Evol. Intel.
DOI 10.1007/s12065-016-0146-1

RESEARCH PAPER

Towards a new Praxis in optinformatics targeting knowledge re-use in evolutionary computation: simultaneous problem learning and optimization

D. Lim[1] · Y. S. Ong[2] · A. Gupta[2] · C. K. Goh[2] · P. S. Dutta[2]

Received: 14 April 2016 / Revised: 8 September 2016 / Accepted: 28 September 2016

Abstract As the field of evolutionary optimization continues to expand, it is becoming increasingly common to incorporate various machine learning approaches, such as clustering, classification, and regression models, to improve algorithmic efficiency. However, we note that although problem learning is popularly used in improving the ongoing optimization process, little effort is ever made in extracting re-usable domain knowledge. In other words, the acquired knowledge is seldom transferred and exploited for future design exercises. Focusing on evolutionary optimization, in this paper we investigate the concept of simultaneous problem learning and optimization inspired by the following notions: (1) that prior/dynamically acquired knowledge can enhance the effectiveness of evolutionary search, and (2) that evolution can be geared towards gathering crucial knowledge about the underlying problem. Taking benchmark functions as well as an engineering (process) design problem into consideration, we demonstrate the efficacy of a novel classifier-assisted constrained EA towards simultaneous evolutionary search and problem learning.

Keywords Optinformatics · Evolutionary computation · Learning · Knowledge transfer · Constrained optimization

✉ A. Gupta
abhishekg@ntu.edu.sg

[1] Computational Intelligence Lab, School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore

[2] Rolls-Royce@NTU Corporate Lab c/o, School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore

1 Introduction

In recent years, the process of design optimization in science and engineering has been massively revolutionized by advances in computing technologies. Besides advancements in hardware, this phenomenon has also been endorsed by the fast-paced research and development in the area of *informatics*. State-of-the-art digital models, computer simulations, and advanced computational methods are being developed with the ultimate goal of arriving at more efficient and versatile products. In light of this fact, some recent works have introduced the term *optinformatics* [19, 35] which is defined as: *"introduction of the informatics specialization for mining the data generated in the optimization process, with the aim of extracting possibly implicit and potentially useful information and knowledge"*. In simple words, the term signifies the incorporation of problem learning on data generated during the search, to eventually achieve acceleration and/or better quality solutions from the optimizer.

The aforementioned situation is indeed realizable within the field of evolutionary computation (EC) which comprises a collection of nature-inspired population-based search algorithms that emulate evolutionary operators to adapt, learn, and innovate promising solutions. In this field, for decades, there has been a rising interest towards hybridized evolutionary algorithms. For instance, from the perspective of constrained optimization alone, methodologies at the intersection of tradtional evolutionary algorithms (EAs) and cultural algorithms can be found in [2, 24, 56]. In addition to combining meta-heursitics that complement one another, a common hybridization procedure is to incorporate machine learning techniques within evolutionary optimization in the spirit of Dawkins' notion of memes [7, 9, 10, 48]. The basic idea is to utilize

Published online: 11 October 2016

techniques such as response surface approximation, neural network prediction, and statistical models to dynamically acquire and exploit problem-specific knowledge so as to improve algorithm effectiveness. Since their emergence, most canonical EAs [21, 55] have been relying on fundamental statistics to model the fitness distribution of the population, which is eventually used to derive more promising descendants from the current parental population. Other examples include the use of response surface models, which are typically used in place of expensive and/or rugged objective functions [37, 38, 49, 58, 65, 66], or the various statistical models which have been embraced into estimation of distribution algorithms [34] as operators. In fact, machine learning techniques have appeared in every stage of the evolutionary process [29]. The knowledge extracted by these techniques provides potentially important and useful insights to guide present as well as future evolutionary searches [9].

It is common practice in industry to exploit knowledge from existing designs and adapt them according to different specifications. Therefore, gaining deeper insight about a particular problem (during the design optimization phase) is expected to be invaluable in facilitating future design pursuits for products/processes of a similar kind. Unfortunately, the current practice among researchers is to treat optimization and problem learning as two distinct entities that are independent from each other. Hence, in practice, it is common to find that problem learning is only used to improve the ongoing optimization process, and is seldom exploited to provide sufficient re-usable knowledge about the underlying problem at hand. *In other words, the acquired knowledge is typically not put to effective use for future design exercises, with the search beginning from scratch each time under assumptions of zero prior knowledge*. To this end, we propose in this paper the concept of simultaneous problem learning and evolutionary optimization. The rationale behind the present study stems from the realization that (1) prior/dynamically acquired knowledge of a specific problem can enhance the effectiveness of evolutionary search, and (2) evolutionary search can be geared towards gathering crucial knowledge about the underlying optimization problem. In particular, we focus herein on constrained optimization problems [5] as a case study since extracting re-usable knowledge about the feasibility structure of complex engineering design problems (which generally involve time consuming computer simulations and/or physical experiments [36]) is expected to be immensely valuable in many real-world applications.

At this juncture, it must be noted that while the present paper pertains to feasibility structure learning from constrained problems, there have recently emerged related works that highlight the general efficacy of knowledge extraction and transfer across distinct optimization instances [15–17, 46]. For example, the beneficial effects of identifying building blocks of knowledge from smaller problems and reusing them in higher complexity problems, have been showcased in [22, 23] for improved scalability in connection with large-scale Boolean tasks. Furthermore, cases of knowledge re-use in genetic programming-based symbolic regression have been presented in [6], while the potential utility of the proposition in the automated design of combinatorial optimization heuristics is discussed in [4]. However, to the best of our knowledge, little prior research has considered learning and transfer (or re-use) of models encompassing feasibility structures in constrained optimization problems, which underlines the key novelty of the present work.

The rest of this paper is structured as follows. In Sect. 2, the background on related topics, i.e., the use of machine learning tools within evolutionary optimization, is briefly reviewed. Section 3 forms the core of this paper. In particular, it presents the proposed simultaneous problem learning and evolutionary optimization framework, and an instantiation of it, i.e., a classifier-assisted constrained EA (CCEA) to solve constrained optimization problems. Subsequently, Sect. 4 presents empirical studies demonstrating the efficacy of the proposed framework. Computational experiments are performed on benchmark functions as well as on a real-world case study from the composites manufacturing industry. Finally, the conclusions and directions for future work are discussed in Sect. 5.

2 Machine learning and evolutionary optimization

In the past decades, EAs have emerged as a powerful paradigm for global optimization. Many successful applications of EAs have been reported, ranging from music composition [27, 44] to financial forecasting [1], aerospace design [11], job-shop scheduling [20], and drug design [33]. As the field continues to expand, there has been rising interest towards the development of hybridized EAs. In addition to being hybridized with other heuristic procedures, such as in memetic algorithms (MA) [39, 43, 47], a promising approach in hybridization is to incorporate machine learning techniques. By incorporating knowledge learnt from historical and/or dynamically acquired data during the evolutionary search, the hybridized EA promises better performance in terms of solution quality and search effectiveness. Among the plethora of machine learning techniques, the ones most often

involved in evolutionary search are clustering, classification, and regression [42].

Some notable studies on using machine learning to assist evolutionary search have been published in the recent decades. From a survey of literature, hybridization seems to have appeared in almost every aspect of the evolutionary process, ranging from *population initialization* to *evolutionary operators*, and *fitness evaluation*. For instance, in [53], population initialization of genetic algorithms (GAs) is biased using case-based reasoning. On the other hand, the informed operator framework [8, 54], generates many offspring using crossover or mutation operators, before utilizing a regression model of the objective function to filter out promising individuals. Meanwhile, the variety of estimation of distribution algorithms (EDAs), namely, iterated density estimation evolutionary algorithm (IDEA) [3], Bayesian optimization algorithm (BOA) [50], marginal histogram model [59], Voronoi-based EDA (VEDA) [45], and regularity model-based multi-objective EDA (RM-MEDA) [64], employ their own statistical models as evolutionary operators to bias the offspring reproduction. Similarly, the majority of surrogate-assisted EAs (SAEAs) [37, 49] use approximation models to replace computationally expensive fitness evaluations. Further under the category of SAEA, [26, 31] perform clustering of the population and restrict expensive evaluations only for the cluster representatives. The learnable evolutionary model (LEM) in [41] drafted a general framework for utilizing other machine learning techniques to extract information on promising individuals into evolutionary optimization. For a more substantial review, interested readers may refer to [25] for further details.

From the discussions above, it is clear that evolutionary optimization and machine learning techniques are inextricably intertwined. It is noted that in most of the aforementioned optimization methods, the common goal is to achieve superior search performance, i.e., in terms of solution quality and/or search efficacy. The opportunity of mining crucial knowledge about the underlying optimization problem, which may be useful for future design exercises, has only received lukewarm research attention till date. With the exception of a few recent works [22, 23], we find that there exists a scarcity of publications that may be considered in line with our idea of achieving global optimality together with mining of re-usable problem knowledge. A noteworthy method that has greatly motivated our work is that of efficient global optimization (EGO), an active learning approach proposed in [28]. EGO begins with a Latin hypercube sampling of the design space, followed by the construction of a Kriging/DACE approximation model. Subsequently, new solutions are generated in those regions of the search space where the so-called *expected improvement* is sufficiently high. The rationale behind doing so is that the expected improvement is typically large in those regions of the search space that are either under-sampled (therefore uncertain) or those that indeed contain potentially high-quality solutions. Thus, the steps involved in EGO serve the dual purpose of pursuing the global optimum while simultaneously unravelling the function landscape by sampling uncertain regions. To the best of our knowledge, such concepts have not been explored thoroughly in the context of evolutionary optimization. While successful hybridizations of active learning with EAs have been proposed in the context of computationally expensive multiobjective optimization [32, 63], the focus mainly lies in improving evolutionary search of the underlying problem, with little consideration for future optimization exercises. This observation inspires us to investigate the design of an evolutionary optimization algorithm that is capable of efficiently converging to the global optimum while simultaneously extracting valuable and re-usable knowledge about the underlying problem at hand.

3 Simultaneous problem learning and evolutionary optimization

3.1 Proposed approach

In this section, we outline the proposed simultaneous problem learning and evolutionary optimization (SPLEO) framework. The conceptual difference between traditional hybrid optimization and SPLEO is illustrated in Fig. 1a, b. In both frameworks, an evolutionary optimizer receives as input the knowledge acquired by the problem learning procedure. This knowledge could be formed from readily available data prior to optimization commencement, and/or from dynamically acquired data during the course of optimization. In most traditional frameworks, the evolutionary optimizer then utilizes the received knowledge to bias the search, with the provision for newly encountered data to be fed back to the machine learning procedure for further processing. In contrast, SPLEO emphasizes on the active interaction between the evolutionary optimizer and problem learning module, as shown in Fig. 1b. In every iteration of the optimization course, knowledge is fed to guide the optimizer. *Subsequently, the optimizer utilizes the knowledge to decide on which regions to exploit such that the search benefits the optimizer as well as problem learning*. One might argue that it is a trivial operation in traditional hybrid optimization to simply store every evaluated individual into a database for future learning. However, in doing so, it is possible that crucial regions of the search space (from the perspective of the learner) will remain under-explored during the optimization process.

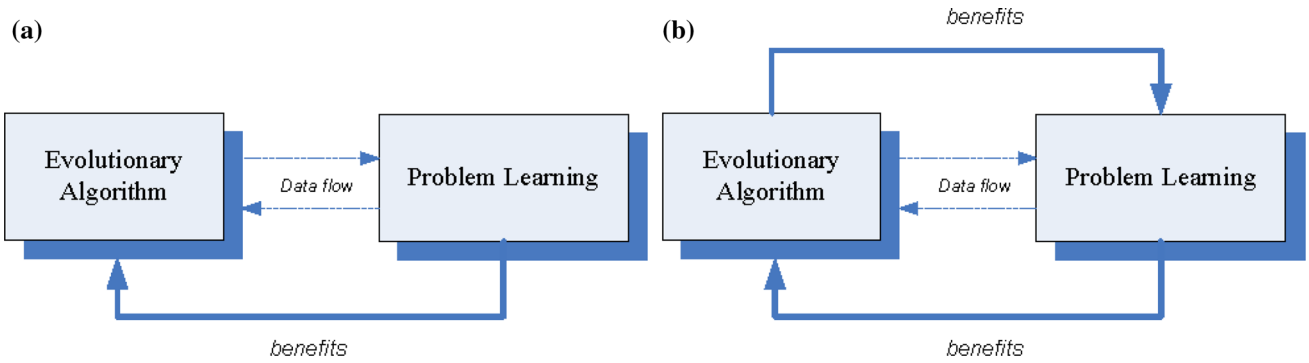

Fig. 1 Traditional hybrid evolutionary optimization framework versus simultaneous problem learning and evolutionary optimization (SPLEO) framework

Thus, the retained knowledge will not properly describe the underlying problem, and will consequently be of little use for future problem solving exercises.

3.2 SPLEO framework for constrained optimization

In the context of constrained optimization, knowledge about the feasible/infeasible separation boundary will enable designers to focus more towards promising regions of the search space, thereby avoiding several unnecessary and time consuming objective function evaluations. The fact that global optimum solutions are often located near the separation boundary between feasible and infeasible regions, where one or more constraints are active, further justifies the importance of extracting such knowledge. Since, there may exist only two classes of solutions, i.e., either feasible or infeasible, a binary classification system may be constructed for this purpose.

Based on the rationale above, we propose the classifier-assisted constrained evolutionary algorithm (CCEA). The main idea behind CCEA is to enable the evolutionary search to simultaneously achieve two major goals: (1) finding a globally optimal solution that satisfies all constraints, and (2) extract useful knowledge on the estimated feasible/infeasible regions. In particular, a conventional memetic algorithm (MA), hybridizing a genetic algorithm with local search, is used for this purpose.

In this paper, we focus on a general non-linear programming problem which is mathematically defined as:

$$\begin{aligned} &\textit{minimize} : f(\boldsymbol{x}), \\ &\textit{subject to} : g(\boldsymbol{x}) \leq 0 \textit{ and } h(\boldsymbol{x}) = 0 \end{aligned} \tag{1}$$

Here, $\boldsymbol{x} \in \mathbb{R}^d$ is the d-dimensional design vector, $f : \mathbb{R}^d \rightarrow \mathbb{R}^{nf}$ is the vector of n_f objective functions, $g : \mathbb{R}^d \rightarrow \mathbb{R}^{ng}$ is the vector of n_g inequality constraints, $h : \mathbb{R}^d \rightarrow \mathbb{R}^{nh}$ is the vector of n_h equality constraints. In particular, we consider $n_f = 1$ and $ng + n_h \geq 1$ in this paper. The workflow of CCEA is depicted in Fig. 2 and detailed further in Algorithm 1.

Algorithm 1 Classifier-assisted Constrained Memetic Algorithm

```
1: Generate and evaluate a population of design vectors
2: Update database and classifier with every newly evaluated design vector, x
3: while Computational budget is not exhausted do
4:    Generate offspring population using evolutionary operators
5:    Evaluate offspring population
6:    if Classifier exists then
7:       Test offspring population against classifier
8:    end if
9:    if No misclassified offspring OR classifier does not exist then
10:      Perform local search on random individuals
11:   else
12:      Perform local search on misclassified individuals
13:   end if
14:   Update database and classifier with every newly evaluated design vector, x
15: end while
```

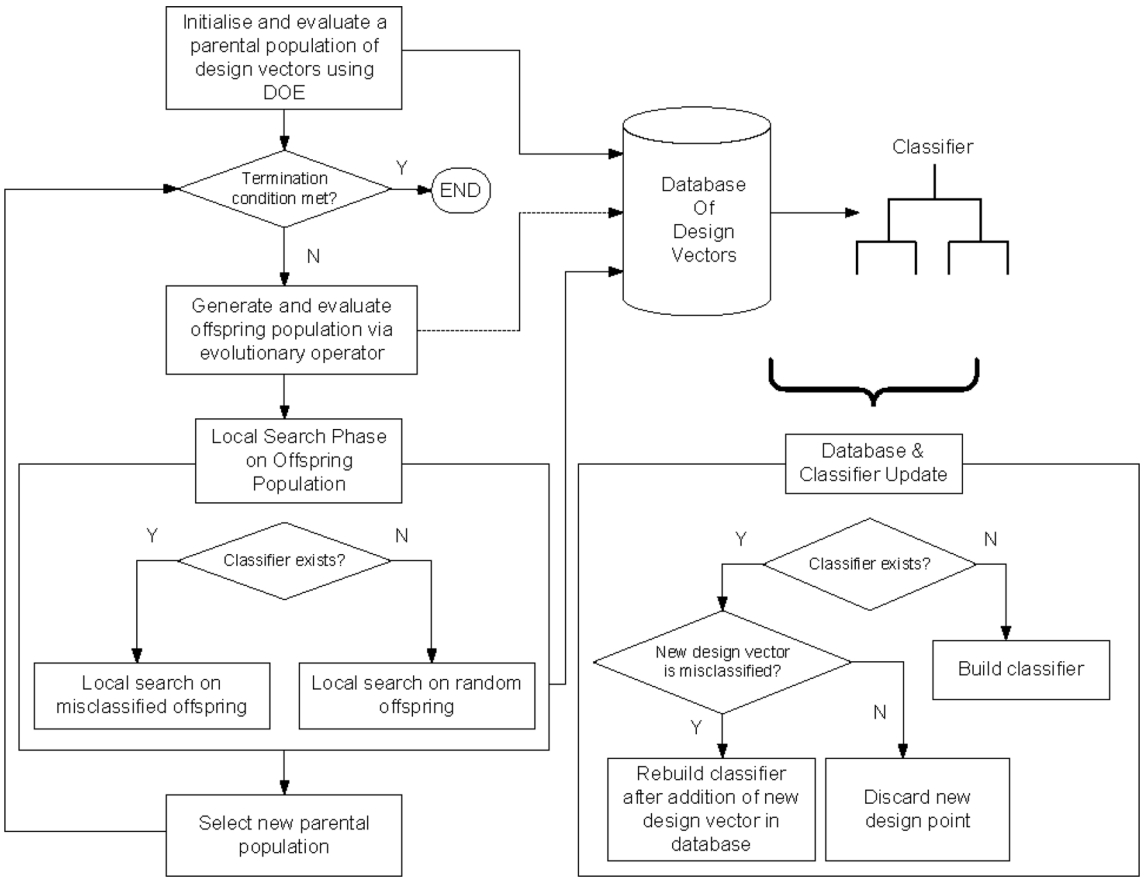

Fig. 2 Classifier-assisted constrained memetic algorithm

The search begins by initializing and evaluating a population of candidate solutions using design of experiment (DOE) technique. All evaluated individuals are archived into a database as training inputs for building a classifier based on their feasibility condition. At this stage, it is possible that no classifier is built if the database has only archived one class, i.e., all data are either feasible or infeasible. Subsequently, the optimization proceeds to the local search phase. During local search, if the classifier exists, it will be used to determine whether or not an individual undergoes local search. In particular, local search is performed on misclassified individuals so as to generate a larger sample of data points in the misclassified (therefore uncertain) regions. However, if the classifier is not built yet, or no misclassification occurred, local search is performed on randomly selected individuals. Note that every newly evaluated design vector during the search will update the database only if it is misclassified by the existing classifier (see Algorithm 2). This makes good sense since only the inclusion of misclassified design vectors will potentially induce changes to the existing classifier. Figures 3 and 4 illustrate two cases where the database update triggers changes in the separation boundary. In the first case (Fig. 3), the new design vectors lie near the separation boundary, hence their inclusion alters the corresponding boundary. On the other hand, Fig. 4 depicts the case when the new design vectors lie in an unexplored region, hence the classifier is updated with an additional separation boundary. In conclusion, it is worth noting that the emphasis of the algorithm is to explore two types of uncertain regions, namely, (1) those that are in the proximity of the separation boundaries, and (2) those that are likely to contain high quality solutions. By doing so, the algorithm can be expected to efficiently converge to the global optimum while simultaneously acquiring valuable knowledge about the feasible/infeasible regions of the underlying problem.

Algorithm 2 Procedure to update database & classifier in CCEA

```
1: if classifier exists then
2:     if new data is misclassified then
3:         Update classifier
4:     else
5:         Discard new data
6:     end if
7: else
8:     Attempt building classifier
9: end if
```

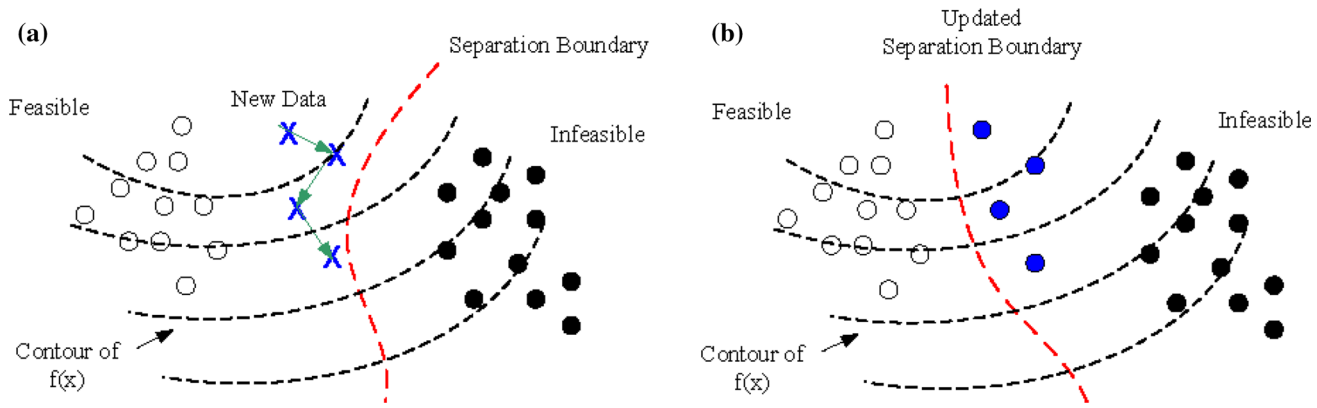

Fig. 3 First possible type of separation boundary update by the classifier in CCEA. **a** Original separation boundary, **b** updated separation boundary

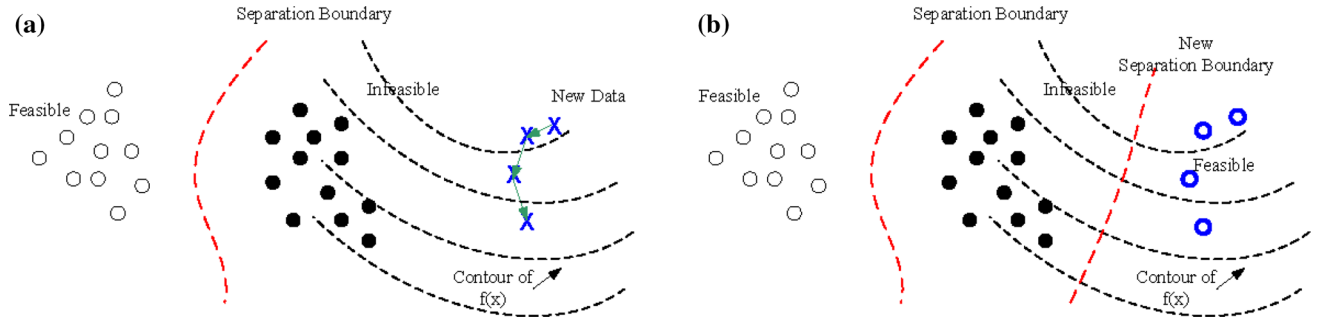

Fig. 4 Second possible type of separation boundary update by the classifier in CCEA. **a** Original separation boundary, **b** updated separation boundary

4 Empirical study

4.1 The local search scheme

The local search used in most conventional MAs occurs in the form of gradient-based, stochastic, or other numerical optimizers. Generally, the right optimizer for a problem can be chosen if the problem landscape is known beforehand. For instance, if a problem is known to be quadratic in nature, a quadratic programming optimizer would certainly excel. However, to reflect the real-world situation where the problem landscape is often unknown, we only employ a single type of local optimizer throughout a set of benchmark problems with diverse characteristics. Without loss of generality, in this paper, we consider the use of a gradient-based local search scheme. In particular, the steepest descent method is adopted for solving constrained optimization locally, as illustrated in Algorithm 3. During local optimization, if an infeasible design vector $\boldsymbol{x}$ is encountered, it is repaired by the procedure in Algorithm 4. In order to strongly emphasize on solution feasibility as part of the local search process, an improvement check is carried out to determine whether a newly constructed design vector is acceptable or not. For this purpose, a simple deterministic ranking scheme is employed throughout all experiments, according to which $\boldsymbol{x}$ is considered as improved based on the following rules:

- If $\boldsymbol{x}$ was originally infeasible, $\boldsymbol{x}$ is improved if it becomes feasible or its constraint violation decreases.
- If $\boldsymbol{x}$ was originally feasible, $\boldsymbol{x}$ is improved if it continues to be feasible and its objective value improves.

Algorithm 3 Steepest-descent based local search starting from a design vector $\boldsymbol{x}$

1: **if** $\boldsymbol{x}$ is infeasible **then**
2: Repair $\boldsymbol{x}$
3: **end if**
4: **while** termination condition is not met **do**
5: $\boldsymbol{x}_l = \boldsymbol{x} - \alpha \cdot \nabla f$, where $\nabla f = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_d}]$
6: **if** $\boldsymbol{x}_l$ is infeasible **then**
7: Repair $\boldsymbol{x}_l$
8: **end if**
9: **if** $\boldsymbol{x}_l$ is better than $\boldsymbol{x}$ **then**
10: $\boldsymbol{x} = \boldsymbol{x}_l$
11: **end if**
12: **end while**

Algorithm 4 Procedure to repair an infeasible design vector $\boldsymbol{x}$

1: **while** termination condition is not met **do**
2: $\mathbf{C} = [g_1, \ldots, g_{n_g}, h_1, \ldots, h_{n_h}]^T$
3: $\Delta \boldsymbol{x} = (\nabla \mathbf{C})^{-1} \cdot \mathbf{C}$, where $\nabla \mathbf{C} = [\nabla g_1; \ldots; \nabla g_{n_g}; \nabla h_1; \ldots; \nabla h_{n_h}]$
4: $\boldsymbol{x} = \boldsymbol{x} - \Delta \boldsymbol{x}$
5: **end while**

4.2 The classifier scheme

One of the major paradigms in machine learning is data classification. In simple words, classifiers map inputs into a set of pre-defined classes based on their attribute values. Given a training set $\mathbf{S} = \{\boldsymbol{x}_i, c_i\}$ for $i = \{1, 2, \ldots, m\}$, where m is the number of training data, $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ is a d-dimensional input, and c_i is the corresponding class. The goal of a classifier is to induce a model $\Phi(\mathbf{S})$ such that the following generalization error is minimized;

$$\varepsilon(\Phi(\mathbf{S})) = \sum_{i=1}^{m} \Psi(S_i, \Phi(S_i)). \quad (2)$$

Here, $\Psi(S_i, \Phi(S_i)) = 1$ if $c_i \neq \Phi(S_i)$, otherwise $\Psi(S_i, \Phi(S_i)) = 0$. Among the plethora of classification techniques, some prominent ones are the decision trees, support vector machines, artificial neural networks, and Bayesian networks. Without loss of generality, for all computational experiments carried out on benchmark functions, we make use of Quinlan's C4.5 decision tree classifier [52]. The C4.5 is an improved version of the original Iterative Dichotomizer 3 (ID3) [51], enhanced with new features such as ability to deal with both discrete and continuous parameter space, training data with missing attributes, and pruning after tree creation. A decision tree can be viewed as a recursive partition of the input space. Figure 5 presents a typical decision tree classifier for continuous parameter input space. It consists of test and leaf nodes. As the name suggests, the test nodes are meant to split the input space into two or more sub-spaces according to a predefined test condition. In the case of continuous parameter input space, this test condition refers to a range of values. On the other hand, each leaf node contains the specific class, which is perceived as the most appropriate class for the corresponding sub-space derived from preceding test nodes. Hence, in the context of continuous parameter input space, a decision tree consists of hyperplanes, each orthogonal to one of the axes.

While the apparent advantage of a decision tree lies in the ease of generating design vectors belonging to a particular preferred class, one major criticism is against its inability to accurately describe nonlinear decision boundaries. However, this is not expected to be a significant shortcoming considering the fact that a nonlinear hyperplane can be approximated by many instances of its linear counterpart.

4.3 Experiments

The performance of the CCEA is first and foremost compared against its underlying genetic algorithm with deterministic ranking (GA-DR), and a traditional memetic algorithm with deterministic ranking (MA-DR). The deterministic ranking, as described in Sect. 3, is based on three simple rules: (1) a feasible solution is always preferred over an infeasible one, (2) between any two feasible solutions, the one with better objective value is naturally

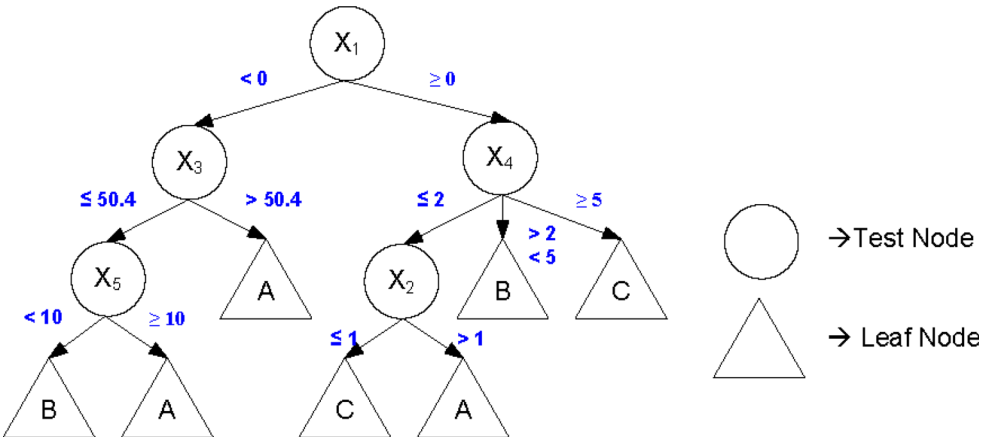

Fig. 5 Decision tree classifier in continuous input space

Table 1 Properties of the benchmark functions used in the empirical study

Benchmark problem	Dimensionality	No. of inequality constraints (n_g)		No. of equality constraints (n_h)		Feasibility (%)	Global optimum
		Linear	Non-linear	Linear	Non-linear		
F1	10	0	0	0	1	0.0000	−1.0005E+00
F2	5	0	6	0	0	26.9356	−3.0666E+04
F3	4	2	0	0	3	0.0000	5.1265E+03
F4	2	0	2	0	0	0.0064	−6.9618E+03
F5	10	3	5	0	0	0.0003	2.4306E+01
F6	2	0	2	0	0	0.8640	−9.5830E−02
F7	7	0	4	0	0	0.5256	6.8063E+02
F8	8	3	3	0	0	0.0005	7.0492E+03
F9	2	0	0	0	1	0.0000	7.4990E−01
F10	3	0	1	0	0	0.0197	−1.0000E+00

Feasibility data has been obtained from [61]

preferred, and (3) between any two infeasible solutions, the one with lower constraint violation is preferred. With regard to the MA, it can be seen as a GA-DR with local search or as the CCEA without an interacting classifier. In addition to the GA-DR and MA-DR, we consider four other algorithms from the literature on constrained evolutionary optimization, namely, evolution strategy with stochastic ranking (ES-SR) [57], Simple multi-membered evolution strategy (SMES) [40], Adaptive tradeoff model with evolution strategy (ATMES) [62], and the multi-objective hybrid constrained optimization EA (HCOEA) [61].

It is worth noting that since GA-DR and MA-DR are conceptually similar to the CCEA, albeit without the special feature of active interaction between problem learning and optimization, their comparison with the CCEA will highlight the contribution of the embedded features. In particular, performance comparison is carried out with respect to a set of commonly used benchmark problems, the properties of which are summarized in Table 1. Note that problems F1 to F10 considered herein correspond to problems `g03` to `g12` in [57], with all maximization problems transformed to minimization by multiplying the objective function by −1. The detailed mathematical expressions of the problems have not been replicated again in this paper for the sake of brevity. We also state that benchmark problems comprising small feasible regions have purposely been chosen so as to devise a stringent test for the proposed algorithm. The compared results are compilations of our own experiments (with GA-DR, MA-DR, and CCEA), and the results obtained from the literature.

The parameters settings presented in Table 2 are used for the CCEA, GA-DR, and MA-DR. The results reported in the literature are typically averages over 30 runs with 2.4E+05 function evaluations (SMES, ATMES, and HCOEA) or 3.5E+05 function evaluations (ES-SR) per run. Accordingly, in this paper, we choose to perform 30 repeat runs of 2.4E+05 function evaluations for CCEA, GA-DR, and MA-DR. Also note that the local search process in CCEA and MA-DR is set to terminate after a maximum of 10 iterations. Finally, the C4.5 classifier uses default parameters as suggested by [52].

Table 2 Settings of experiments for CCEA, GA-DR, and MA-DR

Parameters of CCEA, GA-DR, and MA-DR	
Population size	100
Crossover probability	0.9
Mutation probability	0.1
Maximum number of objective evaluations	2.4E+05
Evolutionary operators	Uniform crossover and mutation, elitism, and deterministic ranking selection.
Number of independent runs	30
Parameters of CCEA and MA-DR	
Local search iteration	10
Number of random individuals undergoing local search	10 %

1. Comparison of optimization results

Results obtained by the CCEA and the other six algorithms are statistically summarized in Table 3. From the *t* test results, several important observations can be drawn. Firstly, it is clear that the CCEA generally outperforms the underlying optimizer (the GA-DR) on top of which it has been built. Only in F10 do the two algorithms show no significant difference. This is so due to the simplicity of the problem, such that GA-DR can itself locate the global optimum without the need for any additional features. With regard to the MA-DR, although it shows much better performance compared to GA-DR, it is outperformed by (or is at most equivalent to) the CCEA in all cases as shown in Table 3. These observations strongly highlight the potential benefits of embedding local search (in MA-DR and CCEA) and SPLEO (only in CCEA) into the underlying GA-DR. It is clear that while MA-DR is an improved GA-DR (due to the local search), CCEA is a further improved version of MA-DR with the incorporation of the simultaneous problem learning and optimization framework.

From the results, it can also be seen that the CCEA performs competitively in comparison to other high-performing algorithms reported in the literature. This is inferred from their insignificant differences in 5 out of the 10 problems, in which they all can locate the global optimum efficiently. Nonetheless, it is worth noting their differences on problems F2, F3, F5, F7, and F8. With regard to F2, F5, and F7 in particular, it is interesting to find the CCEA to be slightly outperformed by other algorithms when dealing with high nonlinearity of constraints. This observation may be explained by the fact that the classifier currently being used in the CCEA, i.e., the decision tree, does not effectively locate separation boundaries when faced with high nonlinearity. In contrast, for problems F3 and F8 where several linear constraints exist, we find that the CCEA outperforms most of the algorithms from the literature.

Within the purview of the *no free lunch theorem*, which states that no single algorithm can excel on all problem instances, we contend that the computational results presented heretofore indicate significant promise in terms of the efficacy of the CCEA in locating globally optimal solutions. Nevertheless, it must be recalled that the purpose of the CCEA is not merely to optimize, but also to extract knowledge about the feasible regions of the search space for future use. The effectiveness of the algorithm with regard to knowledge extraction is discussed next.

2. Verifying the efficacy of knowledge extraction

Here, we measure the quality of the decision tree classifiers obtained from the CCEA. Traditionally, the quality of classifiers are measured by their accuracy, based on the

Table 3 Result of *t* test (with 95 % confidence level) while comparing performance of CCEA against GA-DR, MA-DR, ES-ER, SMES, ATMES, and HCOEA, on F1–F10

Benchmark problem	*p* value					
	GA-DR	MA-DR	ES-SR	SMES	ATMES	HCOEA
F1	**<0.0001(s+)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**
F2	**<0.0001(s+)**	**<0.0001(s+)**	0.0091(*s*−)	0.0091(*s*−)	0.0091(*s*−)	0.0091(*s*−)
F3	**<0.0001(s+)**	**1.0000(≈)**	**0.0004(*s*+)**	**<0.0001(*s*+)**	**0.0009(*s*+)**	**<0.0001(s+)**
F4	**<0.0001(s+)**	**0.1618(≈)**	**0.0047(*s*+)**	**0.1248(≈)**	<0.0001(*s*−)	<0.0001(*s*−)
F5	**<0.0001(s+)**	**0.0028(*s*+)**	<0.0001(*s*−)	**0.1248(≈)**	0.0002(*s*−)	<0.0001(*s*−)
F6	**0.0386(s+)**	**<0.0001(s+)**	**0.7422(≈)**	**0.7422(≈)**	**0.7422(≈)**	**0.7422(≈)**
F7	**0.0153(*s*+)**	**0.8255(≈)**	0.0014(*s*−)	0.0015(*s*−)	0.0014(*s*−)	0.0014(*s*−)
F8	**<0.0001(s+)**	**0.3191(≈)**	**<0.0001(*s*+)**	**<0.0001(s+)**	**<0.0001(*s*+)**	<0.0001(*s*−)
F9	**<0.0001(s+)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**
F10	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**	**1.0000(≈)**

s+, s−, or ≈ are used to indicate if the CCEA is significantly better, significantly worse, or equally good, respectively. Note that cases where the CCEA is significantly better or equally good are highlighted in bold

number of misclassified test items (refer to Eq. 2). However, considering that most of the benchmark problems have an unbalanced distribution of feasible and infeasible regions (as shown in Table 1), alternative measurements must be used. This is so since in the case of unbalanced distributions, one might always select instances from the majority class and be misled to the conclusion that the classifier is highly accurate. Hence, we employ the *sensitivity* and *specificity* measures [18] to determine accuracy. These can be formulated as follows:

$$sensitivity\,(\zeta) = \frac{P_{true}}{P}, \tag{3}$$

$$specificity\,(\xi) = \frac{N_{true}}{N}. \tag{4}$$

Here, P is the number of positive samples and P_{true} is the number of correctly classified positive samples. Similarly, N and N_{true} denote the number of negative samples and correctly classified negative samples, respectively. With regard to the classifiers obtained by the CCEA, the sensitivity and specificity refer to the accuracy on the feasible and infeasible classes, respectively.

To account for the size of the feasible region, as mentioned in Table 4, we randomly collected 70 infeasible and 30 feasible test samples for F2, and 90 infeasible and 10 feasible test samples for the other 9 benchmark problems. From the results tabulated in Table 4, several important observations can be drawn. First of all, it is clearly shown that CCEA is effective in mining important knowledge about the underlying optimization problem. Decision tree classifiers are built and rules on the feasible and infeasible regions are generated. For instance, the generated rule could be as simple as: *feasible region is located on* $C_1 \leq x_1 \leq C_2$ *and* $C_3 \leq x_2 \leq C_4$, as illustrated in Fig. 6.

Table 4 Statistics of the C4.5 decision tree obtained from the experiments

Benchmark problem	Algorithm	Feasibility (%)	Accuracy on feasible samples (ζ)	Accuracy on infeasible samples (ξ)
F1	CCEA	0.0000	**0.1300 ± 0.1784**	0.9896 ± 0.0140
	MA-DR		0.0167 ± 0.0379	0.9922 ± 0.0253
	(p value)		0.0012	0.6242
F2	CCEA	26.9356	0.5800 ± 0.1510	0.7890 ± 0.0567
	MA-DR		0.5500 ± 0.0852	**0.8095 ± 0.0316**
	(p value)		0.3472	0.089
F3	CCEA	0.0000	0.6233 ± 0.1888	0.9741 ± 0.0185
	MA-DR		0.6833 ± 0.2394	0.9963 ± 0.0670
	(p value)		0.2856	0.0855
F4	CCEA	0.0064	**0.7833 ± 0.3174**	0.9978 ± 0.0074
	MA-DR		0.0067 ± 0.0254	0.9978 ± 0.0085
	(p value)		<0.0001	1
F5	CCEA	0.0003	**0.9533 ± 0.073**	0.9781 ± 0.0248
	MA-DR		0.7800 ± 0.1064	0.9896 ± 0.0020
	(p value)		<0.0001	0.0528
F6	CCEA	0.0000	**1.0000 ± 0.0000**	0.9807 ± 0.0157
	MA-DR		0.8133 ± 0.1279	**0.9988 ± 0.0034**
	(p value)		<0.0001	<0.0001
F7	CCEA	0.8640	**0.4667 ± 0.3166**	0.9529 ± 0.0304
	MA-DR		0.1400 ± 0.2191	0.9433 ± 0.1060
	(p value)		<0.0001	0.6353
F8	CCEA	0.5256	0.4733 ± 0.3194	0.9833 ± 0.0148
	MA-DR		0.6067 ± 0.1530	**0.9993 ± 0.0028**
	(p value)		0.0436	<0.0001
F9	CCEA	0.0000	**0.8567 ± 0.1775**	0.7470 ± 0.1577
	MA-DR		0.2567 ± 0.1278	**0.9670 ± 0.0153**
	(p value)		<0.0001	<0.0001
F10	CCEA	0.0197	0.0733 ± 0.1258	0.9381 ± 0.0868
	MA-DR		**0.8200 ± 0.1584**	0.9633 ± 0.0097
	(p value)		<0.0001	0.1195

Note that significantly better values based on t test with 95 % confidence level are highlighted in bold

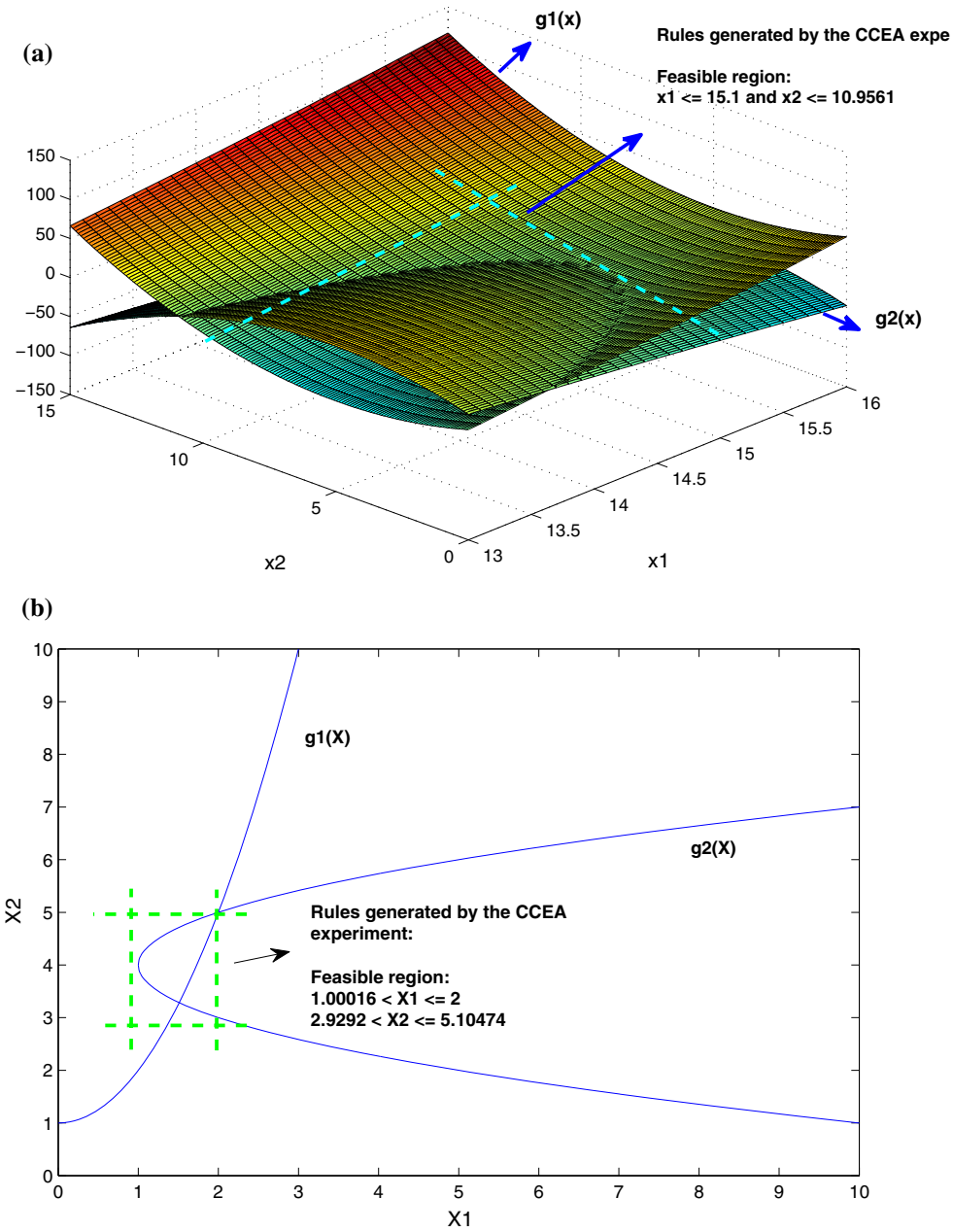

Fig. 6 Rules generated by the CCEA on low-dimensional benchmark problems with inequality constraints: **a** rules for F4, and **b** rules for F6

Since all benchmark problems considered are dominated by their infeasible regions, it is expected that a trained classifier will predict with high accuracy rate on infeasible samples. In contrast, a high accuracy rate while predicting the feasible samples is considerably more challenging to accomplish. From Table 4, on 8 out of 10 problems (F2–F9), we find that the accuracy achieved by the CCEA towards the feasible class is close to or greater than 0.5. In particular, the accuracy on problems F4–F6 and F9 exceeds 0.7. Only on two problems (F1 and F10), out of all ten examples, are the recorded accuracy levels low.

Previously, it has been shown that the CCEA generally outperforms MA-DR in terms of locating global optimums. Here, we further explore the effects of the SPLEO framework in comparison to MA-DR, this time with respect to efficacy of knowledge extraction. The improved problem learning as a consequence of the proposed algorithm is indeed confirmed in Table 4. Note that to arrive at this comparison, every evaluated design vector in the MA-DR is archived (together with its feasibility condition). The database is then used to build classifiers at the end of the search. This procedure is different from the mechanism of the CCEA where the classifiers are formed (and updated) simultaneously with the evolutionary search. Table 4 clearly shows that classifiers formed by the CCEA fare better than or equal to those formed by MA-DR at correctly

identifying feasible samples in 9 out of 10 cases. In contrast, MA-DR classifiers excel only at identifying the infeasible samples. From the perspective of constrained design optimization, it is however considered more meaningful to be able to make a quick estimate of the feasible designs.

Overall, the measurements presented in this subsection indicate that the feasibility structure has indeed been successfully extracted (to a large extent) by the classifier in the CCEA. It is contended that the availability of such knowledge is priceless for complex engineering design problems dealing with expensive computer simulations and/or physical experiments. In essence, the automatically extracted information from related design experiences will allow designers to focus more towards promising regions of the search space, thereby avoiding several unnecessary and time consuming objective function evaluations. A real-world manifestation of our claims shall be discussed next for a process optimization problem from the composites manufacturing industry.

3. Knowledge re-use in evolutionary optimization: a case study in composites manufacturing

For this case study, we consider two distinct composites manufacturing techniques that belong to the same family of rigid-tool liquid composite moulding (LCM) processes. To elaborate, (1) resin transfer moulding (RTM) and (2) injection/compression liquid composite moulding (I/C-LCM) are popular techniques for high volume production of synthetic fibre-reinforced polymer composite parts, and are characterized by the use of high stiffness moulds [12, 13]. Under the assumption of rigidity, the moulds are expected to undergo negligible deflection in response to the large internal forces that originate from the cumulative effect of high fluid (polymer resin) pressure and compression of the fibrous reinforcement. Often, sophisticated peripheral equipment (such as a hydraulic press) is needed to equilibrate the internal forces. Thus, an important aim in the optimal design of an LCM process is to maximize throughput (by minimizing manufacturing time), while satisfying the (often stringent) constraints placed by the availability or cost of peripheral equipment.

Before describing the constrained optimization problem, we present a brief overview of the RTM and I/C-LCM processes. The setup of the RTM process, as illustrated in Fig. 7, comprises a metal mould machined according to the geometry of the composite part to be manufactured. First, a preform of the fibrous reinforcement is placed into the mould cavity [step (a) in Fig. 7]. The mould is then fully closed, thereby compressing the preform to the final thickness of the part [step (b) in Fig. 7]. Prior to introducing the resin, the mould is preheated to a preferred operation temperature. Thereafter, a liquid thermosetting

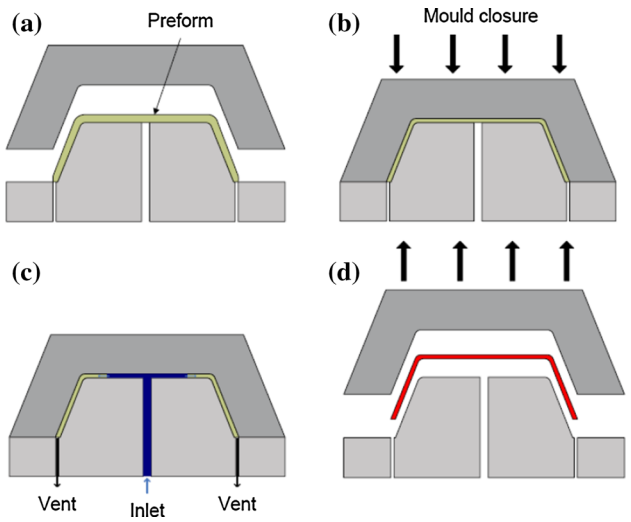

Fig. 7 Steps of the RTM cycle [60]: **a** preform placement, **b** mould closure to final part thickness, **c** resin injection, **d** resin solidification and part extraction

resin is injected into the mould at high pressure until the resin reaches the vents [step (c) in Fig. 7]. The filled mould is then allowed to rest until the resin solidifies, followed by part extraction [step (d) in Fig. 7]. The optimization of the RTM cycle introduces (primarily) three design variables, namely, (1) resin injection pressure (P_{inj}), (2) preheated mould temperature (T_{mould}), and (3) preheated resin temperature (T_{resin}). Thus, a design vector for the RTM cycle can be summarized as (P_{inj}, T_{mould}, T_{resin}).

While the filling of the mould in an RTM cycle is viewed as a single phase process, the same occurs in a two-phase manner in I/C-LCM. As illustrated in Fig. 8, during I/C-LCM, the mould is kept partially opened prior to resin injection [step (b) in Fig. 8]. After the necessary volume of liquid resin has been injected [step (c) in Fig. 8], the mould is fully closed to the final part thickness using a velocity-controlled mechanism [step (d) in Fig. 8]. Due to the inclusion of an in situ mould closure phase [step (d) in Fig. 8], the I/C-LCM cycle introduces two extra design variables (in addition to the three RTM variables), namely, (1) cavity thickness during resin injection (H_{inj}), and (2) velocity of final mould closure ($V_{closure}$). Thus, a design vector for the I/C-LCM cycle can be summarized as (P_{inj}, T_{mould}, T_{resin}, H_{inj}, $V_{closure}$).

While deciding on a manufacturing technique for a particular composite part, the manufacturer must thoroughly investigate both processes to determine the most suitable in terms of minimizing capital layout and running costs while maximizing throughput. In most cases, investigation is performed by coupling process simulation software [12] with an optimization engine [14]. To this end, the formulation of the constrained optimization problem may be stated as:

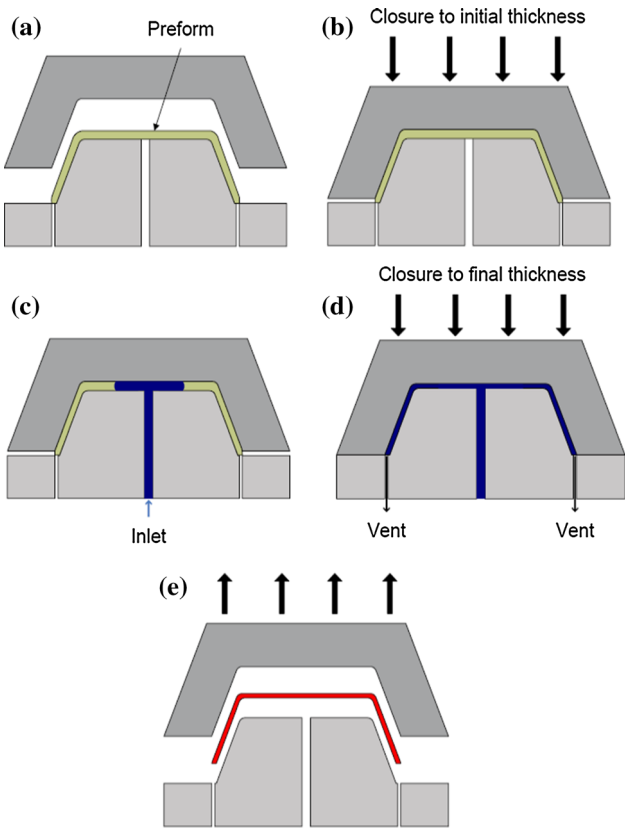

Fig. 8 Steps of the I/C-LCM cycle [60]: **a** preform placement, **b** mould kept partially opened, **c** resin injection, **d** in situ mould closure to final part thickness, **e** resin solidification and part extraction

$$\begin{aligned} &minimize(mould\,filling\,time),\\ &subject\ to: F_{fluid} + F_{fibre} \leq F_{capacity}. \end{aligned} \tag{5}$$

Here, F_{fluid} represents the internal force originating from the high pressure resin injection, F_{fibre} is the response of the compressed fibrous reinforcement, and $F_{capacity}$ is the prescribed capacity of the available hydraulic press. The values of F_{fluid}, F_{fibre}, and the *mould filling time*, for a given combination of the design variables, are obtained via the simulation software which numerically evaluates a series of governing partial differential equations (see "Appendix"). As is well known, such simulations are generally computationally expensive, thereby presenting a steep challenge to the task of optimization.

The described composites manufacturing problem provides an ideal setting for us to demonstrate the utility of knowledge transfer (or re-use) with regard to facilitating accelerated optimization performance. Based on the observation that RTM and I/C-LCM belong to the same family of rigid-tool LCM processes, there intuitively exists the possibility of useful knowledge transfer between the two techniques. Through a case study in simultaneous problem learning and evolutionary optimization, we show that this indeed is true.

Table 5 Extent of the search space for I/C-LCM design variables

Design variable	Lower bound	Upper bound
P_{inj} (MPa)	1	10
T_{mould} (K)	293	373
T_{resin} (K)	293	373
H_{inj} (cm)	0.8	1
$V_{closure}$ (mm/min)	0.5	5

We undertake the simulation-based optimization of the manufacturing process for a circular composite plate. The diameter of the plate is 1 m, with a central hole of 2 cm. The final thickness of the part is 0.75 cm. A glass-fibre chopped strand mat forms the reinforcing material with a final fibre volume fraction of 50 %, while an epoxy resin system forms the polymer matrix. For complete details of the material properties needed for simulations, the reader is referred to [13]. Details are not provided in this paper for the sake of brevity. Finally, it is stated that a hydraulic press of 20 ton (2E+05 N) capacity is made available for manufacturing, i.e., $F_{capacity}$ = 2E+05 N, thereby forming a highly restrictive constraint to be satisfied.

We first investigate the I/C-LCM process by optimizing it via the CCEA. The extent of the search space for the five design variables is reported in Table 5. In this example, we account for the computational expense of the simulation by employing a small population of 20 individuals that are evolved for a maximum of 4000 objective evaluations. The averaged convergence trend achieved by the CCEA is depicted in Fig. 9. Most importantly, the feasibility structure of the problem, as captured by the built classifier for I/C-LCM, can now be transferred across for the purpose of investigating the RTM process.

An illustration of a sample classifier constructed during the CCEA applied to I/C-LCM is depicted in Fig. 10. From an engineering perspective, Fig. 10 is found to contain useful and easily interpretable rules. For instance, following the right-most edge of the decision tree, it may be concluded that high resin injection pressure must be coupled with low resin temperature to assure design feasibility. Furthermore, following the left edges of the decision tree, it may also be inferred that maintaining the mould temperature higher than the resin temperature (i.e., $T_{mould} > T_{resin}$) is more conducive for a feasible process design. Interestingly, H_{inj} and $V_{closure}$ do not appear in the Fig. 10, implying that these variables play a lesser role in ensuring solution feasibility.

While optimizing the RTM cycle (with the standard GA-DR), we use the previously built classifier to generate an initial population that lies in the supposedly feasible region of the search space. Note that the classifier takes five design variables (P_{inj}, T_{mould}, T_{resin}, H_{inj}, $V_{closure}$) as input,

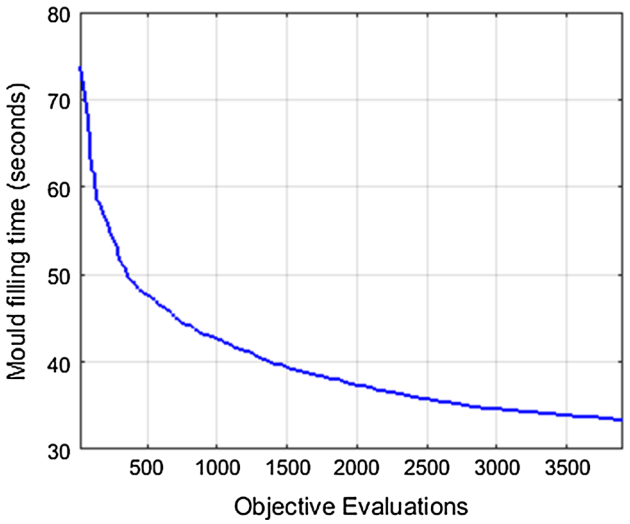

Fig. 9 Averaged convergence trend achieved by the CCEA for the I/C-LCM cycle

while the RTM process possesses only three design variables (P_{inj}, T_{mould}, T_{resin}). Thus, while initializing the population, we simply set $H_{inj} = 0.75$ cm and $V_{closure} = 0$.

In Fig. 11, we present a sample outcome of the (transferred) feasibility structure in the design space of the RTM cycle. The approximate global optimum solution is also highlighted therein. As is clear, the global optimum exists within the region of space that is a priori predicted to contain feasible solutions (thereby verifying the validity of the transferred knowledge). Notably, the volume of subspace that is predicted to be feasible is found to be comprise only 17.8 % of the entire design space. Thus, by eliminating a vast (inferior) portion of the design space from the evolutionary process, the learnt classifier (presented in Fig. 10) is expected to reduce the effort (and implicitly, enhance the effectiveness) of the search. This phenomenon is clearly observable in Fig. 12, where the strong impetus provided to the optimization algorithm is found to provide high quality solutions within remarkably fewer number of objective evaluations. In summary, the results strongly substantiate our claims that the re-use of extracted knowledge from previous problem solving experiences can be invaluable in terms of significantly reducing design time. It enables the designer to immediately focus the search towards promising regions of the search space, thereby avoiding several time consuming objective evaluations.

To conclude this section, we also provide evidence of the claim made in Sect. 3.2 about the likelihood of global optimum solutions existing near the separation boundary between feasible and infeasible regions. To this end, we refer to Fig. 13 depicting the *internal force versus time* curve of the best obtained design for the RTM cycle. As can be seen therein, the peak internal force is found to be equivalent to the specified capacity constraint of the hydraulic press ($F_{capacity} = 2E{+}05$ N). In other words, for a design to be optimum, it is shown that the design must

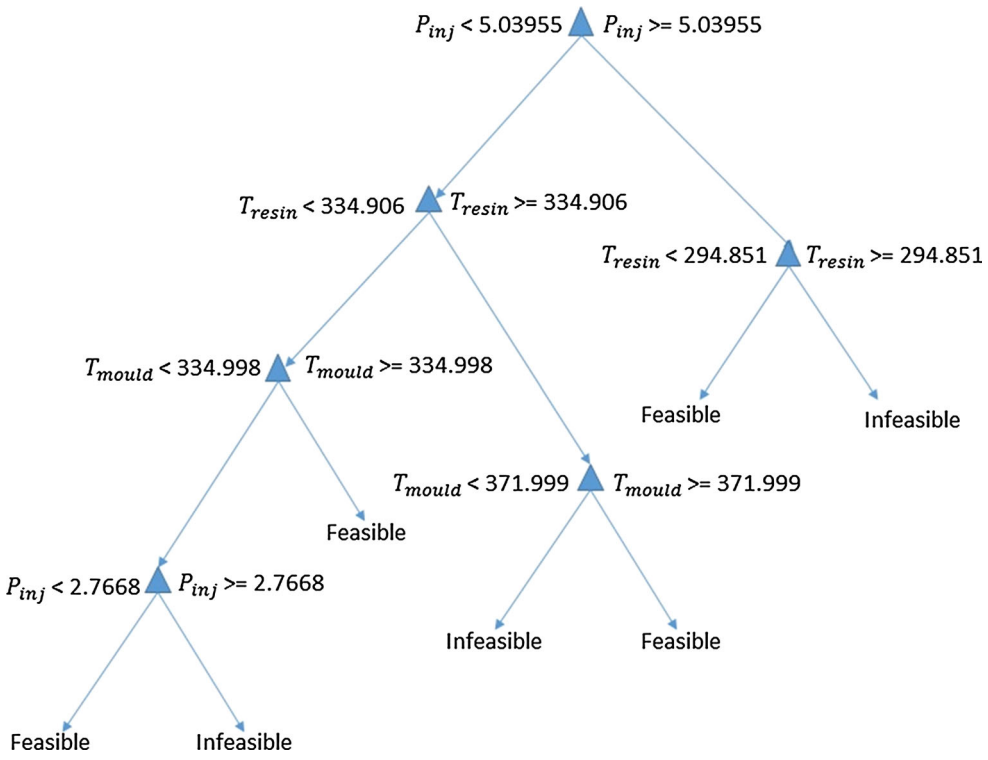

Fig. 10 Sample classifier constructed during application of CCEA to the I/C-LCM cycle

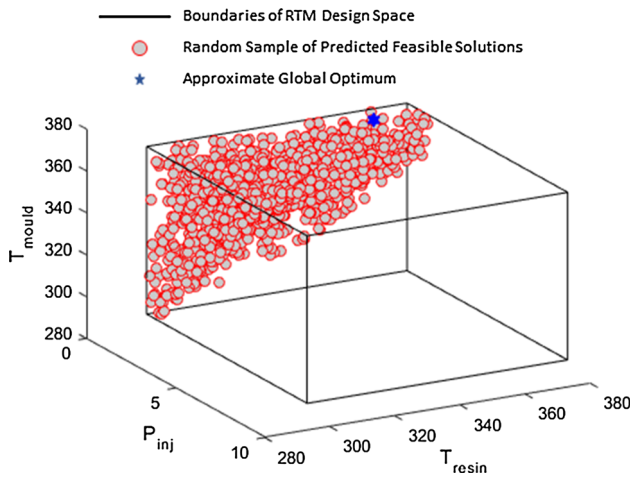

Fig. 11 Illustration highlighting the portion of the entire RTM design space that is predicted to be feasible by the classifier built by CCEA during I/C-LCM optimization. As shown herein, the approximate global optimum for RTM is indeed within the predicted feasible region (which constitutes only 17.8 % of the entire design space)

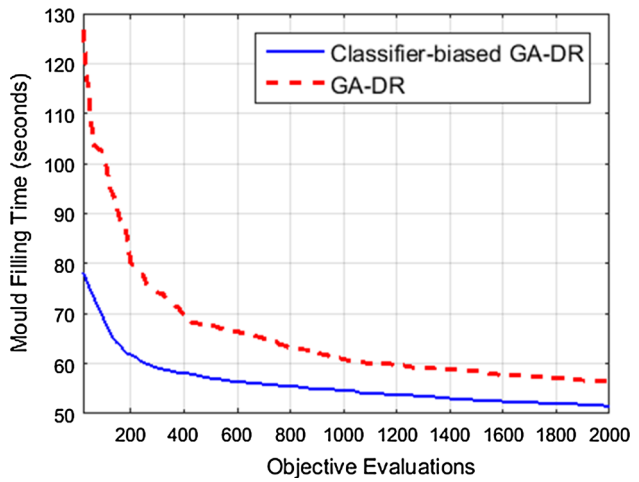

Fig. 12 Comparison of averaged convergence trends for the RTM cycle: GA-DR with random population initialization versus GA-DR with *classifier-biased* population initialization

fully consume the capacity constraint made available by the peripheral equipment (and thereby lie at the boundary of the feasible region).

5 Conclusions and future work

Using the proposed simultaneous problem learning and evolutionary optimization (SPLEO) framework in this paper, we have demonstrated how evolutionary search can be designed to gain better insight about the underlying problem while progressing towards a near-optimum solution. In particular, a realization of the framework in the form of a classifier-assisted constrained EA (CCEA) is

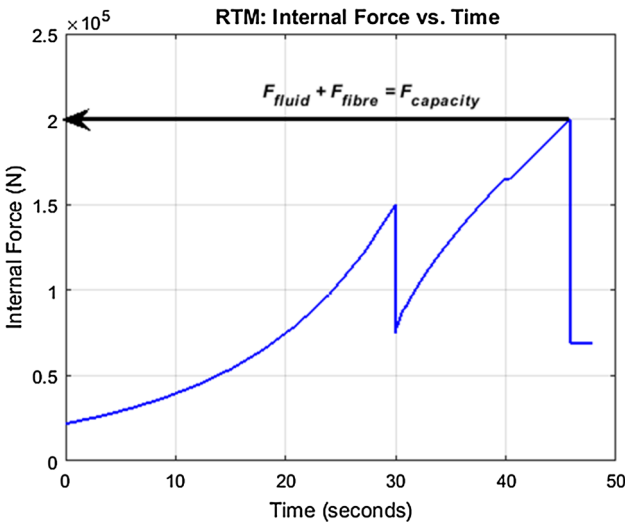

Fig. 13 Simulated internal force versus time curve for the RTM cycle. Notice that the prescribed capacity constraint of 2E+05 N is exactly satisfied

shown to simultaneously achieve two major goals: (1) find a globally optimal solution satisfying non-equality/equality constraints, and (2) extract knowledge on the estimated feasible/infeasible regions. While the computational experiments have shown promising results, there exist certain vulnerabilities that warrant future research attention. For instance, we have identified that the decision tree classifier being used with the CCEA may underperform when faced with high nonlinearity in the constraint functions. A trivial approach to mitigate this issue would be the testing of other classifiers.

In order to highlight the real-world significance of the contributions in this paper, we have carried out a case study in composites manufacturing. It has been shown that the feasibility structure extracted while investigating (by the process of optimization) a particular manufacturing technique may indeed be useful for accelerating the investigation of other possible manufacturing techniques. Thus, from the perspective of engineering design, the results demonstrate the potential benefits of harnessing knowledge from related problem solving experiences, particularly in terms of cutting down the (often overly expensive) design time.

Acknowledgments This work was conducted within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

Appendix

The filling phase of a generic composites manufacturing process is governed by the following PDEs [13]:

$$\nabla \cdot \left(h \frac{\boldsymbol{K}}{\mu} \nabla p \right) = \frac{\partial h}{\partial t}, \tag{6}$$

$$\rho C_p \frac{\partial T}{\partial t} + \rho_r C_{pr} (\boldsymbol{u} \cdot \nabla T) = \nabla \cdot (k \nabla T) + \left(1 - V_f\right) \cdot \dot{H}, \tag{7}$$

$$\varphi \frac{\partial \alpha}{\partial t} + \boldsymbol{u} \cdot \nabla \alpha = \left(1 - V_f\right) \cdot R_\alpha. \tag{8}$$

Equation 6 governs the fluid flow in porous media. Here, h is the thickness of the mould cavity, p is the local resin pressure, $\boldsymbol{K}$ is the reinforcement permeability, t is the time, and $\partial h/\partial t$ represents the speed of mould closure. Note that $\partial h/\partial t$ is zero throughout the RTM cycle, but is zero or strictly negative for the I/C-LCM cycle (due to decreasing cavity thickness during the in situ mould compression phase). The viscosity μ of the resin is a function of the local temperature T and the degree of resin conversion α. The relation may be captured by the following widely used rheological model,

$$\mu = A_\mu e^{E_\mu / RT} \left(\frac{\alpha_g}{\alpha_g - \alpha} \right)^{a + b\alpha}, \tag{9}$$

where α_g is the degree of cure at which resin gel conversion occurs, R is the universal gas constant, E_μ is the activation energy, and A_μ, a and b are other experimentally determined constants.

Equation 7 is a lumped energy equation which governs the temperature distribution within the mould. The material properties ρ, C_p, and k represent the average density, specific heat capacity, and thermal conductivity of the resin-fibre system, respectively. Further, $\boldsymbol{u}$ is the volume averaged resin flow velocity, V_f is the fibre volume fraction, and $\dot{H}$ is a source term representing the thermal energy generated by the resin during its exothermic polymerization reaction.

Finally, Eq. 8 models how the degree of resin conversion varies in the part during filling. Therein, R_α represents the rate of resin polymerization. Kamal and Sourour [30] proposed the following general model which is widely used to describe the polymerization reaction,

$$R_\alpha = \left(A_1 \cdot e^{(-E_1/RT)} + A_2 \cdot e^{(-E_2/RT)} \cdot \alpha^{m_1} \right) \cdot (1 - \alpha)^{m_2}, \tag{10}$$

where A_1, A_2, E_1, E_2, m_1, and m_2 are experimentally determined constants.

For complete details on the material properties and empirical constants used in the composites manufacturing case study, the reader is referred to [13].

References

1. Aranha C, Iba H (2009) The memetic tree-based genetic algorithm and its application to portfolio optimization. Memet Comput 1(2):139–151
2. Becerra RL, Coello CAC (2006) Cultured differential evolution for constrained optimization. Comput Methods Appl Mech Eng 195(33):4303–4322
3. Bosman PA, Thierens D (1999) An algorithmic framework for density estimation based evolutionary algorithms. Utrecht University Repository, Netherlands
4. Branke J, Nguyen S, Pickardt CW, Zhang M (2016) Automated design of production scheduling heuristics: a review. IEEE Trans Evol Comput 20(1):110–124
5. Cagnina LC, Esquivel SC, Coello CAC (2011) Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. Eng Optim 43(8):843–866
6. Chen Q, Xue B, Zhang M (2015) Generalisation and domain adaptation in GP with gradient descent for symbolic regression. In: 2015 IEEE congress on evolutionary computation (CEC), IEEE, pp 1137–1144
7. Chen X, Ong YS, Lim MH, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15(5):591
8. Elliott L, Ingham DB, Kyne AG, Mera NS, Pourkashanian M, Wilson CW (2004) An informed operator based genetic algorithm for tuning the reaction rate parameters of chemical kinetics mechanisms. In: Genetic and evolutionary computation conference, Springer Berlin Heidelberg, pp 945–956
9. Feng L, Ong YS, Lim MH, Tsang IW (2015) Memetic search with interdomain learning: a realization between CVRP and CARP. IEEE Trans Evol Comput 19(5):644–658
10. Feng L, Ong YS, Tan AH, Tsang IW (2015) Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems. Memet Comput 7(3):159–180
11. Giannakoglou KC, Papadimitriou DI, Kampolis IC (2006) Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. Comput Methods Appl Mech Eng 195(44):6312–6329
12. Gupta A, Kelly P (2013) Optimal Galerkin finite element methods for non-isothermal liquid composite moulding process simulations. Int J Heat Mass Transf 64:609–622
13. Gupta A, Kelly PA, Bickerton S, Walbran WA (2012) Simulating the effect of temperature elevation on clamping force requirements during rigid-tool liquid composite moulding processes. Compos A Appl Sci Manuf 43(12):2221–2229
14. Gupta A, Kelly PA, Ehrgott M, Bickerton S (2013) A surrogate model based evolutionary game-theoretic approach for optimizing non-isothermal compression RTM processes. Compos Sci Technol 84:92–100
15. Gupta A, Mańdziuk J, Ong YS (2015) Evolutionary multitasking in bi-level optimization. Complex Intell Syst 1(1–4):83–95
16. Gupta A, Ong YS, Feng L (2016) Multifactorial evolution: toward evolutionary multitasking. IEEE Trans Evol Comput 20(3):343–357
17. Gupta A, Ong YS, Feng L, Tan KC (2016) Multiobjective multifactorial optimization in evolutionary multitasking. IEEE Trans Cybern. doi:10.1109/TCYB.2016.2554622
18. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Amsterdam
19. Handoko SD, Kwoh CK, Ong YS (2010) Feasibility structure modeling: an effective chaperone for constrained memetic algorithms. IEEE Trans Evol Comput 14(5):740–758

20. Hasan SK, Sarker R, Essam D, Cornforth D (2009) Memetic algorithms for solving job-shop scheduling problems. Memet Comput 1(1):69–83
21. Holland JH (1962) Outline for a logical theory of adaptive systems. J ACM 9(3):297–314
22. Iqbal M, Browne WN, Zhang M (2014) Reusing building blocks of extracted knowledge to solve complex, large-scale boolean problems. IEEE Trans Evol Comput 18(4):465–480
23. Iqbal M, Browne WN, Zhang M (2015) Extending XCS with cyclic graphs for scalability on complex Boolean problems. Evolut Comput, Early Access article. doi:10.1162/EVCO_a_00167
24. Jin X, Reynolds RG (1999) Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach. In: Proceedings of the 1999 congress on evolutionary computation, CEC 99, vol 3. IEEE
25. Jin Y, Sendhoff B (2002) Fitness approximation in evolutionary computation—a survey. In: Proceedings of the genetic and evolutionary computation conference, Morgan Kaufmann Publishers Inc., pp 1105–1112
26. Jin Y, Sendhoff B (2004) Reducing fitness evaluations using clustering techniques and neural network ensembles. In: Genetic and evolutionary computation conference, Springer Berlin Heidelberg, pp 688–699
27. Johnson CG, Cardalda JJR (2002) Genetic algorithms in visual art and music. Leonardo 35(2):175–184
28. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Global Optim 13(4):455–492
29. Jourdan L, Dhaenens C, Talbi EG (2006) Using datamining techniques to help metaheuristics: a short survey. In: International workshop on hybrid metaheuristics, Springer Berlin Heidelberg, pp 57–69
30. Kamal MR, Sourour S (1973) Kinetics and thermal characterization of thermoset cure. Polym Eng Sci 13(1):59–64
31. Kim HS, Cho SB (2001) An efficient genetic algorithm with less fitness evaluation by clustering. In: Proceedings of the 2001 congress on evolutionary computation, vol 2. IEEE, pp 887–894
32. Knowles J (2006) ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans Evol Comput 10(1):50–66
33. Lameijer EW, Bäck T, Kok JN, Ijzerman AP (2005) Evolutionary algorithms in drug design. Nat Comput 4(3):177–243
34. Larranaga P, Lozano JA (2002) Estimation of distribution algorithms: a new tool for evolutionary computation, vol 2. Springer, Berlin
35. Le MN, Ong YS, Nguyen QH (2008) Optinformatics for schema analysis of binary genetic algorithms. In: Proceedings of the 10th annual conference on genetic and evolutionary computation, ACM, pp 1121–1122
36. Leifsson L, Koziel S (2016) Surrogate modelling and optimization using shape-preserving response prediction: a review. Eng Optim 48(3):476–496
37. Lim D, Jin Y, Ong YS, Sendhoff B (2010) Generalizing surrogate-assisted evolutionary computation. IEEE Trans Evol Comput 14(3):329–355
38. Lim D, Ong YS, Jin Y, Sendhoff B (2007) A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In: Proceedings of the 9th annual conference on genetic and evolutionary computation, ACM, pp 1288–1295
39. Meuth R, Lim MH, Ong YS, Wunsch DC II (2009) A proposition on memes and meta-memes in computing for higher-order learning. Memet Comput 1(2):85–100
40. Mezura-Montes E, Coello CAC (2005) A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Trans Evol Comput 9(1):1–17
41. Michalski RS (2000) Learnable evolution model: evolutionary processes guided by machine learning. Mach Learn 38(1–2):9–40
42. Mitchell TM (1999) Machine learning and data mining. Commun ACM 42(11):30–36
43. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms*C3P*. Caltech Concurr Comput Progr Rep 826:1989
44. Muñoz E, Cadenas JM, Ong YS, Acampora G (2016) Memetic music composition. IEEE Trans Evol Comput 20(1):1–15
45. Okabe T, Jin Y, Sendoff B, Olhofer M (2004) Voronoi-based estimation of distribution algorithm for multi-objective optimization. In: Congress on evolutionary computation, 2004. CEC2004. vol 2. IEEE, pp 1594–1601
46. Ong YS, Gupta A (2016) Evolutionary multitasking: a computer science view of cognitive multitasking. Cogn Comput 8(2):125–142
47. Ong YS, Keane AJ (2004) Meta-Lamarckian learning in memetic algorithms. IEEE Trans Evol Comput 8(2):99–110
48. Ong YS, Lim MH, Chen X (2010) Research frontier-memetic computation—past, present & future. IEEE Comput Intell Mag 5(2):24
49. Ong YS, Nair PB, Keane AJ (2003) Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA J 41(4):687–696
50. Pelikan M, Goldberg DE, Cantú-Paz E (1999) BOA: The Bayesian optimization algorithm. In: Proceedings of the 1st annual conference on genetic and evolutionary computation, vol 1. Morgan Kaufmann Publishers Inc., pp 525–532
51. Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106
52. Quinlan JR (2014) C4. 5: programs for machine learning. Elsevier, Amsterdam
53. Ramsey CL, Grefenstette JJ (1993) Case-based initialization of genetic algorithms. In: ICGA, pp 84–91
54. Rasheed K, Hirsh H (2000) Informed operators: speeding up genetic-algorithm-based design optimization using reduced models. In: Proceedings of the 2nd annual conference on genetic and evolutionary computation, Morgan Kaufmann Publishers Inc., pp 628–635
55. Rechenberg I (1965) Cybernetic solution path of an experimental problem (Royal Aircraft Establishment Translation No. 1122, B.F. Toms, Trans). Ministry of Aviation, Royal Aircraft Establishment, Farnsborough Hants
56. Reynolds RG, Michalewicz Z, Cavaretta MJ (1995) Using cultural algorithms for constraint handling in GENOCOP. In: Evolutionary programming, pp 289–305
57. Runarsson TP, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput 4(3):284–294
58. Sun G, Li G, Gong Z, He G, Li Q (2011) Radial basis functional model for multi-objective sheet metal forming optimization. Eng Optim 43(12):1351–1366
59. Tsutsui S, Pelikan M, Goldberg DE (2001) Probabilistic model-building genetic algorithms using marginal histograms in continuous domain. In: Proceedings of the international conference on knowledge-based and intelligent information and engineering systems, pp 112–121
60. Walbran WA (2011) Experimental validation of local and global force simulations for rigid tool liquid composite moulding processes. Doctoral dissertation, ResearchSpace@ Auckland
61. Wang Y, Cai Z, Guo G, Zhou Y (2007) Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Trans Syst Man Cybern Part B 37(3):560–575
62. Wang Y, Cai Z, Zhou Y, Zeng W (2008) An adaptive tradeoff model for constrained evolutionary optimization. IEEE Trans Evol Comput 12(1):80–92

63. Zhang Q, Liu W, Tsang E, Virginas B (2010) Expensive multiobjective optimization by MOEA/D with Gaussian process model. IEEE Trans Evol Comput 14(3):456–474
64. Zhang Q, Zhou A, Jin Y (2008) RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm. IEEE Trans Evol Comput 12(1):41–63
65. Zhou Z, Ong YS, Nair PB, Keane AJ, Lum KY (2007) Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Trans Syst Man Cybern Part C 37(1):66–76
66. Zhou Z, Ong YS, Nguyen MH, Lim D (2005) A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In 2005 IEEE congress on evolutionary computation, vol 3. IEEE, pp 2832–2839