6d	$0.288 \pm 0.055 [2.5]$	$0.312 \pm 0.203 [2.5]$	$\bf 0.041 \pm 0.007 [1]$
G24-7	$0.106 \pm 0.022 [3]$	${\bf 0.045 \pm 0.009} [1]$	$0.087 \pm 0.016 [2]$
G24-8b	$0.151 \pm 0.058 [2]$	$0.312 \pm 0.086 [3]$	${\bf 0.055 \pm 0.022} [1]$

Table 1. Comparison results between DDECV, EBBPSO-T and SELS based on experimental results of DDECV and EBBPSO-T in their original papers [1,4], respectively. The best result obtained on each function is marked in **bold**.

 Table 2. Comparison results among eSELS and the other 4 repair algorithms based on the experimental results in their original papers.

Func	dRepairRIGA [rank]	dRepairHyperM [rank]	GSA + Repair [rank]	DDECV + Repair [rank]	eSELS[rank]
G24-l	$0.082 \pm 0.015 [3]$	$0.093 \pm 0.023 [4]$	$0.132 \pm 0.015 [5]$	$0.061 \pm 0.010[2]$	0.013 ± 0.007 [1]
G24-2	$0.162 \pm 0.021 [3.5]$	$0.171 \pm 0.026 [3.5]$	$0.182 \pm 0.019 [5]$	$0.062 \pm 0.006 [2]$	0.030 ± 0.008 [1]
G24-3	$0.029 \pm 0.004 [4]$	$0.027 \pm 0.005 [2.5]$	$0.028 \pm 0.004 [2.5]$	$0.046 \pm 0.006 [5]$	0.018 ± 0.004 [1]
G24-3b	$0.058 \pm 0.007 [2]$	$0.071 \pm 0.014 [3]$	$0.076 \pm 0.009 [4]$	$0.084 \pm 0.006 [5]$	$\bf 0.021 \pm 0.004 [1]$
G24-4	$0.140 \pm 0.028 [5]$	$0.059 \pm 0.010 [2]$	$0.073 \pm 0.012 [3]$	$0.088 \pm 0.011 [4]$	$0.036 \pm 0.009 [1]$
G24-5	$0.152 \pm 0.017 [4.5]$	$0.131 \pm 0.019 [3]$	$0.153 \pm 0.013 [4.5]$	$0.078 \pm 0.008 [2]$	$0.027 \pm 0.024 [1]$
G24-6a	$0.366 \pm 0.033 [4.5]$	$0.358 \pm 0.049 [4.5]$	${\bf 0.033 \pm 0.003} [1]$	$0.036 \pm 0.005 [2.5]$	$0.038 \pm 0.006 [2.5]$
G24-6c	$0.323 \pm 0.037 [4.5]$	$0.326 \pm 0.047 [4.5]$	$0.045 \pm 0.004 [3]$	$0.041 \pm 0.010 [1.5]$	$0.040 \pm 0.007 [1.5]$
G24-6d	$0.315 \pm 0.029 [5]$	$0.286 \pm 0.035[4]$	$0.037 \pm 0.007 [2]$	$0.079 \pm 0.006[3]$	$0.029 \pm 0.004 [1]$
G24-7	$0.154 \pm 0.031 [5]$	$0.067 \pm 0.014 [3]$	${\bf 0.018 \pm 0.002} [1]$	$0.107 \pm 0.011[4]$	$0.035 \pm 0.045[2]$
G24-8b	$0.341 \pm 0.053 [5]$	$0.257 \pm 0.042 [4]$	$0.192 \pm 0.034 [3]$	$0.074 \pm 0.025[2]$	0.025 ± 0.006 [1]

4.3 The Performance Effect of AM, LS and Different Dynamics

We further conducted experiments to check whether the AM and LS can help in SELS, and comparisons were made among (1) SELS without AM or LS (SELS-am-ls in short), (2) SELS without LS (SELS-ls in brief), and (3) SELS. Table 3 summarizes the mean and standard deviation of the modified offline error for each of them along with the comparison results. It is shown that SELS-ls over-all outperformed SELS-am-ls, and the used LS further improves SELS-ls. This demonstrates the benefits of using AM and LS.

To evaluate the performance of SELS on different dynamics, we also conducted experiments on small change severity (i.e., k = 1.0, and s = 10) and large severity (i.e., k = 0.25, and s = 50). Figure 1 gives the evolutionary curves

Table 3. Comparison results among SELS-am-ls, SELS-ls, and SELS based on experimental results implemented on DCOP test functions. Here, +, -, and \approx denotes whether one algorithm is better, worse or equal to another according to Wilcoxon ranksum test with a level of 0.05.

Func	SELS-am-ls	SELS-ls vs SELS-am-ls	SELS vs SELS-ls
G24-1	0.133 ± 0.037	0.069 ± 0.019 +	0.025 ± 0.008 +
G24-2	0.186 ± 0.017	0.121 ± 0.021 +	$0.050 \pm 0.015 ~+$
G24-3	0.124 ± 0.038	$0.118\pm0.025\approx$	0.044 ± 0.022 +
G24-3b	0.233 ± 0.039	0.144 ± 0.021 +	0.052 ± 0.018 +
G24-4	0.199 ± 0.027	0.171 ± 0.030 +	0.082 ± 0.021 +
G24-5	0.162 ± 0.022	0.117 ± 0.017 +	0.054 ± 0.014 +
G24-6a	0.318 ± 0.040	0.163 ± 0.020 +	0.055 ± 0.009 +
G24-6c	0.284 ± 0.030	0.152 ± 0.017 +	0.052 ± 0.008 +
G24-6d	0.198 ± 0.033	0.128 ± 0.023 +	0.041 ± 0.007 +
G24-7	0.121 ± 0.017	0.138 ± 0.018 -	0.087 ± 0.016 +
G24-8b	0.387 ± 0.044	$0.242 \pm 0.031 +$	$0.055 \pm 0.022 \pm$



Fig. 1. The Evolutionary difference curves of SELS between different change severity

of the normalised offline error differences between medium and small severity, and between large and small severity. The X axis denotes the number of objective function evaluations, and the Y axis denotes difference of the normalised offline error at each evaluation, which is normalised on each problem in one test function. In general, we found that SELS performed best on small severity, second best on medium, and worst on large severity.

5 Conclusion and Future Work

In this paper, a novel speciation-based method was proposed to solve DCOPs, which combines speciation methods as well as local search together.

212 X. Lu et al.

Although the techniques used in SELS are not new, the experimental studies demonstrated the combination leads to an effective algorithm. In future work, we will study the performance effect of the choice of local search strategies on the proposed method, and will also evaluate this new method on more test functions.

Acknowledgments. This work was partially supported by NSFC (Grant No. 61329302), EPSRC (Grant No. EP/K001523/1), and Royal Society Newton Advanced Fellowship (Ref. no. NA150123). The authors thank Stefan Menzel for giving the valuable advice.

References

- Ameca-Alducin, M.Y., Mezura-Montes, E., Cruz-Ramirez, N.: Differential evolution with combined variants for dynamic constrained optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 975–982. IEEE (2014)
- Ameca-Alducin, M.Y., Mezura-Montes, E., Cruz-Ramírez, N.: A repair method for differential evolution with combined variants to solve dynamic constrained optimization problems. In: Proceedings of 2015 on Genetic and Evolutionary Computation Conference, pp. 241–248. ACM (2015)
- 3. Campos, M., Krohling, R.: Bare bones particle swarm with scale mixtures of gaussians for dynamic constrained optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 202–209. IEEE (2014)
- Campos, M., Krohling, R.A.: Entropy-based bare bones particle swarm for dynamic constrained optimization. Knowl.-Based Syst. 000, 1–21 (2015)
- 5. Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, DTIC Document (1990)
- Cruz, C., González, J.R., Pelta, D.A.: Optimization in dynamic environments: a survey on problems, methods and measures. Soft. Comput. 15(7), 1427–1448 (2011)
- Darwen, P., Yao, X.: Automatic modularization by speciation. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 88–93. IEEE (1996)
- De, S., Pal, S.K., Ghosh, A.: Genotypic and phenotypic assortative mating in genetic algorithm. Inf. Sci. 105(1), 209–226 (1998)
- Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
- Filipiak, P., Lipinski, P.: Infeasibility driven evolutionary algorithm with feedforward prediction strategy for dynamic constrained optimization problems. In: Esparcia-Alcázar, A.I., Mora, A.M. (eds.) EvoApplications 2014. LNCS, vol. 8602, pp. 817–828. Springer, Heidelberg (2014)
- Filipiak, P., Lipinski, P.: Making IDEA-ARIMA efficient in dynamic constrained optimization problems. In: Mora, A.M., Squillero, G. (eds.) EvoApplications 2015. LNCS, vol. 9028, pp. 882–893. Springer, Heidelberg (2015)
- Grefenstette, J.J., et al.: Genetic algorithms for changing environments. In: PPSN, vol. 2, pp. 137–144 (1992)
- Ho, P.Y., Shimizu, K.: Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. Inf. Sci. 177(14), 2985–3004 (2007)

- Kundu, S., Biswas, S., Das, S., Suganthan, P.N.: Crowding-based local differential evolution with speciation-based memory archive for dynamic multimodal optimization. In: Proceedings of 15th Annual Conference on Genetic and Evolutionary Computation, pp. 33–40. ACM (2013)
- Li, C., Nguyen, T.T., Yang, M., Yang, S., Zeng, S.: Multi-population methods in unconstrained continuous dynamic environments: the challenges. Inf. Sci. 296, 95–118 (2015)
- Mahfoud, S.W.: Niching methods for genetic algorithms. Urbana 51(95001), 62–94 (1995)
- Mezura-Montes, E., Coello Coello, C.A., Tun-Morales, E.I.: Simple feasibility rules and differential evolution for constrained optimization. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 707–716. Springer, Heidelberg (2004)
- Morales, A.K., Quezada, C.V.: A universal eclectic genetic algorithm for constrained optimization. In: Proceedings of 6th European Congress on Intelligent Techniques and Soft Computing, vol. 1, pp. 518–522 (1998)
- 19. Nguyen, T.T.: Continuous dynamic optimisation using evolutionary algorithms. Ph.D. thesis, University of Birmingham (2011)
- Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evol. Comput. 6, 1–24 (2012)
- Nguyen, T.T., Yao, X.: Benchmarking and solving dynamic constrained problems. In: 2009 IEEE Congress on Evolutionary Computation, pp. 690–697. IEEE (2009)
- 22. Nguyen, T.T., Yao, X.: Solving dynamic constrained optimisation problems using repair methods. IEEE Trans. Evol. Comput. (2010, submitted)
- Nguyen, T.T., Yao, X.: Continuous dynamic constrained optimization-the challenges. IEEE Trans. Evol. Comput. 16(6), 769–786 (2012)
- Pal, K., Saha, C., Das, S.: Differential evolution and offspring repair method based dynamic constrained optimization. In: Panigrahi, B.K., Suganthan, P.N., Das, S., Dash, S.S. (eds.) Swarm, Evolutionary, and Memetic Computing. LNCS, vol. 8297, pp. 298–309. Springer, Heidelberg (2013)
- Pal, K., Saha, C., Das, S., Coello, C., et al.: Dynamic constrained optimization with offspring repair based gravitational search algorithm. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 2414–2421. IEEE (2013)
- Richter, H.: Memory design for constrained dynamic optimization problems. In: Di Chio, C., et al. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 552– 561. Springer, Heidelberg (2010)
- Salcedo-Sanz, S.: A survey of repair methods used as constraint handling techniques in evolutionary algorithms. Comput. Sci. Rev. 3(3), 175–192 (2009)
- Voigt, H.M., Lange, J.M.: Local evolutionary search enhancement by random memorizing. In: The 1998 IEEE International Conference on Computational Intelligence, pp. 547–552. IEEE (1998)