Chapter 8
Evolutionary Ensemble Learning

Malcolm I. Heywood

Abstract Evolutionary Ensemble Learning (EEL) provides a general approach for scaling evolutionary learning algorithms to increasingly complex tasks. This is generally achieved by developing a diverse complement of models that provide solutions to different (yet overlapping) aspects of the task. This chapter reviews the topic of EEL by considering two basic application contexts that were initially developed independently: (1) ensembles as applied to classification and regression problems and (2) multi-agent systems as typically applied to reinforcement learning tasks. We show that common research themes have developed from the two communities, resulting in outcomes applicable to both application contexts. More recent developments reviewed include EEL frameworks that support variable-sized ensembles, scaling to high cardinality or dimensionality, and operation under dynamic environments. Looking to the future we point out that the versatility of EEL can lead to developments that support interpretable solutions and lifelong/continuous learning.

8.1 Introduction

Evolutionary Ensemble Learning (EEL) is taken to encompass the development of team behaviours that collectively solve a problem through a process of 'divide-and-conquer'. As such the terms *team* and *ensemble* will be used interchangeably. Team members will be referred to as *participants* or *agents* and a team must have more than one participant. The definition we will assume for a participant will take the following form:

Definition 8.1 Each participant (agent) performs an independent mapping from input (state) space to output (action) space.

Such a definition is adopted in order to distinguish participants from other approaches to 'divide-and-conquer' such as modularity, e.g. automatically defined

M. I. Heywood (✉)
Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada
e-mail: mheywood@dal.ca

W. Banzhaf et al. (eds.), *Handbook of Evolutionary Machine Learning*,
Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-3814-8_8

functions [115]. In short, the above definition implies that all participants are modules (because a participant never acts outside of a team), but not all modules are participants. Indeed, most instances of modularity follow a sequence of a callee referencing the module, passing a set of arguments, the module performing some computation and the callee using the returned variable(s) in some larger calculation. As such modules are typically representation-specific, whereas team participants are generally agnostic to the representation. Given such a stance, the focus of the chapter will be on the mechanisms which define the relationships between participants and therefore facilitate the processes of problem-solving through divide-and-conquer.

Historically, prior assumptions were often made about task decomposition and therefore team complement. For example, entirely homogeneous teams in a herding task [168] or requiring heterogeneous teams to consist of one instance of each agent 'type', e.g. writing and reading to memory [32] or classes in a classification task [147]. The 'level of selection' represents a reoccurring theme (Sect. 8.3), which is to say, does selection/variation/replacement appear at the level of team or participant? Initially, two basic approaches for evolving teams became established: the team as a single unit of selection versus sampling a participant from independent cooperative populations each time a team is composed.[1] Early examples in which teams were the unit of selection assumed a multi-tree representation, e.g. [81, 134]. At the same time, multi-population models (e.g. [87, 201]) increasingly became associated with cooperative coevolution and multi-agent systems (Sect. 8.4).

Diversity maintenance also represents a reoccurring theme, with different state representations having an impact on preferences for heterogeneous versus homogenous team compositions [134]. However, diversity maintenance is also linked to a desire to solve tasks of increasing difficulty [15, 133]. Hence, there is no need to use an ensemble if a single participant can solve the task, but if an ensemble is necessary, how can a *meaningful* division of duties across participants be achieved?

This chapter develops the topic of EEL through two basic perspectives that initially developed independently:

- Ensemble learning as applied to regression and classification or a supervised learning perspective, hereafter supervised EEL or **sEEL** (Sect. 8.2).
- Cooperative coevolution as applied to multi-agent systems or a form of reinforcement learning, hereafter **maEEL** (Sect. 8.4)

Participants in sEEL are always heterogeneous, whereas maEEL places more emphasis on finding m types of agent to appear in a team of n agents where $n \geq m$. The concept of a team also introduces the topic of 'level of selection' (Sect. 8.3) to the team or the agent (or both). Section 8.5 reviews research that attempts to extend the concept of an ensemble to variable-sized ensembles (hereafter **vEEL**), with the objective of further scaling the scope of tasks that EEL can be applied to. The chapter concludes with a summary of applications that potentially benefit from assuming ensemble formulations (Sect. 8.6) and a concluding discussion (Sect. 8.7).

[1] Also synonymous with the Pittsburgh versus Michigan approaches to learning classifier systems.

8.2 Ensembles for Supervised Learning

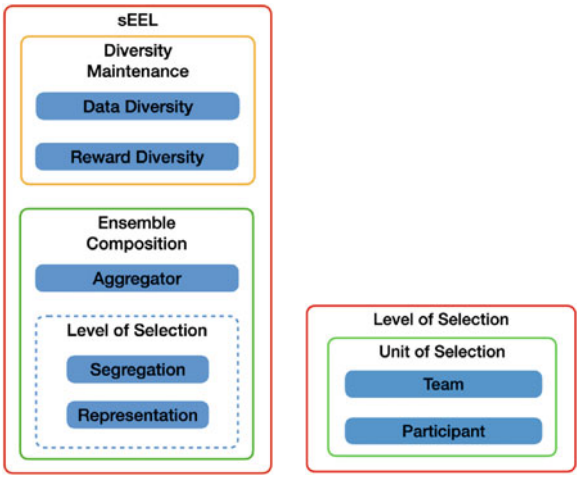
An ensemble under a supervised learning context is taken to imply a partnership between n decision-makers to solve a supervised learning task such as regression or classification. Classically, a bias–variance decomposition [33, 118] might be used to establish the extent to which error can be attributed to:

- the average behaviour differing from the desired function. Typically denoted the *bias* in which case under-fitting the data is said to result in a solution with 'high bias'.
- the ensemble is sensitive to the data set used to train the model. Typically denoted the *variance* in which case overfitting on training leads to high variance on test data.

Ensemble learning in general, therefore, attempts to compose a team of agents that exhibit diversity in their respective behaviours to optimize the bias–variance tradeoff. Such a framework has seen use with neural networks [29, 73, 78], genetic programming [2, 99, 100, 158] and 'wide' neural networks [24] as well as providing early insights to ensemble construction. With this in mind, ensemble learning as practiced outside of evolutionary learning often enforces diversity using one of three mechanisms:

- **Bagging:** n agents are independently constructed from n different samples (or 'bags') taken from the original training partition [33].
- **Boosting:** constructs n agents sequentially with the performance of earlier agents used to bias the selection of data to appear in the training partition for the next agent [34].
- **Stacking:** $n-1$ agents comprising the ensemble are trained on $n-1$ training folds. A "meta agent" is then trained from the $n-1$ agents" predictions to define the ensemble's overall prediction [221]. Other variants include cascading in which n agents are added sequentially, augmenting the data with their prediction [70].

Successful ensembles identify participants that are sufficiently accurate, yet 'disagree' with each other [118, 156]. As a consequence, many approaches have been proposed for maintaining ensemble diversity [119, 120]. However, diversity in itself is not a guarantee for an effective ensemble, i.e. diversity is relative to the behaviour of other participants comprising the ensemble. Moreover, ensemble learning as defined above implicitly assumes that only one candidate solution is developed at a time. Conversely, *evolutionary* learning algorithms maintain multiple candidate solutions simultaneously (the population). This implies that there are potentially more avenues for pursuing diversity and participant composition than under non-evolutionary approaches to ensemble learning. Assuming a broader perspective on diversity and/or composition enables us to identify five themes that sEEL might support exclusively or collectively, Fig. 8.1. We detail each of the five themes in Sect. 8.2.1 and make summary observations on sEEL in Sect. 8.2.2.



(a) sEEL (b) Level of Selection

Fig. 8.1 Properties of significance to Supervised Evolutionary Ensemble Learning (sEEL). **a** Diversity maintenance takes the form of data diversity (what data is an ensemble participant constructed from) and reward diversity (what is the performance function each participant experiences). Ensemble composition reflects (1) the diversity of mechanisms assumed for aggregating participants' predictions, (2) the degree of segregation appearing in the operation of participants and (3) the diversity in representations assumed for participants. **b** Level of Selection has an impact on segregation and representation (Sect. 8.3)

8.2.1 Diversity Maintenance in Supervised Evolutionary Ensemble Learning

Figure 8.1 divides sources of diversity in sEEL as applied to supervised learning tasks such as regression and classification into explicit 'diversity maintenance' versus 'ensemble composition'. Diversity maintenance is divided further into diversity through the data that different ensemble participants experience during training versus adaptation of the performance (reward) function, i.e. each participant potentially experiences a different performance function. Ensemble composition reflects different mechanisms by which the ensemble might be constructed. We divide this concept into three themes. Aggregation defines the mechanism assumed for combining the predictions from individual participants into an overall ensemble recommendation. Segregation characterizes how the role of participants might be revised during evolution and is related to credit assignment but reflects properties specific to evolutionary computation (e.g. the role of selection). Finally, representational diversity captures the ability of evolutionary computation to develop unique topologies as well as parameterize them. That said, most sEEL as applied to supervised learning tasks assume that the number of participants, n, is known a priori. The case of evolved participant complements and consequently context-specific participant deployment will be developed later (Sect. 8.5). In the following, we detail each theme in more detail.

Data diversity: implies that different participants of an ensemble are trained with different distributions of data. Adopting bagging strategies for ensemble learning represents a reoccurring theme [27, 68, 88]. Bagging might also be used with massively parallel computing platforms in order to accelerate the evolution of multiple participants from different partitions of the data, thus scaling sEEL to large regression and classification datasets [17, 18, 212]. Bagging methods were recently revisited for outlier reduction using niching [50], coevolution of participants and ensembles [170] and assessing participants over *multiple* bootstrap samples [214]. Likewise, partitioning the data into n folds (one fold held out from each data subset) enables n ensemble participants to be independently trained [90]. Competitive coevolution [69, 72, 126, 192] or active learning [83, 111, 185] have been used to adapt the data that ensemble participants experience *during* the course of evolution. That is to say, during evolution the most discriminatory exemplars will change as a function of the performance of the ensemble. In a further development, Lexicase selection represents a recent evolutionary selection mechanism for diversity maintenance using as little as a single exemplar to identify a parent [82]. Support for explicitly modular representations, such as sEEL, appears to provide a more effective mechanism for utilizing the available diversity [165]. Finally, ensembles have been incrementally constructed with the first participant evolved against the entire training partition. Thereafter, each additional ensemble participant is only evolved against the data that could not previously be labelled. This focuses each additional participant on what the ensemble cannot previously label [228].

Reward diversity: manipulates the performance function so that different participants of the ensemble experience different rewards. Early examples include boosting [68, 88, 162], cascade correlation [166], fitness sharing [172, 184] and negative correlation [128].[2] Other natural extensions include the use of multi-objective methods to trade off diversity and accuracy of ensemble participants [40, 41, 63, 123] and the simultaneous minimization of the resulting ensemble complexity [42]. Moreover, multi-objective performance criteria may represent a more robust predictor of (post-training) ensemble performance than ranking using a single objective [129]. Recently, novelty measures [37] and surrogate models [36] have been used to evolve ensembles for computer vision benchmarks such as CIFAR and SVHN. Under streaming tasks, other properties such as participant age have been used to prioritize participant replacement [67, 111]. Cooperative coevolutionary formulations imply that ensemble participants are sampled from n different populations [166]. The fitness that ensemble participants receive is used to direct development within each of the n different populations. This introduces issues regarding the fitness of ensembles versus participants (level of selection, Sect. 8.3) and a range of advantages and potential pathologies [159].

Ensemble aggregator: recognizes that once the participants of an ensemble are identified, then mechanisms need to be adopted to map from the n independent (participant) recommendations to a single (ensemble) recommendation [31]. Depending

[2] Negative correlation is related to the concept of covariance, the minimization of which potentially helps address the bias–variance trade off [41].

on the task, the optimal approach for combining recommendations might differ, i.e. majority voting, winner takes all [31, 85, 194] or Bayesian networks [198] in classification versus weighted average [47] and Bayesian model averaging [3] in regression. For example, an averaging assumption might penalize the development of specialists in regression tasks [161]. One potential approach for addressing this issue is to augment the ensemble with an additional participant (the 'gate'). Such a participant learns how to switch or blend the output from the n ensemble 'experts' [161]. Such a 'mixtures of experts' architecture can be deployed hierarchically [96] and has seen widespread use with neural networks [226]. Nguyen et al. go a step further and actively coevolve ensembles using the mixtures of experts framework [150, 151]. More recently, the concept of a convex hull has been deployed to identify the ensemble participants with Boolean operators defining the ensemble aggregation function [123]. Tsakonas and Gabrys use grammatical evolution to optimize a hierarchy of aggregation functions deployed to different combinations of ensemble participants [208]. The aggregator itself can be evolved [31, 121], where this would be synonymous with wrapper approaches to feature construction, i.e. the aggregator would be a regressor or classifier [77]. Evolving either linear or non-linear aggregators has been shown to be more effective than a 'fixed' voting scheme [31, 121]. However, there is a computational overhead associated with the case of evolved non-linear aggregators [121]. Finally, we note that solution interpretability is likely to be impacted by the choice of aggregator, i.e. a 'winner-takes-all' (versus majority) operator attributing a prediction to a single (versus all) participant(s) of an ensemble.

Participant segregation: captures the degree to which the n participants *evolve* independently and is influenced by the approach to selection (level of selection, Sect. 8.3). This is distinct but related to the degree to which the data (or performance function) is manipulated to establish n distinct behaviours. For example, n independent populations could be evolved on the same data (e.g. [90]) as opposed to evolving n independent populations on different samples from the training partition (bagging). This also leads to the use of libraries or archives of previous behaviours so that an evolutionary method can identify: (1) the participants to include in the ensemble from the library; and (2) how to weigh their respective contributions [27, 94]. Hybrid approaches have also been proposed in which: (1) a spatial embedding is used to define the migration protocol between independent populations [68]; or (2) variation is allowed between populations associated with different partitions of the data [27]. Alternatively, the champions from each of the n independent runs can be compared for their relative diversity and entire populations pruned should their champions be deemed too similar [38]. Rebuli and Vanneschi propose a model for multi-class classification with n demes, i.e. variation takes place between demes [167]. Later, a 'phase change' takes place after which the n demes are treated as islands, i.e. variation limited to the same population. Multifactorial evolutionary algorithms solve multiple optimization problems using a single population in which different 'tasks' are present. Unlike cooperative coevolution, sharing takes place between participants associated with different tasks, i.e. soft segregation. The approach has been demonstrated in the context of evolving ensembles for multi-class classification problems [217]. Finally, ensemble participants might instead be strictly organized as a stack/cascade with a

participant evolved to make a prediction or defer judgement to a later participant(s) [228]. Different subsets of the dataset are then labelled by different 'levels' of the stack/cascade.

Representational diversity: implies that participants comprising the ensemble are free to adopt different types of representation. We distinguish between coarse- and fine-grained representational differences. *Coarse-grained representational differences* imply that entirely different machine learning paradigms were deployed to develop participants, e.g. decision tree induction, Naive Bayes, multi-layer perceptron. Such an approach has been demonstrated for non-evolutionary machine learning [132], but could be achieved using a cooperative coevolutionary approach (different populations for each representation). Neural networks potentially benefit from assuming different architectures, thus diversity in the networks appearing within ensembles has been considered [128] as has diversity in the activation function [209]. In addition, rather than assume a behavioural performance metric to select ensemble participants, a genotypic performance metric might be preferred. As such, ensemble participants are selected for those that have the most unique representations [68, 85, 112]. Likewise, grammatical evolution has been used to evolve ensembles with different types of representation for rule induction [95]. Cloud-based computing services have also been used to simultaneously evolve multiple participants with different representations in parallel for solving large classification problems, e.g. learning classifiers versus genetic programming [18]. In a related approach, Fletcher et al. assume that multiple ensembles are trained, each with a different 'base classifier' and an evolutionary multi-objective approach is adopted to select the best ensemble [63]. Note that the base classifier is common to the same ensemble, but different across ensembles.

Fine-grained representational differences acknowledges the ability of the same evolutionary computational paradigm to evolve the topology of individuals as well as parameterize them. As such this will be impacted by the approach to selection or the level of selection, Sect. 8.3. Two general approaches have been widely adopted: **multi-trees** [31, 147, 194] (i.e. specific to genetic programming) or **cooperative coevolution** [159, 166] (i.e. agnostic to the underlying participant representation). Both approaches assume that the number of participants, n, is known a priori. For example, it might be assumed that the number of participants and classes requiring classification are the same. Section 8.5 reviews representations that evolve ensembles/teams that support a variable number of participants. This ultimately implies that participants can be deployed depending on input context rather than always simultaneously deploying all participants.

8.2.2 Discussion

Section 8.2.1 established that sEEL for supervised learning provides multiple paths by which diversity in ensemble learning can be maintained/introduced. Bagging and boosting can be considered specific mechanisms by which data and reward diver-

sity might be supported. Stacking—the third form of diversity classically recognized in non-evolutionary ensemble learning—appears as an instance of representational diversity. However, assuming that participants are 'evolved' using variable length representations such as genetic programming means that unique participant topologies can be discovered.[3] Representational diversity might additionally be considered when the number of participants, n, is not defined a priori as a hyper-parameter. Section 8.5 will develop this topic further. The property of participant segregation is specific to evolutionary computation on account of multiple participants being developed simultaneously. Diversity in the aggregation operation has seen a wide range of approaches, ranging from methods developed for Bayesian and neural networks to evolving the aggregator itself. We note, however, that such approaches are still limited to the 'classical' model of ensemble deployment: an ensemble consists of n participants and all participants participate in each prediction. We consider the implications of relaxing this constraint in Sect. 8.5.2 when graph-based representations are adopted.

A further set of observations can also be made independently of the specific diversity methods adopted, as follows:

- Strong ensemble performance does not imply that individual participants are also strong [31, 195]. Thus, there could be orders of magnitude differences between the performance of the ensemble and the participants comprising the ensemble. Indeed, selection for explicitly strong yet complementary ensemble members, as opposed to the original emphasis on ensembles composed from weak learners alone (e.g. [177]), represents a recent theme in sEEL [180–182].
- Participants of an ensemble are typically much simpler than when single 'monolithic' solutions were evolved for the same task. Depending on the task, the ensemble might also be collectively simpler than single monolithic solutions evolved for the same task [31, 88, 127]. Indeed, the participants of the ensemble might be more effective when simplicity is emphasized [123, 171].
- Assuming pairwise diversity measures does not necessarily lead to system-wide diversity [62]. Conversely, system-wide metrics, such as measuring the expected failure rate [90], have to date only been applied post-training. Using multi-objective formulations may benefit from defining the dominance relation on the basis of an entire performance characteristic, as opposed to single operating points [123], or modifying multi-objective formulations to incorporate validation data [176].
- Methods based on segregation and/or fixed partitions of training data are not able to adapt performance to changes in the behaviour of different ensemble participants. Adaptive reward functions might only be able to achieve this by rendering the training process serial, i.e. cascade correlation [166]. Conversely, serialization of the training process can result in considerable speedups when participants are able to distinguish between making a prediction versus deferring to another participant [228].

[3] Other successful ensemble methods such as Decision Forests also have this property.

- The multi-tree representation assumes that performance is evaluated at the 'level' of the team (Sect. 8.3). Constraints can potentially be enforced to ensure context between different participants. For example, under a class classification problem, crossover might be limited to exchanging material between participants labelling the same class [147]. Additional participant-specific performance functions might be included in order to penalize unwanted participant properties [205, 223], although the ability to do this might be limited to specific applications. Multi-trees can also be evolved to provide *multiple* layers of abstraction. Such an approach has been demonstrated to be particularly effective for designing operators for image processing [89].
- Coevolutionary formulations for composing an ensemble have to sample one participant from n different populations in order to construct a single ensemble. Fitness then needs to be interpreted at the level of individual ensemble participants, resulting in various potential pathologies. This is discussed further in Sect. 8.3 and potential solutions are reviewed in Sect. 8.4.

Finally, we note that an ensemble actually presents multiple decisions regarding the 'level of selection', i.e. the team versus the participant. This represents a theme common to both sEEL and maEEL. Section 8.3 will therefore develop this topic specifically before ensembles are discussed from the perspective of multi-agent systems (Sect. 8.4).

8.3 Level of Selection

The concept of level of selection reflects the two levels at which credit assignment operates when multiple agents are involved in making decisions. As per Definition 1, an 'agent' (or ensemble participant) in this context is a fully functional decision-making partner that performs a mapping from input (state) to output action. Such an agent might be a neural network, genetic program, decision tree, etc. Thus, given a pool of agents and a team/ensemble comprising of n agents, the level of selection problem reflects the following two issues.

Definition 8.2 Team composition: is the likelihood of mapping an agent to a 'position' in the team/ ensemble consisting of a fixed number of agents.[4] There are two extremes: all participants of a team are unique (*heterogeneous team composition*) or all participants of a team are the same (*homogeneous team composition*), Fig. 8.2.

Definition 8.3 Unit of selection: operates at the *level of agents* appearing within a team or at the *level of a team*, as shown in Fig. 8.2. Naturally, this implies that it is possible to express performance objectively at the level in question. Generally, the performance of a team can always be expressed objectively. However, depending on the task, expressing performance at the level of agents participating with a team may

[4] Section 8.5 considers the case of variable-sized teams.

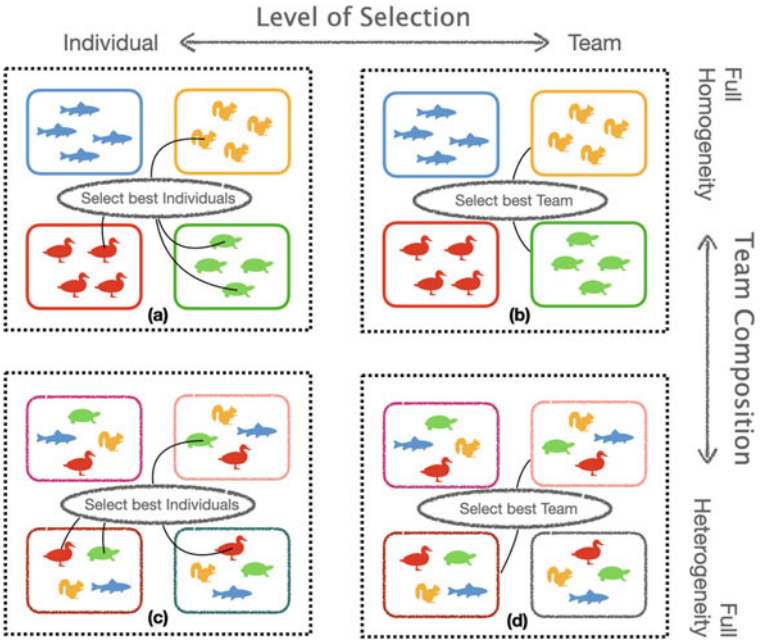


Fig. 8.2 Level of selection [215]. Full homogeneity (**a**) and (**b**) assume that all n agents per team are cloned from one of P genotypes. Full heterogeneity (**c**) and (**d**) assume M teams by n agents. Fitness evaluated at the level of individuals (**a**) and (**c**) versus team-level fitness evaluation (**b**) and (**d**). Reproduction at the level of individuals implies that two agents are chosen and an offspring agent results. Reproduction at the level of teams implies that two teams are chosen and variation operates at the level of team and possibly agent as well

or may not be possible. In the worst case, team and agent performance might not be aligned.

The previous discussion of ensembles (Sect. 8.2) implicitly assumed that agents participating within an ensemble were all different (heterogeneous). Moreover, when the multi-tree representation is assumed the unit of selection is typically that of the team [147]. Two parents are selected using team performance and crossover swaps team participants to create offspring, i.e. performed at the level of the team. In addition, 'strong typing' might also be assumed to direct the action of crossover such that only agents (sub-trees) with the same type are interchanged. Thus, for example, multi-trees applied to classification tasks might limit sub-tree crossover to classes of the same type [147]).

However, this need not be the case. Assuming that suitably informative performance functions could be designed for participants as well as at the complete ensemble/team, then the Orthogonal Evolution of Teams (OET) is able to [173, 204, 205]:

1. select parents at the level of participants but replace at the level of teams (OET1), or;

2. select parents at the level of teams but replace at the level of participants (OET2).

Such an 'orthogonalization' of the processes of selection and replacement was motivated to have credit assignment operate on the two levels simultaneously. Given an ensemble of size n there are n participant populations. Teams potentially represent columns sampled across the n participant populations [173]. Benchmarking with different classification tasks indicated preferences for different OET configurations [205]. However, OET2 appeared to be more effective at 'repair' when applied to a multi-agent foraging task [204]. We also note that the OET approach was independently discovered and used to evolve neural networks with an a priori fixed architecture [76], where each participant population was associated with a specific weight population and the ensemble was a complete network. A similar 'orthogonalized' process for the level of selection was used to direct the action of selection and replacement.

The concept of level of selection implies that decisions made regarding the team composition could have an impact on the degree of specialization versus the generality of agents supported in a team. For example, cooperative coevolution (Sect. 8.4) often assumes fully heterogeneous teams, making it difficult to establish teams composed of multiple instances of different types of agents. Moreover, as the level of selection is often that of the team, coevolutionary pathologies may arise such as:

- **mediocre stable states:** a form of deception in which agents collaborate to lead progress away from the global optima [74, 159].
- **relative overgeneralization:** agents with specialist behaviours are explicitly selected against [159].
- **loss of fitness gradient:** the performance of a few 'affective' agents is hidden by the poor performance of the majority of agents within a team. Also referred to as the 'signal-to-noise' problem [6].
- **hitchhikers:** in this case is synonymous with agents that exist within a team that does not contribute anything. Such agents reproduce, but do not contribute to the performance of the team [138, 141].

Section 8.4 revisits these pathologies in more detail under the context of multi-agent systems (cooperative coevolution is frequently used with multi-agent systems). Figure 8.2 summarizes the level of selection concept, albeit assuming 4 'types' of agent and teams of size $n = 4$. Waibel et al. performed a series of empirical evaluations of all four discrete parameterizations of team composition and level of selection using three tasks [215]. The tasks were designed to reward: (1) individual foraging, (2) cooperative foraging and 3) altruistic foraging. Agent specialization was not considered in these tasks. Heterogeneous teams with individual selection (Fig. 8.2c) were preferable when no cooperation was necessary. When cooperation is necessary, homogenous teams are preferable. However, the study came with some caveats, in particular teams were entirely homogeneous or heterogeneous. This means that it was not possible to construct teams with a instances of agent type i. Such hybrid team compositions might be considered the norm for composing optimal solutions to many tasks, such as multi-agent robotics.

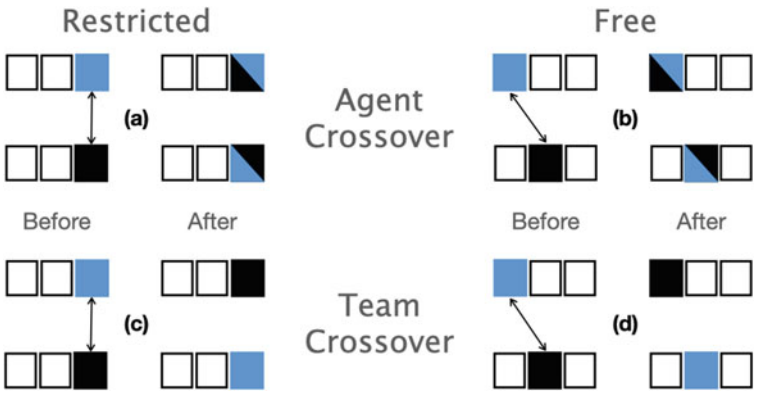


Fig. 8.3 Level of crossover [124]. Agent-level crossover **a** and **b** identify two agents within two teams and recombine the agent's genotypic material. Team-level crossover **c** and **d** identifies two agents within two teams and swaps the (entire) agent genomes. Restricted crossover **a** and **c** assume the position of the first agent. Free crossover **b** and **d** may choose agents from different team positions

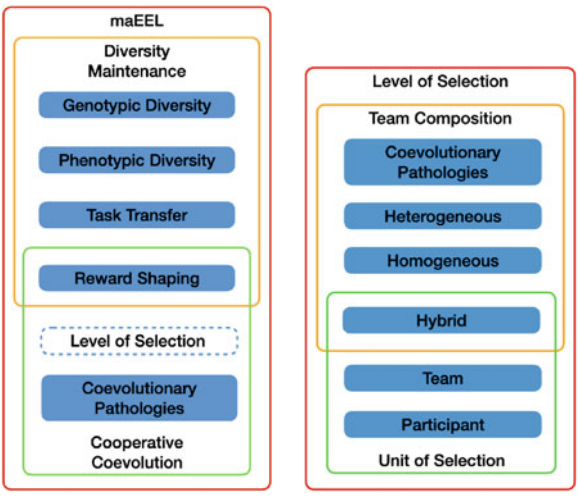
A further study considered the impact of variation operators under hybrid team compositions [124]. The authors set themselves the goal of attempting to evolve teams consisting of 1,000 agents, in which specific combinations of agent types have to be found given 10,000 distinct types of agents. Uniform crossover with free or restricted gene transfer (FAR and RAS respectively) was assumed (Fig. 8.3). The underlying conclusions were that RAS would converge quickly, where this would enable it to solve tasks involving many different teams (highly heterogeneous team compositions). However, when more homogeneous teams were required, the diversity maintenance provided by FAS was the most effective. In addition, the authors show that by deploying FAS for the early generations and RAS in the latter generations, then hybrid team compositions can be discovered. This topic will be particularly important when composing teams for multi-agent systems (Sect. 8.4).

Questions left unanswered include the relative impact of attempting to evolve both agent and team composition simultaneously and the impact of gene linkage during the course of evolving team composition. Also left unanswered is the effect of reinventing agent policies. Lichocki et al. concentrated on team composition [124], whereas agent discovery might benefit from the trading of generic abilities.

8.4 Multi-agent Systems and Cooperative Coevolution

Multi-agent systems attempt to solve a task cooperatively using a finite set of agents and are typically applied to reinforcement learning[5] (RL) tasks involving more than one decision-making agent. On the one hand, it is possible that a task might be solved optimally with the same agent deployed across the entire team or a purely homogeneous deployment. Conversely, at the other extreme, a task might be solved optimally with each agent comprising the team being unique (a purely heterogeneous deployment). Naturally, the number of agents, n, comprising the (multi-agent) team is known a priori. Thus, when describing the players participating in a soccer team, the number of players is known. Likewise, the number of robots available for performing a collective task might well be known. Under the sEEL context (Sect. 8.2) these issues are not as prevalent because the only team compositions that are appropriate are purely heterogeneous. Figure 8.4 summarizes the reoccurring themes that will be developed from an explicitly maEEL perspective.

Under a homogeneous deployment, one population is sufficient for sourcing the agents, and the concept of fitness at the level of team versus individual agent is aligned (Sect. 8.3). However, under heterogeneous settings, a multitude of possible mappings exist between population(s) sourcing the agents, and team composition (Sect. 8.3). At one extreme, a single population exists with each agent representing a participant of the multi-agent team. Under such a setting, incremental models of selection and replacement are assumed in order to gradually turn over the content of the population



(a) maEEL (b) Level of Selection

Fig. 8.4 Properties of significance to Multi-agent Evolutionary Ensemble Learning (maEEL). **a** Two major themes are identified: (1) Diversity maintenance is parameterized from the perspective of genotypic/phenotypic diversity, task transfer and reward shaping. (2) Cooperative evolution which is impacted by the level of selection (Sect. 8.3), coevolutionary pathologies and also reward shaping

[5] Reinforcement learning implies that an agent interacts with an environment and is rewarded for maximizing the cumulative rewards as experienced over a finite or unlimited number of interactions [200]. Applications include robotics, scheduling, game playing and stock trading.

and speciation/fitness sharing used to maintain population diversity, e.g. [197]. At the other extreme, each agent is associated with an independent population, this is the case of cooperative coevolution as formulated by Potter and De Jong [166].

To date, the cooperative coevolutionary formulation represents the typical starting point, Fig. 8.5. As such, one agent is sampled from each population in order to construct a single multi-agent team of n agents [166]. Thus, population i only ever source agents for 'position' i in the multi-agent team. Put another way, population i only ever source agents of 'type' i; therefore, the *context* between agent and position within the team is explicit. Reproduction and replacement only occur between participants of the same population. Moreover, cooperative coevolution is agnostic to the representation assumed to define the agents. Indeed, the populations associated with each agent (type) could have entirely different representations.

Naturally, performance is evaluated at the level of a team. However, fitness of agent i from an n agent team is a function of the other $n-1$ agents participating within the multi-agent team. As a consequence, N samples (and therefore fitness evaluations) are made of the agents from the other $n-1$ populations in order to establish the fitness of agent i. However, cooperative coevolution requires a measure of fitness to be returned to individual agents in order to direct the population-specific process of reproduction and replacement. At this point, pathologies can appear during credit assignment. For example, it became apparent that using the average (team) fitness from the N partnerships used to quantify the performance of agent i results in team compositions that favour mediocre stable states [74, 159]. In addition, relative overgeneralization may appear, where many individuals represent 'jack-of-all-trades' style solutions. This in turn precludes the development of specialists that could improve the overall collective performance of a team [159].

An early mechanism adopted for reducing these biases was to assign an agent its best fitness from the N partnerships as opposed to the average of all N partnerships [220]. This was later refined to using an annealing schedule to reduce the number of partnerships assessed as the number of generations increased [160]. Most recently, the

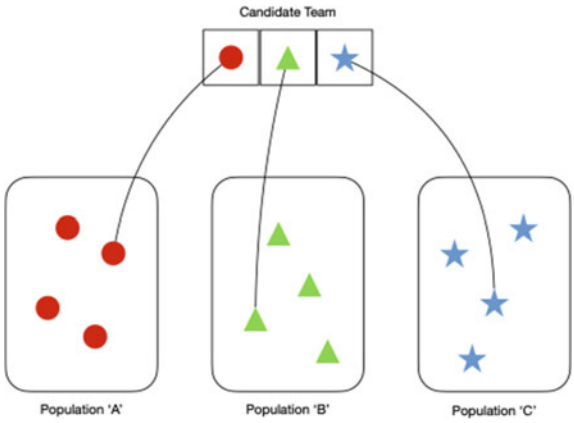


Fig. 8.5 Cooperative Coevolution [166]. Populations A, B and C only provide agents for team positions 1, 2 and 3, respectively. The context/type for each agent is therefore explicit. However, this forces teams to be heterogeneous. Evolving hybrid compositions requires the introduction of different team-level representations (Sect. 8.4.3)

issue of premature convergence in cooperative coevolutionary approaches to multi-agent systems has been addressed through the use of diversity measures (Sect. 8.4.1) and/or task variation (Sect. 8.4.2).

A related pathology that multi-agent systems can experience is with regards to a loss of fitness gradient. Specifically, as the team size increases, then the performance of one 'good' agent can be lost in the 'noise' created by all the poor-performing agents, i.e. a signal-to-noise problem. Attempting to address this problem by increasing the number of partners that agent i is evaluated with will not scale as the team size increases.

Agogino and Tumar proposed to address this problem by adopting factored (difference) performance functions [6]. This implies that for any state, the runner-up solution is known such that the effect of substituting the runner-up for the target agent can be estimated. Such functions have been demonstrated for a cross-section of applications involving agent deployments, e.g. sensor networks [4], air traffic control [5], multi-rover co-ordination [6].

Factored performance functions effectively reshape the reward such that improvements by a single agent also improve the multi-agent reward [44]. Shaping the reward in this way is easier to achieve when the agents are 'loosely coupled'. Loose coupling implies that the actions of one agent are not closely dependent on another, i.e. a form of gene linkage. It is more difficult to formulate factored performance functions when agents are tightly coupled [51]. For example, should one agent be doing something useful, such as attempting to push a highly valued object, unless the other agents also perform the same action, there might be no reward. This plays into being able to more explicitly control the degree of homogeneity/heterogeneity so that there are a instances of agent type i and b instances of agent type k. Hence, rather than attempting to evolve all agents independently, it might only be necessary to evolve 2 different agent types in a team of 20. Evolving teams with a hybrid mix of homogeneity/heterogeneity is discussed further in Sect. 8.4.3.

8.4.1 Diversity Maintenance

Diversity in cooperative coevolution can be promoted using behavioural (phenotypic) or genotypic properties at the level of team and/or agent. Diversity in the underlying team objective is often achieved by adopting multi-objective formulations in which several possibly conflicting objectives describe the underlying goal [43, 227]. Pareto formulations encourage tradeoffs between the different objectives to be investigated by different team complements. Moreover, they can also be used to provide a sequence of objectives of incremental difficulty that can lead to solving some (more difficult) overall objective [207].

Diversity maintenance represents a general challenge when evolving multi-agent systems. As such multi-objective methods have been widely adopted in an attempt to simultaneously develop task-specific objectives and promote diversity [144, 145]. Several approaches have appeared, including initially developing diverse behaviours

on a set of source tasks using multiple novelty objectives. The non-dominated individuals are used to seed the population for the target task(s) [143]. Conversely, Doncieux and Mouret include the task-specific objective throughout, but switch between different diversity objectives [54]. Such task switching has been recognized as a potential mechanism for promoting 'modular' solutions in general [164].

Several studies have demonstrated their applicability across a range of benchmark tasks: predator–prey [74], herding [74], multi-rover [74], half-field offence soccer [103] and Ms Pac-Man [103]. Genotypic diversity can be captured by measuring the pairwise similarity of team content between teams [74]. Moreover, such metrics can also be formulated for variable size teams [103]. Novelty metrics have been evaluated at the level of individual participants of a team as well as at the team level. A distinct preference for maintaining diversity at the level of the team has been reported [74, 152]. Moreover, experiments with and without behavioural diversity, genotypic team diversity and multiple source tasks indicate that the most general solutions appear when multiple forms of diversity appear [74, 103, 152].

8.4.2 Task Transfer

Task transfer (or layered learning) represents a mechanism for scaling learning algorithms in general and multi-agent (or cooperative coevolutionary) systems in particular to tasks that cannot be solved directly (*tabula rasa*), e.g. [178, 199, 202, 219]. This is also referred to as the bootstrap problem [143, 207]. As such, one or more source task(s) need identifying, typically a priori, with the evolution of the multi-agent system first performed on the source task(s). Policies or entire teams are then used as a 'run transferable library' for use during an independent evolutionary cycle conducted against a target task [101, 103]. The library might be used as seed material for initializing the population evolved against the target task, i.e. the agent-teams discovered under the source task are modified. For example, participants taking the form of code fragments[6] have been evolved using learning classifier systems on small-scale Boolean tasks and scaled to solve Boolean problems of larger dimension [13, 91]. Conversely, the solutions from the source task might be referenced by agents as evolved against the target task [101, 103], i.e. the solutions identified under the source task are not subject to variation during the development of the target task. The end result is an ensemble with a variable-sized structure that deploys solutions to source tasks in innovative ways to solve target tasks [108, 189] or the evolution of ensembles for solving multiple target tasks simultaneously [104, 108, 109] (reviewed in Sect. 8.5). In some cases, configurations of the task can be specified by members of an independent population. Competitive coevolution can then be used to establish an 'arms race' between candidate solutions (the teams) and the task [117]. This leads to a more open-ended process of team development [186, 190].

[6] Tree structured GP with a depth limit of two.

To date, task transfer under neural evolution tends to assume a population-seeding approach to task transfer [149]. Early examples evolved a neural network under a lower dimensional input space[7] and transferred this to a higher dimensional version of the task [22]. The HyperNEAT framework was assumed for this purpose, and teams were not explicitly present. However, this set the scene for use of a 'bird's eye' representation in which an explicitly multi-agent task (e.g. evolution of agents to play keepaway soccer) could be evolved to transfer between unrelated tasks [213]. One element of the approach was to reformulate the original egocentric task description to that of a two-dimensional global 'grid world' representation as viewed from above. HyperNEAT could then be used to scale to increasing numbers of players for the keepaway soccer benchmark task without target task training by adding a third dimension to the original bird's-eye view [45, 46]. The concept of a team is now a continuum. HyperNEAT represents an example of a developmental framework in which neural networks evolved with cyclic activation functions (denoted a Composite Pattern Producing Network, CPPN) describing the parameters appearing in the target architecture. The inputs to the CPPN represent the co-ordinates of the input and output of the target architecture. Adding a further HyperNEAT input to index the number of teams effectively scales the approach to arbitrary numbers of agents per team [45, 46]. Diversity was again a significant issue, with the combination of (task-specific) performance objectives and novelty search resulting in the most effective agents under the keepaway soccer task [152]. Such an approach rests on the use of (1) a bird's-eye representation and (2) HyperNEAT. For example, the bird's-eye representation removes some of the properties of the keepaway soccer task that made it challenging (e.g. navigation using an egocentric representation). HyperNEAT also composed solutions in the form of a 160,000 parameter feed-forward neural network, therefore losing solution transparency.

8.4.3 Hybrid Homogeneous–Heterogeneous Multi-agent Systems

The 'signal-to-noise' pathology in multi-agent systems (cooperative coevolution) can potentially be addressed by explicitly supporting the evolution of hybrid team compositions (see also Sect. 8.3). Thus, a team of 11 soccer-playing agents could be parameterized by specifying four types of agents (goalie, defender, mid-field and striker) and the number of each type of agent evolved. Nitschke et al. adapted the classic cooperative coevolutionary framework of Potter and De Jong [166] (fully heterogeneous) to address this issue by introducing an inter-population crossover operator [153, 154]. To do so, the genotypic and behavioural similarities between different populations are measured. This implied that a particular neural encoding had to be adopted. When the inter-population similarity passes a threshold, then

[7] Representing a board game with complete state information.

crossover of material between populations can take place. There are still as many populations as agents, but subsets of populations can now influence each other.

Early examples of evolving hybrid team compositions specific to genetic programming include the use of an 'automatically defined group' [79] and the 'Legion system' [30], both of which assume a tree-structured representation. Automatically defined groups rely on special purpose crossover operations to manage the development of teams over multiple level of selection. The Legion system not only relied on specialized crossover operators (specific to tree-structured genetic programming) but also introduced an entropy based heterogeneity measure in order to encourage/reward the division of labour between agents.

More recently, Gomes et al. explicitly define a team encoding to distinguish between agent type and the number of instances of each agent [75]. Specifically, the Potter–De Jong cooperative coevolutionary framework is still assumed, but this time the number of independent populations reflects the number of agent types. One set of variation operators functions at the team level and another set operates at the agent level [75]. Team-level variation can decrease or increase the number of agent types, thus merges or splits the corresponding agent populations. The approach is still independent of the agent representation, but the same representation has to be employed throughout.

A further aspect of the signal-to-noise pathology is that there are two components to the reward function: a 'high-frequency' component and a 'low-frequency' component. The high-frequency component is associated with the agent to environmental interaction, i.e. reinforcement learning [200]. The low-frequency component is associated with satisfying multi-agent components of the reward. With this in mind, neuro-evolutionary approaches have been proposed in which gradient-based temporal difference methods are used to optimize properties of individual agents, while evolutionary computation is employed to design the team [110]. Naturally, such an approach assumes a real-valued numerical representation [218] in order to support both high-frequency (gradient decent) and low-frequency (evolutionary computation) credit assignment.

Finally, we also note the use of 'tagging' to dynamically identify which team a participant belongs to [86, 169]. Thus, participants are assigned on the basis of the similarity[8] of their respective tag values. This method of dynamic team selection has been extensively analysed within the context of the iterated prisoners dilemma [20]. In particular, only members of the same group play each other, resulting in participants increasingly adopting altruistic strategies as opposed to defector strategies as the number of tags increases. This is to say, the altruistic participants learn to increase the number of teams in order to decrease the likelihood of their team including a defector. More recently, Lalejini et al. used tags to identify the conditions under which agents were associated with states. This enabled agents to evolve 'event driven' decompositions of tasks [122].

[8] The similarity metric could also be probabilistic, resulting in a source of variation in participant-to-team association.

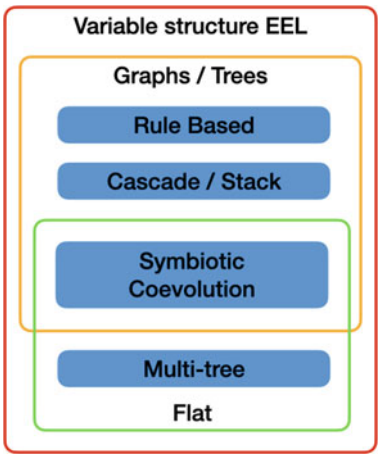


Fig. 8.6 Properties of significance to Variable-sized Evolutionary Ensemble Learning (vEEL). Flat implies that the ensemble is organized with all agents participating in every decision. Graph/Tree implies that ensemble participants are organized hierarchically with different subsets of individuals participating in decisions depending on the input state

8.5 Ensembles with Variable Size-Structures

All the above research assumed ensembles/multi-agent systems in which the number of participants/agents per team was specified a priori. However, it might actually be desirable for this to evolve as well. Figure 8.6 highlights themes of significance for evolving variable-sized evolutionary ensemble learners (vEEL).

One approach to vEEL might be to repeatedly evolve a fixed-sized ensemble approach (Sect. 8.2) over a range of ensemble sizes. Naturally, this would incur a significant computational overhead. Multi-tree representations have been proposed for evolving teams of up to n participants by introducing a 'null' program at the sub-tree level [21]. Multi-objective archiving has also been used to cooperatively evolve ensembles of classifiers for multi-class [139] and binary [25, 26] classification problems. As such the complexity of the resulting archive is a function of the task difficulty, i.e. the number of participants per class is an evolved property. Such an approach deploys participants in parallel (or 'Flat' in Fig. 8.6). Conversely, at the other extreme, participants might be organized as a hierarchy or a cascade [70]. Potter and De Jong assumed the specific case of cascade correlation in order to let cooperative coevolution incrementally evolve a neural network without a priori specifying the number of hidden layer neurones [166]. However, this solution was specific to the case of neural networks with a cascade correlation topology [61], so the coevolutionary process was no longer agnostic to the representation assumed for participants. A further approach to cascade/stack construction has been proposed in which participants distinguish between making a prediction or not [228]. If a prediction is made, no further participants need to make a decision. If a prediction is not made, then the next participant in the hierarchy is queried.

Sections 8.2 and 8.4 for the most part assumed that all participants collaborated at the same level (parallel/flat agent deployment). Conversely, graphs have the ability to describe hierarchical relationships and enforce spatial and/or temporal dependencies

between different participants. Graphs have previously been used as an organizing principle for instructions within programs (e.g. [135, 193]) or state machines (e.g. [16, 91]). However, two works have also considered using conditional statements attached to a 'header' of ensemble participants to define which out of several 'following' participants to execute: PADO [203] and Linear Graph GP [97]. In both cases, given a start or root participant, the participant is executed and the participant's conditional statement is assessed. Depending on the conditional statement, either another participant is executed or not. The conditional statements from PADO assess the state defined in a common memory (to all participants), whereas the conditionals of Linear Graph GP assess the register values of the parent participant. As such, a single participant is associated with graph nodes and arcs are associated with each condition statement. The concept of a team is therefore 'distributed' across the graph as a whole. Note, that a participant's action is now either a reference to another participant or a task-specific action, i.e. the number of actions has increased. This is still consistent with Definition 1 because a participant is completely executed (without transfer of execution to different participants) before action selection can take place. In effect, by adopting a graph, hierarchical relationships now exist so that a participant can defer task-specific action selection to a more specialist participant. We identify these approaches as 'rule based' in Fig. 8.6.

More recently, graphs have been evolved for which each node represents a team and each participant an arc. Given a start or root node, a subset of the teams in the graph is visited, depending on the state of the environment. The process of visiting nodes (teams) continues until an arc is selected that ends in a task-specific action as opposed to another team. In the following, we review attempts to evolve variable-sized ensembles (including trees of teams) using this process (Sect. 8.5.1) and then generalize this to the case of graphs (Sect. 8.5.2).

8.5.1 Variable Size Ensembles Through Symbiosis

Symbiotic models of cooperative coevolution provide a generic approach for discovering the composition of variable-size ensembles / multi-agent teams using only two populations [84]. Thus, unlike the Potter–De Jong formulation of cooperative coevolution (Sect. 8.4), the number of populations is independent of the number of participants appearing in the team. Specifically, there is a team (host) population and a participant/ agent (symbiont) population, Fig. 8.7. The team population attempts to discover useful team compositions whereas the agent population provides the pool of participants that may appear within a team. Participants may appear in multiple teams, and the team composition has to be unique. An early example of symbiosis was used to evolve fixed topology neural networks, i.e equivalent to a fixed size team [142].

In order to extend the two population model into a variable length representation (thus hybrid homogeneous/ heterogeneous compositions), agents need to distinguish

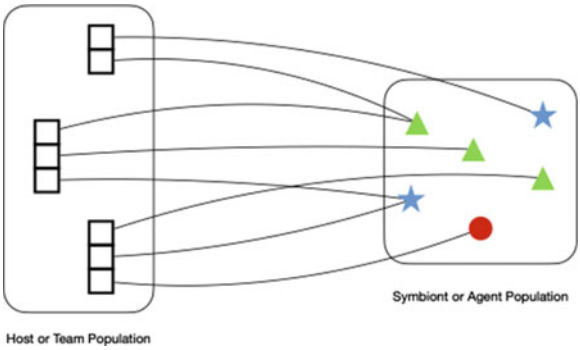


Fig. 8.7 Symbiotic cooperative coevolution with bid based agents [126]. Participants from the Team population (LHS) are defined as pointers to participants of the Agent population (RHS). For illustration the Agent population is considered to consist of three types of action (e.g. class 1, 2, 3), represented by the star, triangle and circle. Valid teams should consist of Agents representing at least two different types of action. The same Agent can appear in multiple Teams, but the team complement should be unique

between context and action [125]. Thus, agent execution[9] is used to identify the bid (or confidence) given the environment's state. All agents from the same team provide a bid; however, only the agent with the highest bid 'wins' the right to suggest its action [126, 127, 223]. This means that multiple agents might appear in the same team with the same action, but with different contexts, adding another degree of flexibility to the process of divide-and-conquer.[10] Moreover, teams need not start with the full complement of agent types, but instead incrementally develop the relevant type complexity.

In the simplest case the action, a, is just a discrete scalar value.[11] Agent actions are chosen from the set of task-specific actions $a \in \mathcal{A}$, e.g. class labels. We now have an agent representation that can evolve teams consisting of any number of team participant types and different sizes. Moreover, the parent pool is identified at the level of teams. Any team participants not associated with a surviving team are deleted, i.e. task specific fitness need only be defined at the level of the team population. Hitchhiking is still an issue but can be addressed by periodically dropping team participants that never win a round of bidding. The resulting symbiotic model of coevolution was demonstrated to be superior to evolution without ensembles [127] and competitive with learning classifiers and support vector machines under multi-class classification tasks [126, 223]. Further developments included scaling to high-dimensional ($\geq 5,000$) classification tasks [56, 127] and operation under non-stationary streaming data classification tasks (Sect. 8.6).

[9] The executable component of an agent could be a program or a neural network.

[10] Agent context divides the input space into a region and associates its region with an action. As multiple programs appear in the same ensemble, multiple regions–actions appear.

[11] Support for real-valued actions introduces an action program at each agent [23, 106].

The approach has also been extended to produce hierarchical teams for reinforcement evolutionary ensemble learning (**rEEL**) [55, 105]. This is distinct from maEEL as rEEL solutions describe a single agent policy but explicitly decompose the task/representation. One implication of this is that strategies for solving one aspect of a task can be reused for solving other aspects of a task [103]. The resulting tree structure represents teams as tree nodes and agents as arcs. Leaves represent atomic actions. The tree is constructed bottom-up (but evaluated top-down from a single root team), with successive layers describing their actions in terms of pointers to previously evolved teams [55, 105].

In order to ensure that different teams represent different strategies, then diversity maintenance (during evolution) represents a re-occurring theme (Sect. 8.4.1). In particular, different tasks could be interleaved with the development of the hierarchical team, thus a natural approach for task transfer [101, 103]. Alternatively, competitive coevolution has been used to develop an 'arms race' between tasks and solution strategies resulting in the organization of thousands of team participants for optimally solving (tens of millions of) Rubik's Cube configurations [186, 190]. As an additional benefit, unlike monolithic solutions (a single large agent), only one team per tree level is evaluated to determine the ultimate action, making for extremely efficient solutions when compared to neural networks [103, 187].

8.5.2 Tangled Program Graphs

The symbiotic framework was also generalized to organizing teams into graphs of teams, leading to results that are competitive with deep learning solutions on (visual) reinforcement learning problems [102, 187]. Indeed, the resulting 'tangled program graphs' could learn policies for playing multiple game titles simultaneously under the ALE benchmark, i.e multitask learning [104]. The graph representation gives direct insights into the nature of the decomposition of agents to decision-making under different game titles, i.e. interpretable machine learning. Later developments demonstrated that the approach also scales to multiple types of partially observable task[12] such as Dota 2 [188] and ViZDoom navigation [108]. Support for real-valued actions under tangled program graphs enabled problems in recursive forecasting [108], multitask control [109], and biped locomotion [14] to be addressed.

One of the unique benefits of the graph-based ensemble is that they are exceptionally efficient to train and deploy. The composition of agents, teams, and graphs is incremental and entirely emergent. This results in solutions whose complexity reflects the properties of the task, not the initial dimensionality of the state space or a priori assumptions about suitable solution topologies. Thus, under visual reinforcement tasks, less than 20% of the pixels are used to reach a decision [102, 104, 187].[13]

[12] Implies that the agents can only solve a task by forming decisions from the internal state (memory) as well as the external state as provided by the environment.

[13] Decreases to $< 5\%$ as the dimension of the visual state space increases [107].

The complexity of the entire solution is typically three or more orders of magnitude lower than deep learning approaches to the same task [102, 104, 187]. Specifically, in order to make a decision, only a fraction of the graph is evaluated, which this changing as a function of the state. Insights are then possible about the functionality of participants in the evolved team [107]. In addition, this has led to the use of graph ensembles in ultra-low power signal processing applications such as real-time intrusion detection on IoT devices [196]. Indeed, solutions to the ALE visual reinforcement learning benchmark [137] have been demonstrated using nothing other than Raspberry PI embedded controllers [48].

8.6 Applications and Future Research

Table 8.1 provides a review of specific application developments that have benefited from adopting an evolutionary ensemble learning approach. Thus, aside from the application of evolutionary ensemble methods to a wide range of regression and classification problems (summarized in Sect. 8.2), we note that the underlying requirements for constructing ensembles are also the requirements for feature construction/engineering using wrapper or filter methods [77]. Specifically, feature construction requires that a diverse set of features be engineered to improve the performance of a regression or classification task. Indeed, many evolutionary approaches to feature engineering assume a multi-tree representation, e.g. [7, 21, 148, 206]. Thus, the number of ensemble participants (n) represents the number of features constructed [7, 116, 148, 183]. More recently, multiple features (participants) have been engineered per class (e.g. [56, 127, 206]) or features (participants) are evolved that are capable of transferring between different environments [8]. Multidimensional genetic programming approaches the (feature construction) task from a different perspective by attempting to discover a low-dimensional space appropriate for describing all aspects of the task [39, 139]. In general, what is particularly impressive with ensemble solutions to the feature construction problem is the capacity to discover very accurate low-dimensional feature spaces from applications described in terms of higher dimensions [56, 148, 171] or from low cardinality datasets [9].

Future work might consider the use of ensemble diversity measures originally developed from the perspective of feature construction. For example, limitations may appear when relying on pairwise (feature) diversity measures [90] or attribute frequency importance [49], whereas a permutation-based sensitivity analysis can avoid the linear correlation constraint (e.g. [12, 49, 93]). Future work might further consider the utility of permutation schemes for interpretable solutions [57].

In general, scalability represents an underlying theme for sEEL. One approach might be to make use of the increasing availability of parallel computing platforms, such as cloud [18, 68, 212] or GPU [17]. Alternatively, methods for algorithmically reducing the number of evaluations might be adopted, such as surrogate models [36] or active learning [185]. Both of the latter have been benchmarked on computer vision benchmarks such as CIFAR resulting in much simpler solutions than

Table 8.1 Summary of EEL research with specific application contexts

Application area	Publication
Control systems	
Direct control	[55, 76, 161]
Dynamical systems modelling	[1, 47]
Path planning/obstacle avoidance	[54, 105, 144, 145]
Robot locomotion	[14]
Data analysis	
Boolean expression learning	[13, 91, 92]
Cancer prediction	[7, 11, 85, 113, 222]
Instance selection	[71]
Interpretable solutions	[35]
Multi-label classification	[146]
Multi-objective	[21, 40, 42, 63, 123, 129, 139, 176]
Outlier reduction	[50]
Software fault utilization	[191]
Feature Construction	
Application specific	[7, 21, 85]
High-dimensional (input)	[56, 127, 148]
Image data	[9, 27, 28, 36, 37, 89, 171, 180, 203]
Inter-task feature transfer	[8, 28, 149]
Low cardinality (training)	[9]
Multi-feature construction	[56, 89, 206]
Multi-task or transfer learning	
Supervised	[8, 28, 149, 178, 180]
Reinforcement learning agents (memory)	[106, 108, 109, 188, 189]
Reinforcement learning agents (reactive)	[22, 48, 101–104, 112, 186, 187, 190]
Multi-agent Reinforcement learning	
Air traffic control	[5]
Five aside soccer	[75]
Keepaway soccer	[101, 152, 213, 219]
Half-field offence	[101, 103]
Multi-agent communication	[52, 140, 225]
Multi-rover	[6, 51, 74, 75, 110, 175, 224]
Preditor–prey	[74, 225]
RoboCup	[15, 133]
Sensor networks	[4]
Scalable training	
Active learning	[69, 72, 126, 127, 139, 185, 192]
Algorithmic efficiency	[214, 228]
Cloud or cluster computing	[18, 68, 212]
GPU platform	[17]
Surrogate fitness	[36]

(continued)

Table 8.1 (continued)

Application area	Publication
Scheduling	
Capacitated arc routing	[216]
Dispatching rules	[58–60, 80, 163]
Trip planning	[98]
Streaming	
Benchmarks	[19, 64, 65, 67, 83, 210]
Churn detection	[211]
Intrusion detection	[66, 111, 196]
Trading agents	[130, 136]

currently available using deep learning. In addition, organizing ensemble agents as a stack/cascade was shown to scale sEEL to data cardinalities in the hundreds of thousands in less than 30 s on a regular CPU [228]. Future work might continue to investigate how different ways of composing ensembles trades off accuracy versus training efficiency versus interpretability [35].

A related application of sEEL is that of streaming data forecasting and classification [83]. Some properties that make the streaming data environment challenging yet appropriate for sEEL might include.

- Non-stationary nature of the underlying task (drift and/or shift) which might imply that mechanisms need to be identified for detecting the onset of change and reacting appropriately [64, 65, 67]. Ensembles are capable of reacting to changes more effectively than non-ensemble approaches because the implicit modularity enables specific participants to be retired/replaced as their performance degrades. This then leads to solutions that are more adaptable than without the use of ensembles [210].
- Anytime nature of deployment implies that in time series classification a champion classifier has to be available for labelling the *next* exemplar before any model has encountered it. This means that the champion classifier might vary over the course of time.
- Imbalanced or limited availability of label information. Given that streaming data is typically experienced on a continuous basis (there is no 'end' to network or stock market data), models are constructed from the content of a sliding window, i.e. a finite number of exemplars. This can lead to different strategies being adopted for retaining data beyond the most recent window content, e.g. data subset archiving and ensemble archiving [19, 111].

To date, streaming ensemble methods have been applied to applications in trading agents [130, 136], intrusion detection [66, 111], electricity utilization [131], satellite data [64], and churn detection [211]. Specifically, Mabu et al. assume a graph representation that explicitly captures dynamics of stock trading, whereas Loginov and Heywood coevolve an ensemble of technical indicators and decision trees for cur-

rency trading and utility forecasting. Both Folino et al. and Khanchi et al. emphasize the ability of ensembles to dynamically react to changes to the underlying properties of the streaming data. A related topic is that of dynamical systems identification in which each (independent) variable has a participant evolved and a suitable aggregation function applied [1]. Particular challenges in this setting might include evolving explainable solutions. Other tasks with dynamic properties that have deployed evolutionary ensemble approaches include different formulations of scheduling tasks that are addressed through the evolution of dispatching rules, e.g. [58–60, 80, 163].

Task transfer and multi-task learning also represent areas in which rEEL can potentially produce advances to the state-of-the-art. Task transfer is typically assumed when the ultimate objective is too complex to solve 'tabula rasa' [219]. Instead, solutions are first evolved to solve simpler source tasks before the ultimate target task is encountered [202]. Likewise, multitask learning requires that solutions to multiple tasks are discovered such that a single champion for all tasks is discovered. This can be particularly difficult as, aside from the difficulty of solving each task, the agent has to establish what environment it is in. Current results indicate that EEL approaches are well placed to incrementally absorb multiple source tasks [8, 101, 103, 149, 186–188] as well as solve multiple tasks simultaneously [28, 98, 104, 108, 109].

Future challenges might include extending these results to environments requiring lifelong/continuous learning (e.g. [179]) and addressing pathologies such as catastrophic forgetting (e.g. [114]) or returning solutions that are interpretable [57, 174]. Given the explicit use of structured representations in EEL, interpretable solutions might represent a potentially significant new development for EEL. Moreover, some of the opaque properties of individual participants might be amenable to simplification using techniques developed for interpretable AI (e.g. model debugging using adversarial learning or perturbation-based analysis [57]). Indeed, there is already a rich history of employing competitive coevolution (the EC approach to adversarial learning) to develop more robust solutions to computer security applications [157].

Multi-agent systems will continue to develop, particularly with respect to evolutionary robotics [53]. One avenue that is beginning to see some results is with regards to the evolution of communication [140] or stigmergy [225] in multi-agent systems. In particular, Mingo and Aler demonstrate that agents can evolve spatial languages with specific syntactical properties using the evolution of grammatical evolution [155]. As the number of agents and objects increases, then the sophistication of the evolved language also increases [140]. Developments of this nature may lead to agents teaching agents [178] and/ or forms of problem-solving that uniquely reflect the mixed ability of the agents to perform different tasks [10].

In addition, multi-agent approaches have appeared in gaming applications in which group behaviours might be desirable. The RoboCup competition represented an early example [15, 133], with more recent works using specific aspects of the full-team competition as smaller scale benchmarks, e.g. keepaway [101, 152, 213, 219], half field offence [101, 103] or five-a-side soccer [75]. First-person video games have also been used to demonstrate the development of squad behaviours using EEL [197] and confirmed that teams increasingly make use of communication as the amount of

visual information decreases [52]. Likewise, partially observable environments have also been used to demonstrate: (1) navigation behaviours under visual reinforcement problems [108, 188, 189] and (2) time series prediction [106, 108] and (3) agents able to solve multiple control problems simultaneously [109]. The resulting EEL graph structures demonstrate an emergent division of duties between, for example, participants that write to memory (a specialist) and those that read (everyone) or the types of tasks addressed by different parts of the graph-based ensemble.

8.7 Discussion

EEL in general has a long and rich history in which the desire to scale evolutionary computation to increasingly more demanding tasks represents an underlying motivation. To do so, the divide-and-conquer approach to problem-solving is assumed as a general principle. However, in doing so, several pathologies potentially appear/need recognition of which level of selection and diversity maintenance represent reoccurring themes. The level of selection reflects the fact that a solution is composed of multiple participants, whereas the performance function might only operate at one level. Moreover, gene linkage can appear between participants and diversity can appear at multiple 'levels', making credit assignment difficult to measure. In addition, EEL as applied to multi-agent systems cannot just assume that teams will be heterogeneous. Instead, specific combinations of different types of agents might be the norm.

Historically, supervised learning applications of EEL have assumed a fixed-sized ensemble defined by a multi-tree representation (Sect. 8.2). This means that the participants are always heterogeneous and the unit of selection is that of the team. However, as demonstrated by the OET algorithm (Sect. 8.3), this might represent a sub-optimal model of selection. Conversely, multi-agent tasks often assume cooperative coevolution as the starting point for defining a team of agents. The cooperative coevolutionary model not only provides a wider opportunity for developing mechanisms for answering the level of selection question but also potentially introduces multiple pathologies (Sect. 8.4). Attempting to develop variable-sized ensembles means that a participant has to distinguish between learning context (decomposing the state/input space into regions) versus suggesting an action (Sect. 8.5). Some of the benefits of adopting a variable-sized team are that the evolved ensemble imparts additional knowledge about what was learnt. However, pathologies such as hitchhiking might result in bloated ensembles unless mitigation strategies are taken.

The future of EEL will likely continue to grow with the development of applications on the one hand and challenges to machine learning as a whole on the other. EEL is central to a body of work on feature construction that is now leading to the adoption of task transfer techniques, e.g. high-dimensional tasks with missing information and/or low cardinality. Likewise, EEL has repeatedly been successfully applied to streaming data in general, but the number of new streaming data applications continues to grow (e.g. IoT, social media, e-commerce). Streaming data applications

also point to the concept of lifelong (continuous) learning in which case there is potentially no end to the learning process.

EEL as formulated to address multi-agent or reinforcement learning problems is able to answer questions about hybrid homogeneous–heterogeneous team composition as well as variable-size ensembles. Incorporating hierarchical relationships into EEL means that participants who systematically mispredict can defer their decision to another (specialist) team that concentrates on resolving this ambiguity. Approaches for discovering graph ensembles provide a further opportunity for establishing structures that might be appropriate for continuous learning and interpretable solutions. A wide range of empirical results has already established that participants are significantly less complex than monolithic solutions (e.g. when using SVM or deep learning). Moreover, the appropriate selection of the ensemble aggregation operation (e.g. winner-tasks-all or the additive operator) provides explicit support for interpretable solutions [174]. This in combination with parsimonious participants may lead to truly scalable ensembles that support low-dimensional saliency, i.e. do not rely on post hoc 'explanations'. In short, there is still 'plenty of room' in the divide-and-conquer approach to evolutionary machine learning.

Acknowledgements The author gratefully acknowledges support from the NSERC Discovery program (Canada).

References

1. Abdelbari, H., Shafi, K.: A genetic programming ensemble method for learning dynamical system models. In: Proceedings of the International Conference on Computer Modeling and Simulation, pp. 47–51. ACM (2017)
2. Agapitos, A., Loughran, R., Nicolau, M., Lucas, S.M., O'Neill, M., Brabazon, A.: A survey of statistical machine learning elements in genetic programming. IEEE Trans. Evol. Comput. **23**(6), 1029–1048 (2019)
3. Agapitos, A., O'Neill, M., Brabazon, A.: Ensemble bayesian model averaging in genetic programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2451–2458. IEEE (2014)
4. Agogino, A.K., Parker, C.H., Tumer, K.: Evolving distributed resource sharing for cubesat constellations. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1015–1022. ACM (2012)
5. Agogino, A.K., Tumer, K.: Evolving distributed agents for managing air traffic. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1888–1895. ACM (2007)
6. Agogino, A.K., Tumer, K.: Efficient evaluation functions for evolving coordination. Evol. Comput. **16**(2), 257–288 (2008)
7. Ain, Q.U., Al-Sahaf, H., Xue, B., Zhang, M.: A genetic programming approach to feature construction for ensemble learning in skin cancer detection. In: C.A.C. Coello (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1186–1194. ACM (2020)
8. Al-Helali, B., Chen, Q., Xue, B., Zhang, M.: Multitree genetic programming with new operators for transfer learning in symbolic regression with incomplete data. IEEE Trans. Evol. Comput. **25**(6), 1049–1063 (2021)
9. Al-Sahaf, H., Al-Sahaf, A., Xue, B., Zhang, M.: Automatically evolving texture image descriptors using the multitree representation in genetic programming using few instances. Evol. Comput. **29**(3), 331–366 (2021)

10. Albrecht, S.V., Liemhetcharat, S., Stone, P.: Special issue on multi-agent interaction without prior coordination: guest editorial. Autonomous Agents and Multi Agent Systems **31**(4), 765–766 (2017)
11. Ali, S., Majid, A.: Can-evo-ens: Classifier stacking based evolutionary ensemble system for prediction of human breast cancer using amino acid sequences. J. Biomed. Inform. **54**, 256–269 (2015)
12. Altmann, A., Toloşi, L., Sander, O., Lengauer, T.: Permutation importance: a corrected feature importance measure. Bioinformatics **26**(10), 1340–1347 (2010)
13. Alvarez, I.M., Nguyen, T.B., Browne, W.N., Zhang, M.: A layered learning approach to scaling in learning classifier systems for boolean problems. CoRR **abs/2006.01415** (2020)
14. Amaral, R., Ianta, A., Bayer, C., Smith, R.J., Heywood, M.I.: Benchmarking genetic programming in a multi-action reinforcement learning locomotion task. In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM (2022)
15. Andre, D., Teller, A.: Evolving team darwin united. In: RoboCup-98: Robot Soccer World Cup II, *LNCS*, vol. 1604, pp. 346–351. Springer (1998)
16. Angeline, P.J., Saunders, G.M., Pollack, J.B.: An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans. Neural Networks **5**(1), 54–65 (1994)
17. Arnaldo, I., Veeramachaneni, K., O'Reilly, U.: Flash: A GP-GPU ensemble learning system for handling large datasets. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 8599, pp. 13–24 (2014)
18. Arnaldo, I., Veeramachaneni, K., Song, A., O'Reilly, U.: Bring your own learner: A cloud-based, data-parallel commons for machine learning. IEEE Comput. Intell. Mag. **10**(1), 20–32 (2015)
19. Atwater, A., Heywood, M.I.: Benchmarking Pareto archiving heuristics in the presence of concept drift: diversity versus age. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 885–892. ACM (2013)
20. Axelrod, R.: Evolution of co-operation. Basic Books (1984)
21. Badran, K.M.S., Rockett, P.I.: Multi-class pattern classification using single, multi-dimensional feature-space feature extraction evolved by multi-objective genetic programming and its application to network intrusion detection. Genet. Program Evolvable Mach. **13**(1), 33–63 (2012)
22. Bahçeci, E., Miikkulainen, R.: Transfer of evolved pattern-based heuristics in games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, pp. 220–227. IEEE (2008)
23. Bayer, C., Amaral, R., Smith, R.J., Ianta, A., Heywood, M.I.: Finding simple solutions to multi-task visual reinforcement learning problems with tangled program graphs. In: Genetic Programming Theory and Practice, vol. XVIII, pp. 1–19 (2022)
24. Belkin, M., Hsu, D., Ma, S., Mandal, S.: Reconciling modern machine learning and the bias-variance trade-off. Proceedings of the National Academy of Science **116**(32), 15849–15854 (2019)
25. Bhowan, U., Johnston, M., Zhang, M., Yao, X.: Evolving diverse ensembles using genetic programming for classification with unbalanced data. IEEE Trans. Evol. Comput. **17**(3), 368–386 (2013)
26. Bhowan, U., Johnston, M., Zhang, M., Yao, X.: Reusing genetic programming for ensemble selection in classification of unbalanced data. IEEE Trans. Evol. Comput. **18**(6), 893–908 (2014)
27. Bi, Y., Xue, B., Zhang, M.: A divide-and-conquer genetic programming algorithm with ensembles for image classification. IEEE Trans. Evol. Comput. **25**(6), 1148–1162 (2021)
28. Bi, Y., Xue, B., Zhang, M.: Learning and sharing: A multitask genetic programming approach to image feature learning. IEEE Trans. Evol. Comput. **26**(2), 218–232 (2022)
29. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
30. Bongard, J.C.: The legion system: A novel approach to evolving heterogeneity for collective problem solving. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 1802, pp. 16–28. Springer (2000)

31. Brameier, M., Banzhaf, W.: Evolving teams of predictors with linear genetic programming. Genet. Program Evolvable Mach. **2**(4), 381–407 (2001)
32. Brave, S.: The evolution of memory and mental models using genetic programming. In: Proceedings of the Annual Conference on Genetic Programming. Morgan Kaufmann (1996)
33. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
34. Breiman, L.: Arcing classifier. Annuals of. Statistics **26**(3), 801–849 (1998)
35. Cagnini, H.E.L., Freitas, A.A., Barros, R.C.: An evolutionary algorithm for learning interpretable ensembles of classifiers. In: Proceedings of the Brazilian Conference on Intelligent Systems, *Lecture Notes in Computer Science*, vol. 12319, pp. 18–33. Springer (2020)
36. Cardoso, R.P., Hart, E., Kurka, D.B., Pitt, J.: Augmenting novelty search with a surrogate model to engineer meta-diversity in ensembles of classifiers. In: Proceedings of the European Conference on Applications of Evolutionary Computation, *LNCS*, vol. 13224, pp. 418–434 (2022)
37. Cardoso, R.P., Hart, E., Kurka, D.B., Pitt, J.V.: Using novelty search to explicitly create diversity in ensembles of classifiers. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 849–857. ACM (2021)
38. Castelli, M., Gonçalves, I., Manzoni, L., Vanneschi, L.: Pruning techniques for mixed ensembles of genetic programming models. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 10781, pp. 52–67 (2018)
39. Cava, W.G.L., Moore, J.H.: Learning feature spaces for regression with genetic programming. Genet. Program Evolvable Mach. **21**(3), 433–467 (2020)
40. Chandra, A., Chen, H., Yao, X.: Trade-off between diversity and accuracy in ensemble generation. In: Y. Jin (ed.) Multi-Objective Machine Learning, *Studies in Computational Intelligence*, vol. 16, pp. 429–464. Springer (2006)
41. Chandra, A., Yao, X.: Evolving hybrid ensembles of learning machines for better generalization. Neurocomputing **69**, 686–700 (2006)
42. Chen, H., Yao, X.: Multiobjective neural network ensembles based on regularized negative correlation learning. IEEE Trans. Knowl. Data Eng. **22**(12), 1738–1751 (2010)
43. Coello, C.A.C.: Evolutionary multi-objective optimization: a historical view of the field. IEEE Comput. Intell. Mag. **1**(1), 28–36 (2006)
44. Colby, M.K., Tumer, K.: Shaping fitness functions for coevolving cooperative multiagent systems. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 425–432. IFAAMAS (2012)
45. D'Ambrosio, D.B., Lehman, J., Risi, S., Stanley, K.O.: Evolving policy geometry for scalable multiagent learning. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 731–738. IFAAMAS (2010)
46. D'Ambrosio, D.B., Stanley, K.O.: Scalable multiagent learning through indirect encoding of policy geometry. Evol. Intel. **6**(1), 1–26 (2013)
47. Defoin-Platel, M., Chami, M., Clergue, M., Collard, P.: Teams of genetic predictors for inverse problem solving. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 3447, pp. 341–350 (2005)
48. Desnos, K., Sourbier, N., Raumer, P., Gesny, O., Pelcat, M.: Gegelati: Lightweight artificial intelligence through generic and evolvable tangled program graphs. In: Workshop on Design and Architectures for Signal and Image Processing, pp. 35–43. ACM (2021)
49. Dick, G.: Sensitivity-like analysis for feature selection in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 401–408. ACM (2017)
50. Dick, G., Owen, C.A., Whigham, P.A.: Evolving bagging ensembles using a spatially-structured niching method. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 418–425. ACM (2018)
51. Dixit, G., Zerbel, N., Tumer, K.: Dirichlet-multinomial counterfactual rewards for heterogeneous multiagent systems. In: IEEE International Symposium on Multi-Robot and Multi-Agent Systems, pp. 209–215. IEEE (2019)
52. Doherty, D., O'Riordan, C.: Effects of shared perception on the evolution of squad behaviors. IEEE Transactions on Computational Intelligence and AI in Games **1**(1), 50–62 (2009)

53. Doncieux, S., Bredèche, N., Mouret, J., Eiben, A.E.: Evolutionary robotics: What, why, and where to. Frontiers Robotics AI **2**, 4 (2015)
54. Doncieux, S., Mouret, J.: Behavioral diversity with multiple behavioral distances. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1427–1434. IEEE (2013)
55. Doucette, J.A., Lichodzijewski, P., Heywood, M.I.: Hierarchical task decomposition through symbiosis in reinforcement learning. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 97–104. ACM (2012)
56. Doucette, J.A., McIntyre, A.R., Lichodzijewski, P., Heywood, M.I.: Symbiotic coevolutionary genetic programming: a benchmarking study under large attribute spaces. Genet. Program Evolvable Mach. **13**(1), 71–101 (2012)
57. Du, M., Liu, N., Hu, X.: Techniques for interpretable machine learning. Commun. ACM **63**(1), 68–77 (2020)
58. Durasevic, M., Jakobovic, D.: Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. Genet. Program Evolvable Mach. **19**(1–2), 53–92 (2018)
59. Durasevic, M., Jakobovic, D.: Creating dispatching rules by simple ensemble combination. J. Heuristics **25**(6), 959–1013 (2019)
60. Durasevic, M., Planinic, L., Gala, F.J.G., Jakobovic, D.: Novel ensemble collaboration method for dynamic scheduling problems. CoRR **abs/2203.14290** (2022)
61. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture. In: Advances in Neural Information Processing Systems, vol. 2, pp. 524–532. Morgan Kaufmann (1989)
62. Feldt, R.: Generating diverse software versions with genetic programming: and experimental study. IEE Proceedings-Software **145**(6), 228–236 (1998)
63. Fletcher, S., Verma, B.K., Zhang, M.: A non-specialized ensemble classifier using multi-objective optimization. Neurocomputing **409**, 93–102 (2020)
64. Folino, G., Guarascio, M., Papuzzo, G.: Exploiting fractal dimension and a distributed evolutionary approach to classify data streams with concept drifts. Appl. Soft Comput. **75**, 284–297 (2019)
65. Folino, G., Papuzzo, G.: Handling different categories of concept drifts in data streams using distributed GP. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 6021, pp. 74–85. Springer (2010)
66. Folino, G., Pisani, F.S., Pontieri, L.: A gp-based ensemble classification framework for time-changing streams of intrusion detection data. Soft. Comput. **24**(23), 17541–17560 (2020)
67. Folino, G., Pizzuti, C., Spezzano, G.: Mining distributed evolving data streams using fractal GP ensembles. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 4445, pp. 160–169. Springer (2007)
68. Folino, G., Pizzuti, C., Spezzano, G.: Training distributed GP ensemble with a selective algorithm based on clustering and pruning for pattern classification. IEEE Trans. Evol. Comput. **12**(4), 458–468 (2008)
69. Gagné, C., Sebag, M., Schoenauer, M., Tomassini, M.: Ensemble learning for free with evolutionary algorithms? In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1782–1789. ACM (2007)
70. Gama, J., Brazdil, P.: Cascade generalization. Mach. Learn. **41**(3), 315–343 (2000)
71. García-Pedrajas, N., del Castillo, J.A.R., Ortiz-Boyer, D.: A cooperative coevolutionary algorithm for instance selection for instance-based learning. Mach. Learn. **78**(3), 381–420 (2010)
72. Gathercole, C., Ross, P.: Dynamic training subset selection for supervised learning in genetic programming. In: Proceedings of the Parallel Problem Solving from Nature Conference, *LNCS*, vol. 866, pp. 312–321. Springer (1994)
73. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Comput. **4**(1), 1–58 (1992)
74. Gomes, J.C., Mariano, P., Christensen, A.L.: Novelty-driven cooperative coevolution. Evol. Comput. **25**(2), 275–307 (2017)
75. Gomes, J.C., Mariano, P., Christensen, A.L.: Dynamic team heterogeneity in cooperative coevolutionary algorithms. IEEE Trans. Evol. Comput. **22**(6), 934–948 (2018)

76. Gomez, F.J., Schmidhuber, J., Miikkulainen, R.: Accelerated neural evolution through cooperatively coevolved synapses. J. Mach. Learn. Res. **9**, 937–965 (2008)
77. Guyon, I., Nikravesh, M., Gunn, S.R., Zadeh, L.A. (eds.): Feature Extraction - Foundations and Applications, *Studies in Fuzziness and Soft Computing*, vol. 207. Springer (2006)
78. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. **12**(10), 993–1001 (1990)
79. Hara, A., Nagao, T.: Emergence of the cooperative behavior using adg; automatically defined groups. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1039–1046. Morgan Kaufmann (1999)
80. Hart, E., Sim, K.: On constructing ensembles for combinatorial optimisation. Evolutionary Computation **26**(1) (2018)
81. Haynes, T., Sen, S., Schoenefeld, D.A., Wainwright, R.L.: Evolving a team. In: AAAI Fall Symposium on Genetic Programming, pp. 23–30. AAAI Press (1995)
82. Helmuth, T., Spector, L., Matheson, J.: Solving uncompromising problems with lexicase selection. IEEE Trans. Evol. Comput. **19**(5), 630–643 (2015)
83. Heywood, M.I.: Evolutionary model building under streaming data for classification tasks: opportunities and challenges. Genet. Program Evolvable Mach. **16**(3), 283–326 (2015)
84. Heywood, M.I., Lichodzijewski, P.: Symbiogenesis as a mechanism for building complex adaptive systems: A review. In: Proceedings of the Applications of Evolutionary Computation Conference–Part I, *LNCS*, vol. 6024, pp. 51–60. Springer (2010)
85. Hong, J., Cho, S.: The classification of cancer based on DNA microarray data that uses diverse ensemble genetic programming. Artif. Intell. Med. **36**(1), 43–58 (2006)
86. Howley, E., O'Riordan, C.: The emergence of cooperation among agents using simple fixed bias tagging. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1011–1016. IEEE (2005)
87. Iba, H.: Emergent cooperation for multiple agents using genetic programming. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, *LNCS*, vol. 1141, pp. 32–41. Springer (1996)
88. Iba, H.: Bagging, boosting, and bloating in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1053–1060. Morgan Kaufmann (1999)
89. Ibarra-Vázquez, G., Olague, G., Chan-Ley, M., Puente, C., Soubervielle-Montalvo, C.: Brain programming is immune to adversarial attacks: Towards accurate and robust image classification using symbolic learning. Swarm Evol. Comput. **71**, 101059 (2022)
90. Imamura, K., Soule, T., Heckendorn, R.B., Foster, J.A.: Behavioral diversity and a probabilistically optimal GP ensemble. Genet. Program Evolvable Mach. **4**(3), 235–253 (2003)
91. Iqbal, M., Browne, W.N., Zhang, M.: Reusing building blocks of extracted knowledge to solve complex, large-scale boolean problems. IEEE Trans. Evol. Comput. **18**(4), 465–480 (2014)
92. Iqbal, M., Browne, W.N., Zhang, M.: Extending xcs with cyclic graphs for scalability on complex boolean problems. Evol. Comput. **25**(2), 173–204 (2017)
93. Ivert, A., de Castro Aranha, C., Iba, H.: Feature selection and classification using ensembles of genetic programs and within-class and between-class permutations. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1121–1128. IEEE (2015)
94. Johansson, U., Löfström, T., König, R., Niklasson, L.: Building neural network ensembles using genetic programming. In: Proceedings of the International Joint Conference on Neural Networks, pp. 1260–1265. IEEE (2006)
95. Johansson, U., Sönströd, C., Löfström, T., König, R.: Using genetic programming to obtain implicit diversity. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2454–2459. IEEE (2009)
96. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Comput. **6**(2), 181–214 (1994)
97. Kantschik, W., Banzhaf, W.: Linear-graph GP - A new GP structure. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 2278, pp. 83–92. Springer (2002)

98. Karunakaran, D., Mei, Y., Zhang, M.: Multitasking genetic programming for stochastic team orienteering problem with time windows. In: IEEE Symposium Series on Computational Intelligence, pp. 1598–1605. IEEE (2019)
99. Keijzer, M., Babovic, V.: Genetic programming, ensemble methods and the bias/variance tradeoff – introductory investigations. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 1802, pp. 76–90 (2000)
100. Keijzer, M., Babovic, V.: Declarative and preferential bias in gp-based scientific discovery. Genet. Program Evolvable Mach. **3**(1), 41–79 (2002)
101. Kelly, S., Heywood, M.I.: Knowledge transfer from keepaway soccer to half-field offense through program symbiosis: Building simple programs for a complex task. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1143–1150. ACM (2015)
102. Kelly, S., Heywood, M.I.: Emergent tangled graph representations for atari game playing agents. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 10196, pp. 64–79 (2017)
103. Kelly, S., Heywood, M.I.: Discovering agent behaviors through code reuse: Examples from half-field offense and ms. pac-man. IEEE Transactions on Games **10**(2), 195–208 (2018)
104. Kelly, S., Heywood, M.I.: Emergent solutions to high-dimensional multitask reinforcement learning. Evolutionary Computation **26**(3) (2018)
105. Kelly, S., Lichodzijewski, P., Heywood, M.I.: On run time libraries and hierarchical symbiosis. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012)
106. Kelly, S., Newsted, J., Banzhaf, W., Gondro, C.: A modular memory framework for time series prediction. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 949–957. ACM (2020)
107. Kelly, S., Smith, R.J., Heywood, M.I.: Emergent policy discovery for visual reinforcement learning through tangled program graphs: A tutorial. In: W. Banzhaf, L. Spector, L. Sheneman (eds.) Genetic Programming Theory and Practice XVI, Genetic and Evolutionary Computation, pp. 37–57. Springer (2018)
108. Kelly, S., Smith, R.J., Heywood, M.I., Banzhaf, W.: Emergent tangled program graphs in partially observable recursive forecasting and vizdoom navigation tasks. ACM Transactions on Evolutionary Learning and Optimization **1**(3), 11:1–11:41 (2021)
109. Kelly, S., Voegerl, T., Banzhaf, W., Gondro, C.: Evolving hierarchical memory-prediction machines in multi-task reinforcement learning. Genet. Program Evolvable Mach. **22**(4), 573–605 (2021)
110. Khadka, S., Majumdar, S., Miret, S., McAleer, S., Tumer, K.: Evolutionary reinforcement learning for sample-efficient multiagent coordination. In: Proceedings of the International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 119, pp. 6651–6660. PMLR (2020)
111. Khanchi, S., Vahdat, A., Heywood, M.I., Zincir-Heywood, A.N.: On botnet detection with genetic programming under streaming data label budgets and class imbalance. Swarm Evol. Comput. **39**, 123–140 (2018)
112. Kim, K., Cho, S.: Systematically incorporating domain-specific knowledge into evolutionary speciated checkers players. IEEE Trans. Evol. Comput. **9**(6), 615–627 (2005)
113. Kim, K., Cho, S.: An evolutionary algorithm approach to optimal ensemble classifiers for DNA microarray data analysis. IEEE Trans. Evol. Comput. **12**(3), 377–388 (2008)
114. Kirkpatrick, J., Pascanu, R., Rabinowitz, N.C., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. **114**(13), 3521–3526 (2017)
115. Koza, J.R.: Genetic Programming II: Automatic discovery of reusable programs. MIT Press (1994)
116. Krawiec, K.: Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genet. Program Evolvable Mach. **3**(4), 329–343 (2002)
117. Krawiec, K., Heywood, M.I.: Solving complex problems with coevolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (Tutorial Program), pp. 832–858. ACM (2020)

118. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Advances in Neural Information Processing Systems, vol. 7, pp. 231–238. MIT Press (1994)
119. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley (2004)
120. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach. Learn. **51**(2), 181–207 (2003)
121. Lacy, S.E., Lones, M.A., Smith, S.L.: A comparison of evolved linear and non-linear ensemble vote aggregators. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 758–763. IEEE (2015)
122. Lalejini, A., Ofria, C.: Evolving event-driven programs with signalgp. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1135–1142. ACM (2018)
123. Levesque, J., Durand, A., Gagné, C., Sabourin, R.: Multi-objective evolutionary optimization for generating ensembles of classifiers in the ROC space. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 879–886. ACM (2012)
124. Lichocki, P., Wischmann, S., Keller, L., Floreano, D.: Evolving team compositions by agent swapping. IEEE Trans. Evol. Comput. **17**(2), 282–298 (2013)
125. Lichodzijewski, P., Heywood, M.I.: Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 464–471. ACM (2007)
126. Lichodzijewski, P., Heywood, M.I.: Managing team-based problem solving with symbiotic bid-based genetic programming. In: C. Ryan, M. Keijzer (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, pp. 363–370. ACM (2008)
127. Lichodzijewski, P., Heywood, M.I.: Symbiosis, complexification and simplicity under GP. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 853–860. ACM (2010)
128. Liu, Y., Yao, X., Higuchi, T.: Evolutionary ensembles with negative correlation learning. IEEE Trans. Evol. Comput. **4**(4), 380–387 (2000)
129. Löfström, T., Johansson, U., Boström, H.: Ensemble member selection using multi-objective optimization. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, pp. 245–251. IEEE (2009)
130. Loginov, A., Heywood, M.I.: On the impact of streaming interface heuristics on GP trading agents: an FX benchmarking study. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1341–1348. ACM (2013)
131. Loginov, A., Heywood, M.I., Wilson, G.C.: Benchmarking a coevolutionary streaming classifier under the individual household electric power consumption dataset. In: International Joint Conference on Neural Networks, pp. 2834–2841. IEEE (2016)
132. Lu, Z., Wu, X., Bongard, J.C.: Active learning through adaptive heterogeneous ensembling. IEEE Trans. Knowl. Data Eng. **27**(2), 368–381 (2015)
133. Luke, S., Hohn, C., Farris, J., Jackson, G., Hendler, J.A.: Co-evolving soccer softbot team coordination with genetic programming. In: RoboCup-97: Robot Soccer World Cup I, *LNCS*, vol. 1395, pp. 398–411. Springer (1997)
134. Luke, S., Spector, L.: Evolving teamwork and coordination with genetic programming. In: Proceedings of the Annual Conference on Genetic Programming, pp. 150–156. MIT Press (1996)
135. Mabu, S., Hirasawa, K., Hu, J.: A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning. Evol. Comput. **15**(3), 369–398 (2007)
136. Mabu, S., Hirasawa, K., Obayashi, M., Kuremoto, T.: Enhanced decision making mechanism of rule-based genetic network programming for creating stock trading signals. Expert Syst. Appl. **40**(16), 6311–6320 (2013)
137. Machado, M.C., Bellemare, M.G., Talvitie, E., Veness, J., Hausknecht, M.J., Bowling, M.: Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. Journal of Artificial Intelligence Research **61**, 523–562 (2018)
138. Maynard-Smith, J., Haigh, J.: The hitch-hiking effect of a favourable gene. Genet. Res. **23**(1), 23–35 (1974)

139. McIntyre, A.R., Heywood, M.I.: Classification as clustering: A pareto cooperative-competitive GP approach. Evol. Comput. **19**(1), 137–166 (2011)
140. Mingo, J.M., Aler, R.: Evolution of shared grammars for describing simulated spatial scenes with grammatical evolution. Genetic Programming and Evolvable Machine **19**(1–2), 235–270 (2018)
141. Mitchell, M., Forrest, S., Holland, J.H.: The Royal Road for genetic algorithms: Fitness landscapes and GA performance. In: Proceedings of the European Conference on Artificial Life, pp. 245–254. MIT Press (1992)
142. Moriarty, D.E., Miikkulainen, R.: Efficient reinforcement learning through symbiotic evolution. Mach. Learn. **22**(1–3), 11–32 (1996)
143. Moshaiov, A., Tal, A.: Family bootstrapping: A genetic transfer learning approach for onsetting the evolution for a set of related robotic tasks. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2801–2808. IEEE (2014)
144. Mouret, J., Doncieux, S.: Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1161–1168. IEEE (2009)
145. Mouret, J., Doncieux, S.: Encouraging behavioral diversity in evolutionary robotics: An empirical study. Evol. Comput. **20**(1), 91–133 (2012)
146. Moyano, J.M., Ventura, S.: Auto-adaptive grammar-guided genetic programming algorithm to build ensembles of multi-label classifiers. Inf. Fusion **78**, 1–19 (2022)
147. Muni, D.P., Pal, N.R., Das, J.: A novel approach to design classifiers using genetic programming. IEEE Trans. Evol. Comput. **8**(2), 183–196 (2004)
148. Muni, D.P., Pal, N.R., Das, J.: Genetic programming for simultaneous feature selection and classifier design. IEEE Transactions on Systems, Man, and Cybernetics - Part B **36**(1), 106–117 (2006)
149. Muñoz, L., Trujillo, L., Silva, S.: Transfer learning in constructive induction with genetic programming. Genet. Program Evolvable Mach. **21**(4), 529–569 (2020)
150. Nguyen, M.H., Abbass, H.A., McKay, R.I.: A novel mixture of experts model based on cooperative coevolution. Neurocomputing **70**(1–3), 155–163 (2006)
151. Nguyen, M.H., Abbass, H.A., McKay, R.I.: Analysis of CCME: coevolutionary dynamics, automatic problem decomposition, and regularization. IEEE Transactions on Systems, Man, and Cybernetics-Part C **38**(1), 100–109 (2008)
152. Nitschke, G., Didi, S.: Evolutionary policy transfer and search methods for boosting behavior quality: Robocup keep-away case study. Frontiers Robotics AI **4**, 62 (2017)
153. Nitschke, G.S., Schut, M.C., Eiben, A.E.: Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task. Evol. Intel. **3**(1), 13–29 (2010)
154. Nitschke, G.S., Schut, M.C., Eiben, A.E.: Evolving behavioral specialization in robot teams to solve a collective construction task. Swarm Evol. Comput. **2**, 25–38 (2012)
155. O'Neill, M., Ryan, C.: Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 3003, pp. 138–149 (2004)
156. Opitz, D.W., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research **11**, 169–198 (1999)
157. O'Reilly, U., Toutouh, J., Pertierra, M.A., Sanchez, D.P., Garcia, D., Lugo, A.E., Kelly, J., Hemberg, E.: Adversarial genetic programming for cyber security: a rising application domain where GP matters. Genet. Program Evolvable Mach. **21**(1–2), 219–250 (2020)
158. Owen, C.A., Dick, G., Whigham, P.A.: Characterizing genetic programming error through extended bias and variance decomposition. IEEE Trans. Evol. Comput. **24**(6), 1164–1176 (2020)
159. Panait, L., Luke, S., Wiegand, R.P.: Biasing coevolutionary search for optimal multiagent behaviors. IEEE Trans. Evol. Comput. **10**(6), 629–645 (2006)
160. Panait, L., Sullivan, K., Luke, S.: Lenience towards teammates helps in cooperative mutiagent learning. Tech. Rep. GMU-CS-TR-2013-2, George Mason University (2013)

161. Pardoe, D., Ryoo, M.S., Miikkulainen, R.: Evolving neural network ensembles for control problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1379–1384. ACM (2005)
162. Paris, G., Robilliard, D., Fonlupt, C.: Applying boosting techniques to genetic programming. In: International Conference on Artificial Evolution, *LNCS*, vol. 2310, pp. 267–280 (2001)
163. Park, J., Nguyen, S., Zhang, M., Johnston, M.: Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 9025, pp. 92–104 (2015)
164. Parter, M., Kashtan, N., Alon, U.: Facilitated variation: How evolution learns for past environments to generalize to new environments. PLoS Computational Biology **4**(11), e1000206:1–16 (2008)
165. Portman, B., Heywood, M.I.: On the interaction between lexicase selection, modularity and data subsets. In: Proceedings of the Genetic and Evolutionary Computation Conference (Companion), pp. 586–589. ACM (2022)
166. Potter, M.A., Jong, K.A.D.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evol. Comput. **8**(1), 1–29 (2000)
167. Rebuli, K.B., Vanneschi, L.: Progressive insular cooperative GP. In: T. Hu, N. Lourenço, E. Medvet (eds.) Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 12691, pp. 19–35 (2021)
168. Reynolds, C.W.: An evolved, vision-based behavioral model of coordinated group motion. In: Proceedings of the International Conference on Simulation of Adaptive Behavior, pp. 384–392. MIT Press (1993)
169. Riolo, R.L.: The effects and evolution of tag-mediated selection of partners in populations playing the iterated prisoner's dilemma. In: Proceedings of the International Conference on Genetic Algorithms, pp. 378–385. Morgan Kaufmann (1997)
170. Rodrigues, N.M., Batista, J.E., Silva, S.: Ensemble genetic programming. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 12101, pp. 151–166 (2020)
171. Rodriguez-Coayahuitl, L., Morales-Reyes, A., Escalante, H.J., Coello, C.A.C.: Cooperative co-evolutionary genetic programming for high dimensional problems. In: Proceedings of the Parallel Problem Solving from Nature Conference–Part II, *LNCS*, vol. 12270, pp. 48–62 (2020)
172. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. Evol. Comput. **5**(1), 1–29 (1997)
173. Rubini, J., Heckendorn, R.B., Soule, T.: Evolution of team composition in multi-agent systems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1067–1074. ACM (2009)
174. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
175. Sachdeva, E., Khadka, S., Majumdar, S., Tumar, K.: MAEDyS: multiagent evolution via dynamic skill selection. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 163–171. ACM (2021)
176. dos Santos, E.M., Sabourin, R., Maupin, P.: Overfitting cautious selection of classifier ensembles with genetic algorithms. Information Fusion **10**(2), 150–162 (2009)
177. Schapire, R.E.: The strength of weak learnability. Mach. Learn. **5**, 197–227 (1990)
178. da Silva, F.L., Warnell, G., Costa, A.H.R., Stone, P.: Agents teaching agents: a survey on inter-agent transfer learning. Autonomous Agents and Multi Agent Systems **34**(1), 9 (2020)
179. Silver, D.L., Yang, Q., Li, L.: Lifelong machine learning systems: Beyond learning algorithms. In: Papers from the Spring Symposium, *AAAI Technical Report*, vol. SS-13-05. AAAI (2013)
180. Sipper, M.: Classy ensemble: A novel ensemble algorithm for classification. CoRR **abs/2302.10580** (2023)
181. Sipper, M., Moore, J.H.: Symbolic-regression boosting. Genet. Program Evolvable Mach. **22**(3), 357–381 (2021)
182. Sipper, M., Moore, J.H.: AddGBoost: A gradient boosting-style algorithm based on strong learners. Machine Learning with Applications **7**(100243) (2022)

183. Smith, M.G., Bull, L.: Genetic programming with a genetic algorithm for feature construction and selection. Genetic Programming Evolvable Machines **6**(3), 265–281 (2005)
184. Smith, R.E., Forrest, S., Perelson, A.S.: Searching for diverse, cooperative populations with genetic algorithms. Evol. Comput. **1**(2), 127–149 (1993)
185. Smith, R.J., Amaral, R., Heywood, M.I.: Evolving simple solutions to the CIFAR-10 benchmark using tangled program graphs. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2061–2068. IEEE (2021)
186. Smith, R.J., Heywood, M.I.: Coevolving deep hierarchies of programs to solve complex tasks. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1009–1016. ACM (2017)
187. Smith, R.J., Heywood, M.I.: Scaling tangled program graphs to visual reinforcement learning in vizdoom. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 10781, pp. 135–150 (2018)
188. Smith, R.J., Heywood, M.I.: Evolving Dota 2 shadow fiend bots using genetic programming with external memory. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 179–187. ACM (2019)
189. Smith, R.J., Heywood, M.I.: A model of external memory for navigation in partially observable visual reinforcement learning tasks. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 11451, pp. 162–177 (2019)
190. Smith, R.J., Kelly, S., Heywood, M.I.: Discovering rubik's cube subgroups using coevolutionary GP: A five twist experiment. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 789–796. ACM (2016)
191. Sohn, J., Yoo, S.: Why train-and-select when you can use them all?: ensemble model for fault localisation. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1408–1416. ACM (2019)
192. Song, D., Heywood, M.I., Zincir-Heywood, A.N.: Training genetic programming on half a million patterns: an example from anomaly detection. IEEE Trans. Evol. Comput. **9**(3), 225–239 (2005)
193. Sotto, L.F.D.P., Kaufmann, P., Atkinson, T., Kalkreuth, R., Basgalupp, M.P.: Graph representations in genetic programming. Genet. Program Evolvable Mach. **22**(4), 607–636 (2021)
194. Soule, T.: Voting teams: a cooperative approach to non-typical problems using genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 916–922. Morgan Kaufmann (1999)
195. Soule, T.: Cooperative evolution on the intertwined spirals problem. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 2610, pp. 434–442. Springer (2003)
196. Sourbier, N., Desnos, K., Guyet, T., Majorczyk, F., Gesny, O., Pelcat, M.: SECURE-GEGELATI always-on intrusion detection through GEGELATI lightweight tangled program graphs. Journal of Signal Processing Systems **94**(7), 753–770 (2022)
197. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Real-time neuroevolution in the NERO video game. IEEE Trans. Evol. Comput. **9**(6), 653–668 (2005)
198. Stefano, C.D., Fontanella, F., Folino, G., di Freca, A.S.: A bayesian approach for combining ensembles of GP classifiers. In: Proceedings of the International Workshop on Multiple Classifier Systems, *LNCS*, vol. 6713, pp. 26–35. Springer (2011)
199. Stone, P.: Layered learning in multiagent systems - a winning approach to robotic soccer. MIT Press, Intelligent robotics and autonomous agents (2000)
200. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An introduction, 2nd edn. MIT Press (2018)
201. Tackett, W.A., Carmi, A.: The donut problem: Scalability and generalization in genetic programming. In: K.E. Kinnear (ed.) Advances in Genetic Programming, pp. 143–176. MIT Press (1994)
202. Taylor, M.E., Stone, P.: An introduction to intertask transfer for reinforcement learning. AI Magine **32**(1), 15–34 (2011)

203. Teller, A., Veloso, M.M.: A controlled experiment: Evolution for learning difficult image classification. In: Proceedings of the Portuguese Conference on Progress in Artificial Intelligence, *LNCS*, vol. 990, pp. 165–176. Springer (1995)
204. Thomason, R., Heckendorn, R.B., Soule, T.: Training time and team composition robustness in evolved multi-agent systems. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 4971, pp. 1–12 (2008)
205. Thomason, R., Soule, T.: Novel ways of improving cooperation and performance in ensemble classifiers. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1708–1715. ACM (2007)
206. Tran, B., Xue, B., Zhang, M.: Genetic programming for multiple-feature construction on high-dimensional classification. Pattern Recogn. **93**, 404–417 (2019)
207. Trianni, V., Lopez-Ibanez, M.: Advantages of task-specific multi-objective optimization in evolutionary robotics. PLOS one **10**(10), e0140056:1–27 (2015)
208. Tsakonas, A., Gabrys, B.: GRADIENT: grammar-driven genetic programming framework for building multi-component, hierarchical predictive systems. Expert Syst. Appl. **39**(18), 13253–13266 (2012)
209. Turner, A.J., Miller, J.F.: Neuroevolution: Evolving heterogeneous artificial neural networks. Evol. Intel. **7**(3), 135–154 (2014)
210. Vahdat, A., Morgan, J., McIntyre, A.R., Heywood, M.I., Zincir-Heywood, A.N.: Evolving GP classifiers for streaming data tasks with concept change and label budgets: A benchmarking study. In: A.H. Gandomi, A.H. Alavi, C. Ryan (eds.) Handbook of Genetic Programming Applications, pp. 451–480. Springer (2015)
211. Vahdat, A., Morgan, J., McIntyre, A.R., Heywood, M.I., Zincir-Heywood, A.N.: Tapped delay lines for GP streaming data classification with label budgets. In: Proceedings of the European Conference Genetic Programming, *LNCS*, vol. 9025, pp. 126–138. Springer (2015)
212. Veeramachaneni, K., Arnaldo, I., Derby, O., O'Reilly, U.: Flexgp - cloud-based ensemble learning with genetic programming for large regression problems. Journal of Grid Computing **13**(3), 391–407 (2015)
213. Verbancsics, P., Stanley, K.O.: Evolving static representations for task transfer. J. Mach. Learn. Res. **11**, 1737–1769 (2010)
214. Virgolin, M.: Genetic programming is naturally suited to evolve bagging ensembles. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 830–839. ACM (2021)
215. Waibel, M., Keller, L., Floreano, D.: Genetic team composition and level of selection in the evolution of cooperation. IEEE Trans. Evol. Comput. **13**(3), 648–660 (2009)
216. Wang, S., Mei, Y., Zhang, M.: Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem. In: A. Auger, T. Stützle (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1093–1101. ACM (2019)
217. Wen, Y., Ting, C.: Learning ensemble of decision trees through multifactorial genetic programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 5293–5300. IEEE (2016)
218. Whiteson, S.: Adaptive representations for reinforcement learning, *Studies in Computational Intelligence*, vol. 291. Springer (2010)
219. Whiteson, S., Kohl, N., Miikkulainen, R., Stone, P.: Evolving soccer keepaway players through task decomposition. Mach. Learn. **59**(1–2), 5–30 (2005)
220. Wiegand, R.P., Liles, W.C., Jong, K.A.D.: An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1235–1245. Morgan Kaufmann (2001)
221. Wolpert, D.H.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)
222. Worzel, W.P., Yu, J., Almal, A., Chinnaiyan, A.: Applications of genetic programming in cancer research. The International Journal of Biochemistry & Cell Biology **41**, 405–413 (2009)
223. Wu, S.X., Banzhaf, W.: Rethinking multilevel selection in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1403–1410. ACM (2011)

224. Yates, C., Christopher, R., Tumar, K.: Multi-fitness learning for behaviour-driven cooperation. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 453–461. ACM (2020)
225. Yong, C.H., Miikkulainen, R.: Coevolution of role-based cooperation in multiagent systems. IEEE Trans. Auton. Ment. Dev. **1**(3), 170–186 (2009)
226. Yüksel, S.E., Wilson, J.N., Gader, P.D.: Twenty years of mixture of experts. IEEE Transactions on Neural Networks and Learning Systems **23**(8), 1177–1193 (2012)
227. Zhou, A., Qu, B., Li, H., Zhao, S., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm Evol. Comput. **1**(1), 32–49 (2011)
228. Zhou, Z., Qiu, Z., Niblett, B., Johnston, A., Zincir-Heywood, N., Heywood, M.I.: A boosting approach to constructing an ensemble stack. In: Proceedings of the European Conference on Genetic Programming, *LNCS*, vol. 13986. Springer (2023)