Grammar-Based Autonomous Discovery of Abstractions for Evolution of Complex Multi-Agent Behaviours

Dilini Samarasinghe*, Michael Barlow, Erandi Lakshika, Kathryn Kasmarik

School of Engineering and IT, University of New South Wales at ADFA, Canberra, Australia

Abstract

This paper presents a grammar-based evolutionary approach that facilitates autonomous discovery of abstractions to learn complex collective behaviours through manageable sub-models. We propose modifications to the design of the genome structure of the evolutionary model and the grammar syntax to facilitate representation of abstractions in separate partitions of a genome. Two learning architectures based on parallel and incremental learning are proposed to automatically derive abstractions. The evaluations conducted with three different complex task environments indicate that the proposed approach with both architectures surpass the performance of generic grammar-based evolutionary models by automatically identifying appropriate abstractions and generating more complex rule structures. The evolutionary process shows further performance improvements with the use of scaffolded environments which were used to train the models in increasingly complex environments across several stages. The results infer that the proposed approach incorporating grammatical evolution with techniques to autonomously discover abstractions can facilitate solving complex problems of agent systems in real-world domains.

Keywords: Grammatical evolution, multi-agent systems, incremental learning, parallel learning, curriculum learning, abstractions

1. Introduction

Multi-agent systems (MASs) are often recognised as models that help study and solve complex and intricate problems in diverse application domains. Artificial intelligence (AI) techniques such as reinforcement learning (RL) [1] and evolutionary computing [2] are commonly used to model such systems to be efficient and effective in addressing challenging real-world problems. However, the increasing complexity of the problems have rendered many multi-agent models infeasible in application domains due to sub-optimal solutions and prolonged learning times to design and train the models [3]. Further, such AI systems

*Corresponding author

are often influenced by errors associated with modelling, and dynamics, disturbances, and uncertainties in real-world environments [4].

As a means of addressing this challenge, techniques have been used to modularise the learning process into manageable sub-components. Approaches such as incremental learning [5], transfer learning [6], and iterative learning control [7] are used not only in the multi-agent domain but also in other applied and theoretical domains such as robotics, batch processing, and periodic disturbance rejection problems [8, 9, 10]. These techniques utilise abstractions to support approaching problems through multiple intermediary steps instead of looking at the problem as a whole. An abstraction is a compact representation of knowledge about a task that helps learn a related more complex task [11]. The idea is inspired from the process adopted by humans in problem solving, as we exploit concepts and knowledge acquired in a simpler situation for learning tasks in a similar but more complex setting. In the context of multi-agent behaviour learning, we define an abstraction as a self-contained modular representation derived from the task or the environment associated with a problem. It is a lossy compression of an originally complex representation, and multiple such simpler compressed components combine to derive the complete solution to the problem.

However, the existing models that utilise such abstractions rely on assumed knowledge of the problem at hand. That is, the abstractions or the intermediary steps of learning are decided manually, and the agents are forced to learn behaviours addressing a pre-determined set of sub-problems. This approach does not support agents with the flexibility to test other routes which may be more efficient in reaching the final goal but not easily comprehensible to human cognition. Although manually configured abstractions can help an artificial learner to approach a challenging task, they may not lead to optimal behaviours at times or may take longer to reach the final goal [12].

Our work focuses on this limitation of the existing AI models that utilise abstractions. We identify that given enough flexibility over the process to the AI itself, an automated learning mechanism has the potential to derive its own set of abstractions. These abstractions may not immediately be evident to human comprehension, but could result in the expected level of performance. Therefore, the paper proposes an alternative technique through an automated evolutionary computing mechanism to derive the representations of sub-behaviours to achieve complex tasks.

We adopt a novel grammar-based evolutionary model with modifications to its genome structure to support the autonomous evolution of the abstractions. Grammatical Evolution (GE) [13] is an evolutionary mechanism inspired by genetic programming. It adopts a genome encoding mechanism that represents behaviour rule structures in binary form which can be mapped to the behaviour syntax based on a grammar. We incorporate GE in an automated mechanism to distribute the learning load across more manageable sub-components. In contrast to existing models, we investigate techniques to give more flexibility over the process to the learning algorithm itself to reduce the bias associated with assumed knowledge of the modules and allow the evolutionary model to au-

tonomously decide the number and the functions of the abstractions in reaching the original complex task. In doing so, we propose modifications to the original GE model with: a genome partitioning technique to facilitate evolution of abstractions in separate partitions; two novel learning architectures based on parallel and incremental learning; and abstraction derivation techniques based on decomposition of the behaviour and environment into simpler components.

The following contributions listed are elaborated in the successive sections of the paper describing the techniques and evolutionary procedure employed in addressing the said issues:

1. A grammar-based evolutionary model that can autonomously evolve abstractions with a MAS is proposed to achieve complex goals.

2. Abstraction deriving techniques are proposed based on two factors: decomposing the agent behaviour into multiple components (behaviour decomposition); and decomposing the environment that the agents behave in (environmental scaffolding).

3. Two learning architectures are experimented to evolve abstractions: incremental learning (learning abstractions incrementally) and parallel learning (learning all abstractions simultaneously)

4. The performance of evolved behaviours is compared against generic evolutionary models where abstractions are inspired by handcrafted subgoals rather than being automatically derived, in three different complex task settings.

We conduct experiments with three different complex tasks designed to evaluate the abstraction learning model. Evaluations are conducted to evaluate the proposed evolutionary model that autonomously evolve abstractions against multiple criteria: (1) compare the performance of the model with and without restrictions on the number of abstractions it can automatically generate; (2) compare the performance against generic evolutionary models and models where genome partitioning is used but with no autonomous evolution; (3) evaluate the contribution of different abstraction derivation techniques based on behaviour and environment decomposition; and (4) evaluate the complexity of derived behaviours under different conditions.

The experimental evaluations demonstrate that the proposed GE model incorporated with the novel learning architectures has potential to autonomously identify suitable behaviour modules to reach complex task requirements with a higher performance compared to generic evolutionary models. Both parallel and incremental learning architectures achieved similar results demonstrating equal competency to autonomously evolve behaviours to solve complex tasks. The models demonstrated higher performances when the learning process is supported by environmental scaffoldings as the models tend to explore a wider range of rule structures with diverse and higher complexity levels when scaffoldings are used. Further, the rule structures evolved with the proposed autonomous

abstraction techniques are more complex than the generic models that do not use abstractions. This indicates that the proposed model has the capacity to generate more intricate behaviours that can address multifaceted problems encountered in real world domains.

The rest of the paper is organised as follows. Section 2 reviews the existing literature related to the use of abstraction learning and grammar-based evolution. Section 3 introduces the problem statement that is addressed within this paper with regard to learning complex multi-agent behaviours through abstractions. The derived grammar-based evolutionary framework is presented in Section 4 and the parallel and incremental learning architectures that are used with the evolutionary framework are introduced in Section 5. Section 6 illustrates the experimental setups and the evaluation results. Finally, Section 7 concludes the paper with a discussion of the results and possible future directions.

2. Related Work

This section elaborates the limitations of existing AI techniques that led towards autonomous derivation of abstractions and the adaptation of a grammatical evolution based learning model.

2.1. Abstraction Learning

As discussed in Section 1, multi-agent models that are required to perform in dynamic and complex environments are difficult to be modelled with traditional techniques that approach problems as a whole due to the computational burden associated with the increased complexity. The need for a modularised or a structured learning process arises from limitations such as: (1) combinatorial explosion that results from the increasing number of actions and states associated with a complex environment; (2) imperfect and partial perception of an environment; and (3) difficulty in identifying an appropriate reward mechanism for multifaceted problems [14]. As such, modularising techniques to approach a problem through the merge of several simpler problem components has been achieved with several architectures.

Incremental learning [5] is a popular concept that investigates techniques to learn complex behaviours through sub-versions by increasing the complexity of the task progressively. This architecture has been adopted in multi-agent domains utilising differential evolution [15], hierarchical and deep reinforcement learning (RL) [16, 17, 18], particle swarm optimisation (PSO) [19] and neural networks [20] based approaches. Variations and extensions of incremental learning architecture have also been proposed, such as, curriculum learning [21] which is inspired by the curriculum structuring of education systems and transfer learning [6] which is focussed on transferring the learned knowledge on one task to learning a related different task. Parallel learning [22] is another way of structuring the abstraction learning process by decomposing a complex task into simpler sub-versions and learning them in separate settings simultaneously.

It has been studied in the context of MASs using techniques such as PSO [23], artificial neural networks (ANNs) [24], RL [25] and evolutionary strategies [26].

The motivations to derive abstractions are often derived from the nature of the behaviour requirements and the nature of the environment in which the agents operate. Behavioural decomposition strategy uses the set of subtasks of a complex task to modularise the learning process [2, 27]. In contrast, environmental scaffolding (complexification) strategy leverages the incremental learning architecture by successively increasing the complexity of the environment such that the agents learn the same behaviours but in increasingly difficult settings [28, 29]. In our study, we adopt these techniques to guide the evolutionary process and update the fitness components while the abstractions are identified automatically by the proposed algorithms.

2.2. Mechanisms to Derive Abstractions Automatically

While abstractions support the learning process in complex MASs, several studies have also explored the drawbacks in the existing learning mechanisms that use abstractions. These studies have specifically looked at the limitations of using assumed knowledge when deriving abstractions that could lead towards achieving poor outcomes. Several attempts have been made in proposing techniques for automatic discovery of abstractions to overcome the said limitations primarily with approaches such as ANNs and RL based approaches.

Graves et al. [30] proposed a mechanism to automatically decide the subgoals to follow through a stochastic policy for neural networks. Mouret et al. [31] investigated biases in manually identifying subgoals for incremental learning, and proposed a multi-objective optimisation-based mechanism to automatically identify and switch between the subgoals required for incremental evolution of feed-forward neuro-controllers in generating behaviours of a robot that seeks a light while avoiding obstacles. A more recent work on a meta learning approach [32] discussed a mechanism for automatic derivation and learning of a hierarchy of subpolicies that aims to learn a master policy faster in a multi-agent context. Mcgovern and Barto [11] and Jardim et al. [33] explored ways of automatically discovering temporal abstractions for subgoals to accomplish a more complex high level goal. Experiments were conducted with a RL mechanism in simulated robot tasks to evolve the behaviours. However, to the best of our knowledge, abstraction learning and autonomous discovery techniques of abstractions have not been experimented with GE based approaches.

2.3. Autonomous Discovery of Abstractions with Grammar-based Evolution

Popular AI techniques such as RL and neural networks suffer from innate limitations of credit assignment problem which is associated with determining the influence of individual actions towards future rewards [34]. Evolutionary computing techniques such as genetic algorithms [35] and genetic programming (GP) [36] are also commonly used in multi-agent domain in designing agent behaviours. However, all these techniques require continuous and extensive manual tuning of parameters for formulation of behavioural rules as well as in

translating the reward function into a quantitative measure [37]. Further, these AI techniques are prone to catastrophic forgetting [38] when combined with an abstraction learning technique such as incremental or parallel learning. As they do not naturally support the management of previously learned knowledge, these systems tend to forget the knowledge of the previously learned abstraction when incremented to a complex level of the problem.

We identify GE as a solution to both manual tuning of parameters behaviour rules, and using abstractions to learn complex tasks. Therefore, this paper explores means to facilitate an automated mechanism to derive abstractions with grammar-based evolution by merging the strengths of both GE and abstraction-based learning. GE is closely related to GP due to their similarities in representing the solution space as programs (tree structures). In contrast to GP that performs genetic operations on the programs themselves, GE represents the genotype in variable length binary strings where a consecutive group of 8 bits is known as a 'codon', and utilises a context-free grammar (CFG) to govern the mapping of the genotype to a valid program [39]. Due to the unique mapping process and the program representation, GE has been used in numerous multi-agent domains including robotics [40] and simulations [41, 42], gaming [43, 44] and visualisations [45].

The conventional grammar-based models, however, do not explore means of automatically deriving behaviour rules of agents. For example, GEESE is a grammar-based evolutionary algorithm that solve swarm tasks based on a predefined set of behaviours where the grammar is used to manipulate only the combination of behaviours to solve a particular task [46]. Similarly, behaviour trees are evolved for Mario AI benchmark using GE with a set of pre-determined behaviour blocks and grammatical evolution approach being used to decide the sequence of actions [47]. These approaches do not evolve agent rules from their atomic components, rather, combine a set of low-level behaviours designed manually to evolve more complex rules.

In contrast, in a recent study [48] we presented a grammar-based evolutionary approach for MASs to automatically derive behaviour rules of agents with limited human intervention in the design process. We proposed a mechanism to evolve complete rule structures from their atomic (or the most primitive) components by synthesising the rules from a pool of control structures, logical and relational connectives, parameters, and elementary actions. We experimented the model with both homogeneous as well as heterogeneous agent systems [49, 50] to evolve complex behaviour rules. Further, as the first attempt in literature of utilising abstractions with GE, we presented an approach where abstractions or the simpler representations of a problem are manually decided while the behaviour rules associated with the abstractions are evolved autonomously [3]. This approach requires external support of a designer to make decisions on the number and nature of abstractions that the learning process should go through in solving a complex task. The work presented in this paper stands out from the previously proposed learning architectures by automating the entire process including abstraction derivation. We investigate novel abstraction learning algorithms to distribute the learning process into manageable sub-components

that can result in the expected performance levels. Here we propose: (1) a novel GE-based framework where genomes can be partitioned into any number of components to incorporate autonomous evolution of abstractions; (2) two novel learning architectures based on parallel and incremental learning to discover abstractions autonomously; (3) two techniques to support the algorithms for derivation of abstractions based on behaviour and environmental decomposition; and (4) compare the evolutionary results against a generic evolutionary model and our previous abstraction learning model which involved manually defined abstractions. We demonstrate that the proposed model can manage the learning process of complex challenges through their simpler representations discovered through the algorithm and automatically derive appropriate behaviour rules to address the challenge.

3. Problem Statement

In this section we introduce the specific problem that is addressed within this paper. We use the term goal to refer to the state that the agent system is required to be in to achieve a successful outcome. Subgoals are intermediate targets that should be achieved by the agent system in order to reach the ultimate goal state. The goal and subgoals are out of the control of agents; rather, they are part of the environment they are performing in. A sub-problem refers to an abstraction (a simpler variant) of the complex problem that targets the goal. A subgoal could be represented by a single sub-problem or a combination of several sub-problems as identified through the evolutionary process with no manual intervention. The evolutionary algorithm determines the number of abstractions that are derived to address a particular subgoal through the learning process. A behaviour module is how the agents should behave in achieving a particular sub-problem. Each behaviour module corresponds to a rule component that is derived based on the grammar to describe how the agents should perform to solve the sub-problem. A combination of multiple behaviour modules addressing each sub-problem (abstraction) result in the complete solution to the problem faced by the agent system.

As such, we investigate how to automate the process of deriving behaviour abstractions to structure the learning process over manageable sub-problems. To test the algorithms, we employ collective decision making tasks which consist of multiple subgoals implying complexity of the required agent behaviours to complete the tasks. Multiple rule components, each representing a single beahviour module corresponding to a single abstraction (sub-problem) are evolved with the evolutionary algorithm. Therefore, considering a complex goal, the agents derive an aggregated set of rule components, each corresponding to a behaviour module that is decided through the learning process, which they will follow to achieve the original goal in unison.

The evaluations are conducted on a homogeneous set of agents which can be defined with the tuple $\mathrm{M} = \{ \Psi, R, G \}$, where:

- $\Psi = \{\ \psi_1, \psi_2, .., \psi_\mu;\ 1<\mu\ \}$ is a homogeneous set of more than one agent ψ, where μ is the number of agents
- $G = \{\ \delta_1, \delta_2, .., \delta_\eta;\ 1<\eta\ \}$ is the goal state the agent system is required to achieve through completion of more than one subgoal δ, where η is the number of subgoals
- $R = \{\ r(\Im_1),\ r(\Im_2),\ ..,\ r(\Im_\lambda)\ \}$ is the set of rule components r, each corresponding to a behaviour module $\Im$, that in combination produce the expected emergent behaviour. λ is the number of behaviour modules.

The expected emergent behaviour for the goal state can be defined in the evolutionary model by the objective function $O(G)$. This corresponds to the sum of objective functions for all the associated subgoals as depicted in Function 1:

$$O(G) = \sum_{i=1}^{\eta} O(\delta_i) \tag{1}$$

Function 2 illustrates that the objective can be achieved only by optimising all the fitness components of the individual subgoals:

$$O(G) \Leftarrow \max\ F(\Psi_G) = \max \sum_{i=1}^{\eta} F(\Psi_{\delta_i}) \tag{2}$$

The objective of the MAS with an agent set of Ψ is to identify each of the behaviour module $\Im$, and the set R of rule components that in aggregation give rise to the best resemblance of the expected goal G.

4. Proposed Grammar-based Evolutionary Framework

Figure 1 illustrates how the autonomously discovered abstractions are utilised in generating complex multi-agent behaviours.

If a direct evolutionary approach was used, it would evolve the behaviour rules in a single genome targeting the global objective function $F(\Psi_G)$ for the goal state G in environment E_τ. But with the proposed autonomous abstraction discovery approach, the evolutionary process is given the flexibility to decompose the global behaviour among multiple rule components by deciding the number of behaviour modules and the function of each module in reaching the expected final goal. The behaviour modules being evolved are guided through the individual fitness components of the subgoals associated with the complex task, as: $\sum_{i=1}^{\eta} F(\Psi_{\delta_i})$. Further, in a second approach, environmental scaffoldings are also used to progressively introduce the evolutionary model to more complex environments so that the learning process can use the previous abstractions learned to better address the task in successive environments. Over multiple generations, the evolutionary process decides the best set of behaviour modules that can achieve the goal G.

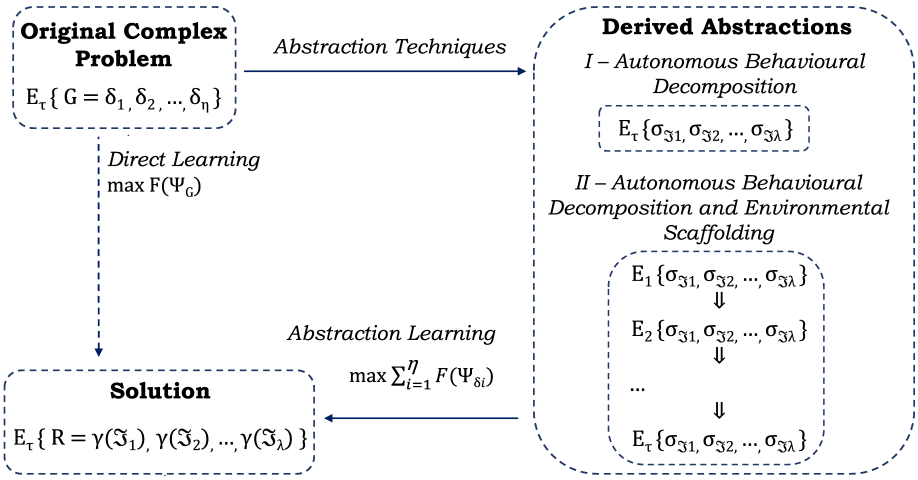


Figure 1: The autonomous abstraction techniques used in the evolutionary model for solving the complex task. In the first strategy, the evolutionary process autonomously identifies the best decompositions of behaviour modules into which the global behaviour can be divided. The evolution is then guided by the fitness components of the subgoals associated with the complex task. In the second strategy, the learning process is also supported by environmental scaffoldings in which a set of increasingly complex environments are associated with the discovered behaviour modules. The derived behaviour modules are mapped to a set of rule components that together address the global behaviour requirement, addressing the complex task presented.

The evolutionary framework for the proposed model is as illustrated in Figure 2. As the first step, a rule space is designed by identifying the respective control structures, parameters, logical and relational connectives and elementary actions of the particular task environment.

Second, a CFG is designed following the syntax presented in the figure. It supports the representation of individual rule components in independent genome partitions each corresponding to a separate behaviour module. Figure 3 illustrates an example application of the proposed genome partitioning technique and the mapping process. As the evolutionary algorithm decides the number of behaviour modules (and thus the number of genome partitions), each partition at the time of generation is separately mapped to the grammar with the mapping function in the top left which is the original mapping function of GE [39]. The final composite rule is an equi-weighted aggregation of all rule components.

The third step of the framework is the evolutionary algorithm itself. The initial population generated based on the above CFG from the rule space is fed to the evolutionary algorithm, where the fitness evaluation is carried out based on the learning architecture used. For parallel learning, the fitness function used is an aggregation of all individual fitness components associated with all subgoals. For incremental learning, the fitness components are introduced incrementally to the system and the genomes are evaluated based on an aggregation

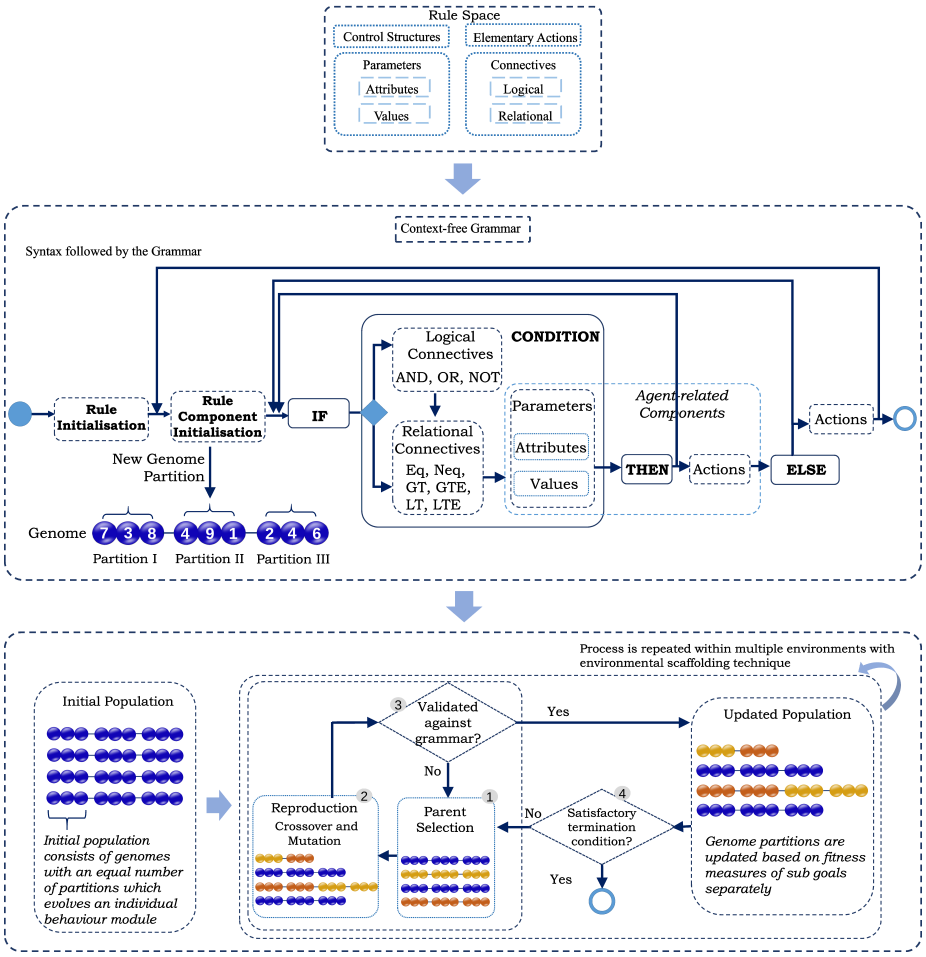


Figure 2: The proposed framework for abstraction learning process with autonomous discovery of behaviour modules. The rule space is first designed which is used to design the CFG based on the syntax presented. The initial population defined based on the said CFG consists of individuals with several partitions, each representing a different behaviour module. The population is fed to the evolutionary model, and the fitness evaluation is conducted based on the learning architecture adopted. The selected parents are then applied with genetic operations and the new offspring is evaluated against the grammar to ensure each rule component represented by every genome partition is syntactically valid. In case a genome contains a partition with an invalid rule component, genetic operations are repeatedly applied until the entire population consists of valid rule components representing behaviour modules. The evolutionary process is iterated until the termination criteria is met. The entire process is repeated within multiple environments with the environmental scaffolding technique.

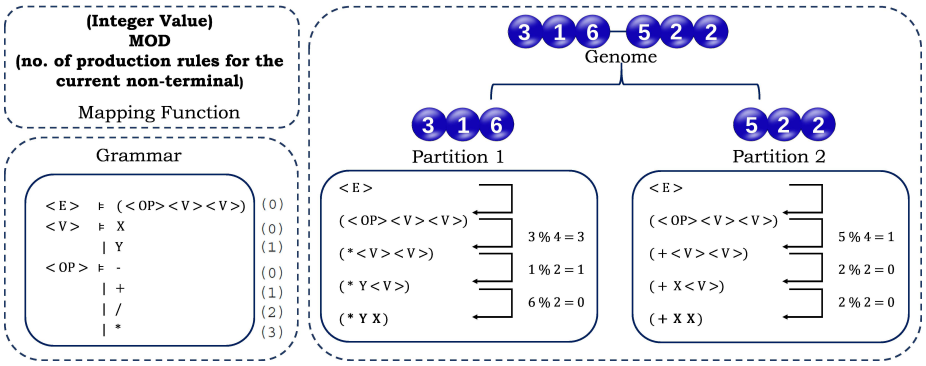


Figure 3: An example genome with two partitions and their mapping to the phenome using the GE mapping function. The original genome represents two rule components and the genome is partitioned accordingly into two, each associated with one rule component. Both partitions have three codons each. After applying the mapping function, the first rule component maps to (* Y X) and the second rule component maps to (+ X X). The aggregation of both is considered as the final solution for the original problem.

of the currently active fitness components. The new offspring generated after fitness evaluation, parent selection and the reproduction steps can have a varying number of behaviour modules which are evolved over the generations. The genetic operations are repeated until all genome partitions are valid. For the environmental scaffolding technique, the process is repeated within increasingly complex environments.

In facilitating the evolutionary process to automatically evolve the genome partitions for behaviour modules, the genetic operations, crossover and mutation are modified by applying them on the genome at different levels. Figure 4 illustrates the application of genetic operators in deriving new genomes for the subsequent generation. Crossover operates at the partition level as it identifies crossover points for genome partitions and the new genomes are generated through merging of these partitions. As each partition corresponds to an independent rule component distinct from other partitions, the crossover operation does not generate invalid individuals. However, the combination of behaviour modules from different genomes to generate offspring leads to diversifying the population and providing the expected level of variety in performance. The single point mutation operation is performed at the codon level by selecting a single codon of a randomly selected genome partition for mutation. The operation ensures diversity within a behaviour module as a new rule component is produced from the mutated codon. If a valid rule component cannot be derived from the new genome partition, the process is repeated with new parent genomes. The application of the two genetic operations, one at partition level and the other at codon level, ensures appropriate diversity while also preserving sufficient knowledge accumulated from previous generations.

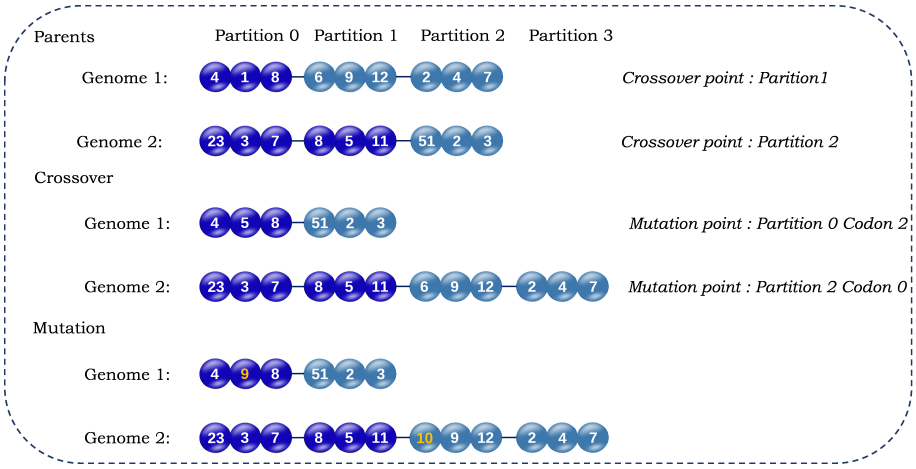


Figure 4: An example application of genetic operations on parents. The two parent genomes consist of three partitions each and three codons in each partition at the beginning of the evolutionary process. The crossover points are selected as partition [1] and [2] for the two parents, respectively. As such, partition [0] of genome 1 and partition [3] of genome 2 are merged, resulting in one new offspring, and the other is the result of combining partitions [0] and [1] of genome 2 and partitions [1] and [2] of genome 1. As the mutation operation is performed at codon level, codon [2] of partition [0] is replaced with a new codon in offspring 1, and codon [2] of partition [0] is replaced in offspring 2.

5. Proposed Learning Architectures

The evolutionary algorithm discussed in the previous section is implemented with the two abstraction learning architectures: parallel learning; and incremental learning. Two techniques are used to guide the evolutionary process based on: subgoals identified with behavioural decomposition; and based on both subgoals and increasingly complex environments identified with behavioural decomposition and environmental scaffolding. When behavioural decomposition is used, the fitness function is decomposed into several components each representing a single subgoal and the abstractions are guided based on each fitness component. For the approach using both behavioural decomposition and environmental scaffolding, several simpler versions of the environment are designed. While the abstractions are guided based on the fitness components, they are also learned in increasingly difficult environments.

5.1. Parallel Learning

The flowchart in the third step of the framework illustrated in Figure 2 demonstrates the learning algorithm. When the algorithm is implemented with a parallel learning architecture, it evolves all genome partitions simultaneously. The framework supports the evolution of multiple behaviour modules to collaboratively address a single subgoal. The procedure is as demonstrated in

Algorithm 1, where the entire genome is subjected to evolution at the same time and the genetic operations are performed based on the aggregated fitness obtained for all subgoals. For those evaluations where environmental scaffolding is adopted, each iteration updates and checks the condition for introducing a new environment. If the condition is met, the current environment is replaced with a slightly more complex environment so that the behaviour modules can use the knowledge from the previous environment to easily approach the problem in the new environment without starting from scratch.

The parallel learning algorithm requires the number of generations the simulation should be run for (ω), the size of the population (β), and the CFG to generate the necessary behaviour rules (ρ) (line 1). As the first step, the population of genomes is initialised (line 2). The initial population is a random set of rules derived based on the CFG illustrated in the step 2 of the flowchart in Figure 2. As parallel learning evolves all abstractions simultaneously, all subgoals are evaluated together. Therefore, all η subgoals are activated (line 3) signaling that the evolutionary process should consider the fitness functions of all subgoals when evaluating the performance of agents. As the agents will be trained across increasingly difficult environments (if the environmental scaffolding technique is followed), the algorithm starts off by activating only the

Algorithm 1 Parallel Learning

Input: β : Size of the population
Ω : Maximum no. of generations
η : Number of subgoals
τ : Number of environments for scaffolding
ρ : Set of CFG production rules
Output: R : Best behaviour module set

1: **procedure** ParallelLearning(ρ, β, Ω)
2: $pop \leftarrow$ InitialisePopulation(β, ρ)
3: $sub\_goal_{1\,to\,\eta} \leftarrow 1$
4: $env_1 \leftarrow 1$ AND $env_{2\,to\,\tau} \leftarrow 0$
5: $\omega \leftarrow 0$
6: EvaluateFitness($pop, sub\_goal_{1\,to\,\eta}$)
7: $R \leftarrow$ GetMostFitRule(pop)
8: **while** $\omega \neq \Omega$ **do**
9: $pop \leftarrow$ GeneticOperations($pop, sub\_goal_{1\,to\,\eta}$)
10: **if** $\tau > 1$ **then**
11: UpdateCondition(env_{cond})
12: **if** env_{cond} **then**
13: $env_{next} \leftarrow 1$
14: EvaluateFitness($pop, sub\_goal_{1\,to\,\eta}$)
15: $R \leftarrow$ GetMostFitRule(pop)
16: $\omega++$
17: **return** R

first (simplest) environment to train the agents in (line 4). If the environmental scaffolding technique is not followed, the first environment corresponds to the only (ultimate) environment that the agents are performing in. The algorithm commences in generation 0 (line 5) and evaluates the fitness of each genome in the population against fitness measures of all subgoals (line 6). The genome with the best fitness value (based on the corresponding behaviour rule applied against the fitness measures) is selected as the current best behaviour for the agent system (line 7). At the end of the generation, genetic operations are applied on the genomes as described in Algorithm 3 (line 9). As mentioned previously, the experiments are two folds where: a set of experiments are conducted with environmental scaffolding; and a set is conducted without environmental scaffoldings. If the environmental scaffoldings are used, then at the end of each generation the condition for updating to the next environment (reaching the expected number of generations in our case) is checked and the environment is updated to the next complexity if the condition has been met (lines 10-13). The fitness evaluation is then repeated (line 14) and the genome with the best fitness is selected as the current best behaviour rule set for the system (line 15). The process is repeated until the expected number of generations are covered.

5.2. Incremental Learning

The incremental learning process is illustrated in Algorithm 2. With this technique, the number and functions of the genome partitions will be automatically decided through the evolutionary process. However, the fitness components for each subgoal will be introduced to the learner in an incremental fashion, unlike parallel learning where all fitness functions are simultaneously taken into account during the fitness evaluation process. The condition adopted to introduce new fitness components is completing a certain number of generations, such that the partitions learn all components currently active during a given instance. The fitness evaluation is carried out based on the aggregation of the fitness functions of the currently active subgoals, but the replacement mechanism and application of genetic operations is similar to that of parallel learning approach.

The evolutionary process starts by initialising the population with random behaviour rules as with parallel learning (line 2). Only the first fitness component is activated at the start (line 3) and the condition for introducing a new fitness component to the algorithm is made false at the begining (line 4). Similar to the parallel learning algorithm, only the first (or the only - if environmental scaffoldings are not used) environment is activated to train the agents in the simplest environment first (line 5). The fitness evaluation is conducted with the fitness measure of the subgoal that the agents are currently being trained to achieve (line 7) and the best genome is selected (line 8). Genetic operations described in Algorithm 3 are applied at the end of the generation when the entire population is evaluated for their fitness. At the end of every generation, the condition for updating the fitness function is checked. If it has been reached, the next subgoal is activated and the respective fitness component is incorporated into the fitness function such that the agents can be trained to achieve

Algorithm 2 Incremental Learning

Input: β : Size of the population
Ω : Maximum no. of generations
η : Number of subgoals
τ : Number of environments for scaffolding
ξ : Condition for introducing new fitness component
ρ : Set of CFG production rules

Output: R : Best behaviour module set

1: **procedure** IncrementalLearning(ρ, β, Ω)
2: $pop \leftarrow$ InitialisePopulation(β, ρ)
3: $sub\_goal_1 \leftarrow 1$ AND $sub\_goal_{2\,to\,\eta} \leftarrow 0$
4: $\xi \leftarrow false$
5: $env_1 \leftarrow 1$ AND $env_{2\,to\,\tau} \leftarrow 0$
6: $\omega \leftarrow 0$
7: EvaluateFitness(pop, sub_goal_1)
8: $R \leftarrow$ GetMostFitRule(pop)
9: **while** $\omega \neq \Omega$ **do**
10: $pop \leftarrow$ GeneticOperations(pop, sub_goals_{active})
11: Update(ξ)
12: **if** $\xi == true$ **then**
13: $sub\_goal_{next} \leftarrow 1$
14: **if** $\tau > 1$ **then**
15: UpdateCondition(env_{cond})
16: **if** env_{cond} **then**
17: $env_{next} \leftarrow 1$
18: $sub\_goal_1 \leftarrow 1$ AND $sub\_goal_{2\,to\,\eta} \leftarrow 0$
19: EvaluateFitness(pop, sub_goals_{active})
20: $R \leftarrow$ GetMostFitRule(pop)
21: $\omega++$
22: **return** R

all subgoals currently active (lines 12-13). Similar to the parallel learning algorithm, the condition for updating to the next complex environment is also checked at the end of each generation if the environmental scaffoldings are used, and it is updated accordingly if the condition has been met. If the environment is updated, the evaluation process is repeated starting again from only the first subgoal and incrementally introducing the other subgoals to the system (lines 16-18). The fitness evaluation is then repeated with the active subgoal fitness components (line 19) and the genome with the best fitness is selected at the end of each generation (line 20). The process is repeated until the expected number of generations is reached.

Algorithm 3 Genetic Operations

Input: pop : Population
ρ : Set of CFG production rules
ϖ : Set of active subgoals
Output: pop_{new} : Population with new offspring

1: **procedure** GeneticOperations(pop, ϖ)
2: $valid \leftarrow false$
3: **while** $!valid$ **do**
4: $parents \leftarrow$ ParentSelection(pop, ϖ)
5: $\lambda_1 \leftarrow$ NumberOfPartitions($parent_1$)
6: $\lambda_2 \leftarrow$ NumberOfPartitions($parent_2$)
7: $c_{M1}, c_{M2} \leftarrow null$
8: $c_{part_1} \leftarrow$ RandomPartitionNumber(λ_1)
9: $c_{part_2} \leftarrow$ RandomPartitionNumber(λ_2)
10: **for** $p_1, p_2 \in parents$ **do**
11: $children \leftarrow$ Crossover($p_1, p_2, c_{part_1}, c_{part_1}$)
12: $m_{1_{part}} \leftarrow$ RandomPartitionNumber(λ_1)
13: $m_{2_{part}} \leftarrow$ RandomPartitionNumber(λ_2)
14: **for** $c_1, c_2 \in children$ **do**
15: $c_{M1} \leftarrow$ Mutate($c_1, prob_{mut}, m_{1_{part}}$)
16: $c_{M2} \leftarrow$ Mutate($c_2, prob_{mut}, m_{2_{part}}$)
17: **if** MapCFG(ρ, c_{M1}) AND MapCFG(ρ, c_{M2}) **then**
18: $valid \leftarrow true$
19: $I_W \leftarrow$ GetTwoWorstFitIndivs(pop)
20: **for** $I_{W1}, I_{W2} \in I_W$ **do**
21: $I_{W1} \leftarrow$ ReplaceIndiv(I_{W1}, c_{M1})
22: $I_{W2} \leftarrow$ ReplaceIndiv(I_{W2}, c_{M2})
23: $pop_{new} \leftarrow$ Update(pop, I_{W1}, I_{W2})
24: **return** pop_{new}

The algorithm for modified genetic operations in Figure 4 is illustrated in Algorithm 3. Parent selection is first performed based on the fitness values obtained for each genome using the aggregation of fitness components. The two genomes with the best fitness are selected from the population (line 4). The

number of genome partitions (each representing a single behaviour module) is noted for both genomes (lines 5-6). Then a random partition number is selected for each genome as the crossover point. With this mechanic, different behaviour modules can be mixed to diversify the behaviour of the agent system in the next generation (lines 8-9). Next, offspring is generated by applying crossover at the selected partitions on the parents (line 11). To further enhance the exploitation of behaviour rules, a random codon in a random genome partition of each offspring is selected to be applied with mutation (lines 12-13) and mutation is performed (lines 14-16). At the end of applying genetic operations, the offspring is then mapped against the CFG to check the validity of the newly generated behaviour rules (lines 17-18). If the genome does not map to valid rule components, the process is repeated with new parents. Once valid offspring are generated, the two genomes with the least good fitness values are replaced by the new offspring (lines 19-22), and the population is updated and returned for fitness evaluation with the parallel or incremental learning model (lines 23-24).

6. Experimental Evaluations

The experimental evaluations are focused on three main areas of interest: the performance of the proposed learning architectures in automatically discovering abstractions; the impact and implications of the automatically derived modules and the use of behavioural decomposition and environmental scaffoldings to support the evolutionary process; and the complexity of the rule structures evolved.

6.1. Problem Definition

Given,
μ - number of agents in the system
λ - number of genome partitions = behaviour modules
β - evolutionary population size

The simulation model consists of a set B of simulation agents (b) such that:

$$B = \{b_1, b_2, \ldots .., b_\mu\} \tag{3}$$

The learning population P consists of κ genomes (φ), each with a distinct solution to address the complex task requirement:

$$P = \{\varphi_1, \varphi_2, \ldots .., \varphi_\beta\} \tag{4}$$

Each solution represented by a genome φ consists of a number of partitions σ which can be different from the number of partitions of other genomes. Each rule component represented by a partition σ, corresponds to a single behaviour module ($\Im$). The genomes are evaluated with the agent set A, using an aggregated fitness function based on the applied learning architecture:

$$\varphi = \{\sigma(\Im_1), \sigma(\Im_2), \ldots .., \sigma(\Im_\lambda)\} \tag{5}$$

6.2. Task Definition

As discussed in Section 4, the proposed evolutionary model is designed for evolution of complex tasks with multiple subogals. Therefore, three task environments were designed to evaluate the proposed models which consist of three subgoals: coordination; obstacle avoidance; and target search, that should be completed in different orders. The expectation is for the agents to try to navigate as a group while avoiding obstacles and locate a target. Collisions cause agents to bounce back, and their vision is restricted beyond obstacles and walls. Figures 5, 6 and 7 represent the designed tasks.

- **Task with unordered subgoals:** This task requires agents to concentrate on all three requirements simultaneously. The agents should form a group and navigate through obstacles and locate the target. Fitness of the agent system is calculated by aggregating the fitness component of each subgoal. There is no order as to which subgoal the agents should learn first. They can focus on any subgoal first (or some/all) and improve their performance by learning the other subgoals subsequently. However, for the agents to achieve a higher performance, they should be competent in all three subgoals equally. The environments E1, E2, and E3 are described below.

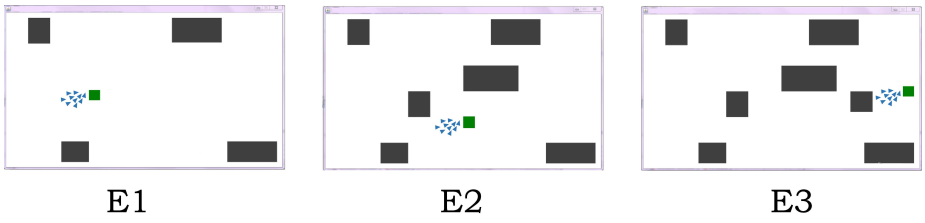


Figure 5: The designed task with unordered subgoals. The three subgoals — coordination, obstacle avoidance and target search — are to be accomplished simultaneously. E1 and E2 are simpler environments designed to test the environmental scaffolding technique for the agents to learn the original problem in E3 through simpler representations.

- **Task with sequential subgoals:** This task has three subgoals that the agents have to master sequentially as they will not be able to reach the next stage without mastering competency in one subgoal. To facilitate this requirement, two walls are introduced partitioning the environment into three compartments. Consider the task of E1 in column 1 of Figure 6: the agents start from the first compartment (Figure 6 E1 row 1). They should learn to form a group and navigate together (with a satisfactory average distance from each other determined by the fitness function described below) to get the first wall opened. The wall wouldn't open until coordination behaviour is learned. An opened wall stays open for the rest of the simulation (Figure 6 E1 row 2). A target is placed in the next

compartment for the agents to locate. The agents should maintain coordination and navigate together as a group and then locate the target. The wall to the third compartment would only open if the agents can successfully coordinate and locate the target. When this wall opens, the target moves to a location in the third compartment. The compartment also consists of obstacles (Figure 6 E1 row 3). The agents then have three subgoals to achieve: coordinate as a group when navigating, locate the new target, and avoid obstacles. To achieve expected level of performance, the agents have to learn coordination, target search, and obstacle avoidance in the sequential order and also remember the previously learned behaviours while learning a new subgoal.

- **Task with hierarchical subgoals:** Hierarchical subogals imply that certain subgoals should be mastered simultaneously while others should be learned in subsequent order. This task has a single wall partitioning the environment into two compartments. In the first apartment where the agents are first located, they should learn to coordinate and form a group when navigating and should locate a target placed in the compartment (Figure 7 E1 row 1). The wall only opens up once the agents locate the target as a group. Once both subgoals are achieved opening the wall, the target shifts to the next compartment (Figure 7 E1 row 2), where obstacles are also present. The agents then have to achieve three subgoals: coordinate, avoid obstacles, and locate the target. As such, the agents should first learn coordination and target search subgoals simultaneously, and should then master obstacle avoidance subgoal afterwards.

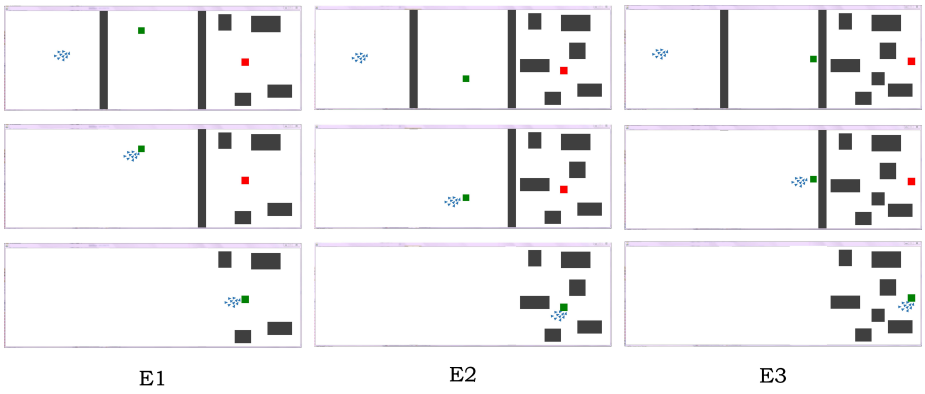


Figure 6: The designed task with sequential subgoals. The three subgoals — coordination, target search and obstacle avoidance — should be accomplished sequentially. E1 and E2 are simpler environments designed to test the environmental scaffolding technique to learn the original problem in E3 through simpler representations. Targets in green represent the currently active target, and the red targets are activated after reaching the green target in the environment.

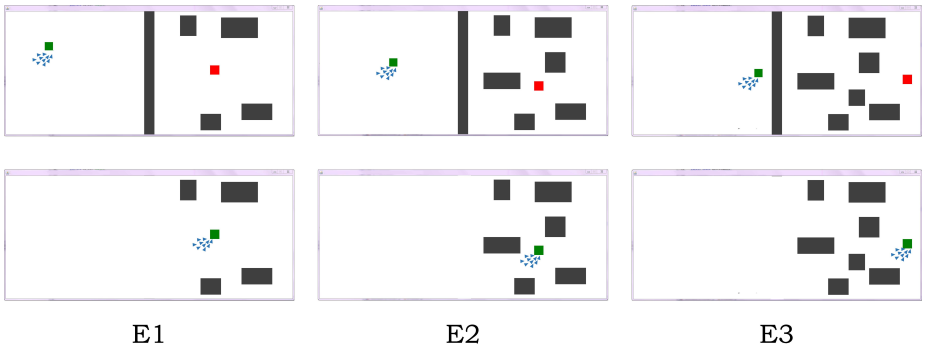


Figure 7: The designed task with hierarchical subgoals. Coordination and target search should be accomplished together before obstacle avoidance. E1 and E2 are simpler environments designed to test the environmental scaffolding technique to learn the original problem in E3 through simpler representations. Targets in green represent the currently active target, and the red targets are activated after reaching the green target in a particular environment.

For experiments evaluating the environmental scaffolding technique two simpler environments are designed along with the original complex environment. In the Figures 5, 6 and 7, E3 illustrates the original complex problem environment and E2 and E1 are their simpler versions. The difficulty of the task is progressively increased over the environments E1, E2, and E3 by changing the arrangement of the target and increasing the number of obstacles.

The attributes altered for each environment are presented in Table 1. Each environment (from E1-E3) has increasingly more obstacles thus increasing the total area of the surface covered by the obstacles. As a result, each environment is made increasingly difficult to navigate for the agents. The Table 1 documents the number of obstacles as 4, 6, and 7 for each environment E1, E2, and E3 in the three tasks. The attribute 'area covered by obstacles' represent the percentage surface area that is covered by all obstacles in each environment from E1-E3. As can be seen, the area is also increased for each increasingly complex environment. This makes it increasingly difficult for agents to avoid obstacles and helps evaluate whether mastering the behaviours in a simpler environment with less obstacles help learn more efficiently when a more complex is introduced. Further, the difficulty of finding the target is increased by placing it within an increasing number of obstacles and/or by placing it further away from the start position of the agents in each subsequent environment. The start position of the agents is the left end of the environment for all 3 tasks. The attribute 'location of target (1/2) from left end' denotes the distance that the target is placed from the start position as a percentage of the total width of the environment. The distance is increased across the environments E1-E3 for all three tasks such that it is increasingly placed further away from the agents' start position.

Equations 6, 7 and 8 illustrate the quantitative fitness measures (minimising functions in the range 0-1) of coordination, obstacle avoidance and target search,

Table 1: Attributes Related to Environmental Scaffolding

Attribute	Environment			
	E1	E2	E3	
Task with unordered subgoals				
Number of obstacles	4	6	7	
Location of target from left end	30%	50%	94%	(as a % of world width)
Area covered by obstacles	8%	13%	14%	(as a % of total area)
Task with sequential subgoals				
Number of obstacles	4	6	7	
Location of target 1 from left end	45%	50%	63%	(as a % of world width)
Location of target 2 from left end	82%	83%	97%	(as a % of world width)
Area covered by obstacles	4%	7%	8%	(as a % of total area)
Task with hierarchical subgoals				
Number of obstacles	4	6	7	
Location of target 1 from left end	14%	24%	42%	(as a % of world width)
Location of target 2 from left end	71%	76%	96%	(as a % of world width)
Area covered by obstacles	6%	8%	9%	(as a % of total area)

respectively. Performances closer to the best are rewarded higher by using a non-linear modified four parameter logistic regression equation. Agents receive a lower value (corresponding to a higher fitness as it is a minimising function) for coordination f_c, when they are within an average separation distance of 100 (5 - 10% dispersion for all 3 environments) units or less from the other agents. Only a highly coordinated group can result in a similar behaviour, allowing the agent group to be rewarded more for a higher performance. The fitness gradient becomes steeper for obstacle avoidance measure f_a if the number of collisions fall below five (= 0.5 collisions allowed for each of the 10 agents in the system), proving that the group is avoiding collisions actively. Similar to coordination measure, the inflection point for target search measure f_t is also set at 100 units from the target, as the agents can be considered to be within satisfactory range from the target.

$$f_c = \frac{1}{\mu}\sum_{i=1}^{\mu}(1 - \frac{1}{(1+\frac{x}{100})}) \qquad ; x = \frac{1}{\mu - 1}\sum_{j=1}^{\mu} dist|l_i - l_j| \; ; \; j \neq i \tag{6}$$

$$f_a = \frac{1}{\mu}\sum_{i=1}^{\mu}(1 - \frac{1}{(1+\frac{x}{5})}) \qquad ; x = number\,of\,collisions \tag{7}$$

$$f_t = \frac{1}{\mu}\sum_{i=1}^{\mu}(1 - \frac{1}{(1+\frac{x}{100})}) \qquad ; x = dist|l_t - l_i| \; ; \; l_t = location\,of\,target \tag{8}$$

6.3. Experimental Setup

The proposed autonomous discovery model is evaluated with a virtual multi-agent simulation system of 10 agents in the three environments with differently arranged subgoals. The behaviour modules of agents are evolved through automatically derived abstractions, and the agents can choose to learn these subgoals in multiple behaviour modules. The agent system adopts a homogenous architecture with the same structural attributes for all agents. The agents have the capability of interacting with the objects in the environment and the neighbouring agents based on their range of vision.

All fitness components are evolved simultaneously (⬇⬇⬇) over 1350 generations with the parallel learning approach. When environmental scaffolding is used, the agents are introduced to each new environment at equal intervals of 450 generations. For incremental learning evaluations, three patterns are tested where fitness components are introduced strictly after one another (⋮), two components are introduced together and the next one presented after a number of generations (⬇⬇ / ⬇), and one component is introduced first after which the other two are introduced together (⬇ / ⬇⬇). With environmental scaffolding, the fitness component transition takes place within each environment.

Table 2 illustrates the evolutionary attributes of the experiments designed. For the offspring generation with GE, steady state replacement mechanism [39] is adopted where only the two individuals with the least fitness scores are replaced in each iteration. The number of genome partitions is decided through the evolutionary process. A wrapping limit of 10 is introduced to limit the destructive nature of GE by reusing the same codon for a number of wraps until a terminating state for the rule can be obtained. The world sizes of the task environments are adjusted to allow for equal widths in all compartments to eliminate any biases towards a certain subgoal.

The initial conditions that influence the evolutionary process are the starting locations of agents in the system and their range of vision. To address the impact from agent start positions, the analysis is conducted with 30 evolutionary runs of each experiment with random agent locations for each new run in a wraparound environment. Known seed values were used for the random number

generator of each experiment to ensure repeatability of the experiments. The attributes related to the vision range: distance of vision, and angle of vision, are incorporated into the grammar syntax such that the evolutionary algorithm itself can evolve the best applicable attributes without being limited by any initial values stated manually.

6.4. Performance Evaluation

To investigate the capacity of the proposed model to perform under restricted conditions, the results are compared across two methods:

- Unrestricted Modules: The learning models are not restricted for the number of behaviour modules they can evolve in achieving a given task. The algorithms are allowed the flexibility to change the number of modules for each individual during evolution with the genetic operations discussed.
- Restricted Modules: The learning models are restricted to a maximum of 10 modules throughout the evolutionary process. The genetic operations are modified to allow for selection of a crossover point such that this condition is met for the new offspring to be generated. The value of 10 was

Table 2: Evolutionary Attributes of the Setup for Autonomous Discovery of Abstractions

Attribute	Value
Genome	
No. of Genome Partitions	= No. of Behaviour Modules $\neq$ No. of Subgoals
Component Size	100 codons of 8 bits
Maximum Wrappings	10
Evolutionary Algorithm	
Population Size	50
Evolutionary Strategy	SSR
Parent Selection	Tournament (size:5)
Mutation Probability	0.5
Crossover	1.0
Maximum Generations	1350
Evolutionary Runs	30
Simulation Environment	
No. of Agents	10
World Size	1000 x 600 - Unordered
	1800 x 600 - Sequential
	1400 x 600 - Hierarchical
World Nature	Wraparound
Reaction on Collisions	Bounce Off
Vision beyond Obstacles	Blocked
Agent Speed	4 units per tick

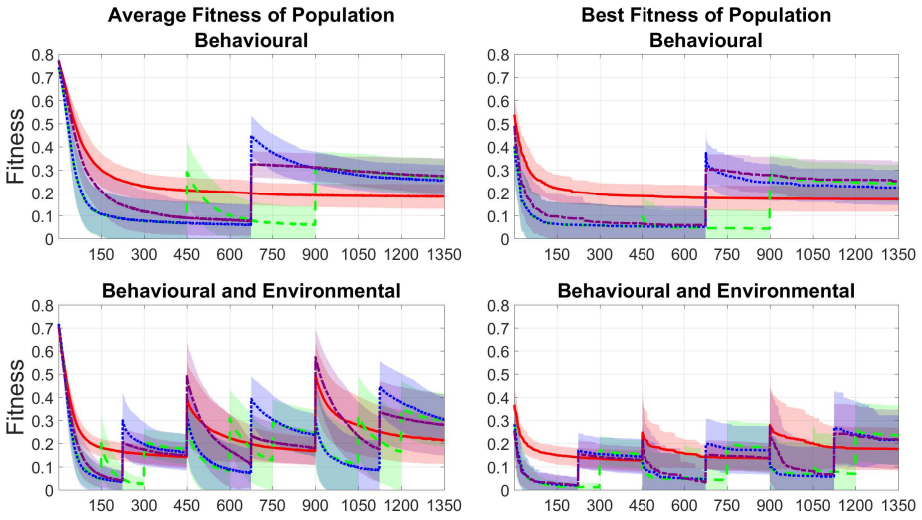
selected as an intermediate number greater than the number of observable subgoals of the tasks (coordination, obstacle avoidance, target search) but is still restrictive in terms of the achievable complexity of rules.

Figures 8, 9 and 10 illustrate the evolutionary results for the four learning methods with and without environment scaffoldings in environments with unordered, sequential and hierarchical subgoals, respectively. The results are presented with both techniques, using restricted and unrestricted modules, for 1350 generations averaged across 30 runs. The respective statistical results are presented in Tables 3 and 4.

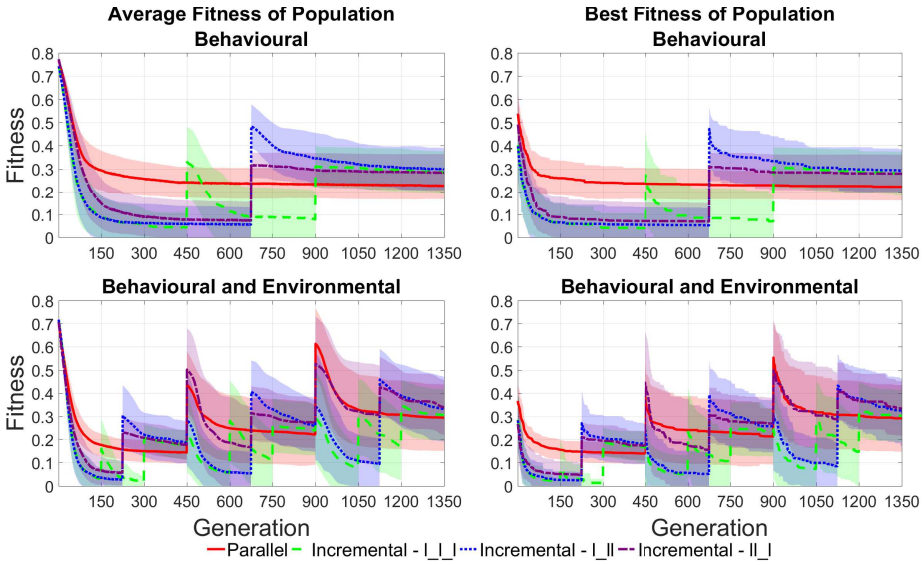
Table 3: Statistical Results Summary for the Performance of Autonomous Abstraction Architectures with Behavioural Decomposition

Method	Unrestricted		Restricted	
	K-W H-test (p-value)	M-W U-test (p-value)	K-W H-test (p-value)	M-W U-test (p-value)
Unordered	0.00		0.01	
PL - INC I_I_I		**0.00**		**0.01**
PL - INC I_II		**0.00**		**0.00**
PL - INC II_I		**0.00**		**0.01**
INC I_I_I - INC I_II		0.39		0.58
INC I_I_I - INC II_I		0.75		0.70
INC I_II - INC II_I		0.22		0.83
Sequential	0.33		0.29	
PL - INC I_I_I		0.28		0.34
PL - INC I_II		0.64		0.83
PL - INC II_I		0.25		**0.04**
INC I_I_I - INC I_II		0.13		0.64
INC I_I_I - INC II_I		0.17		0.51
INC I_II - INC II_I		0.86		0.19
Hierarchical	0.96		0.11	
PL - INC I_I_I		0.94		0.06
PL - INC I_II		0.70		0.21
PL - INC II_I		0.90		**0.02**
INC I_I_I - INC I_II		0.56		0.43
INC I_I_I - INC II_I		0.89		0.77
INC I_II - INC II_I		0.75		0.30

The Mann-Whitney U test and Kruskal-Wallis H test are used for comparisons with the statistical significance level set at $p = 0.05$. Based on the results, both architectures are capable of improving agent behaviours consistently within restricted and unrestricted conditions. In the task with unordered subgoals, both restricted and unrestricted versions with no environmental scaffoldings show that the parallel learning architecture outperforms the other architectures

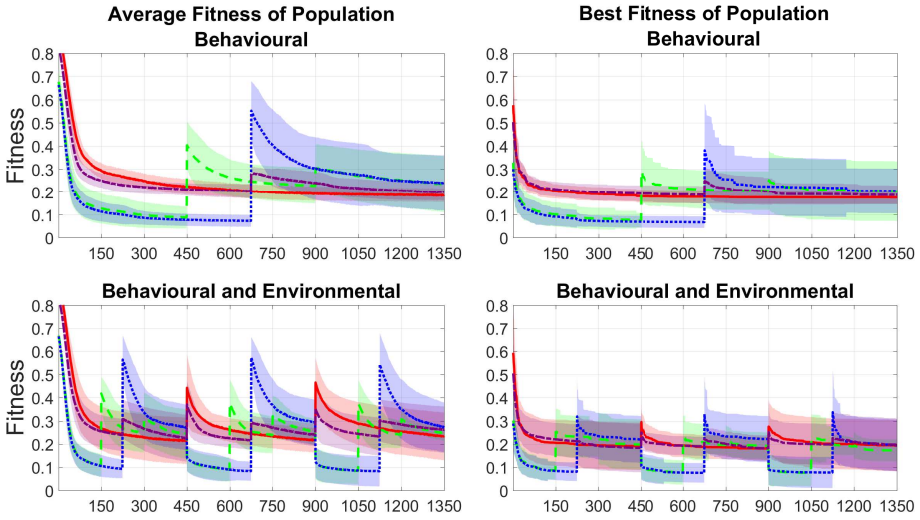


(a) Unordered : Unrestricted Modules

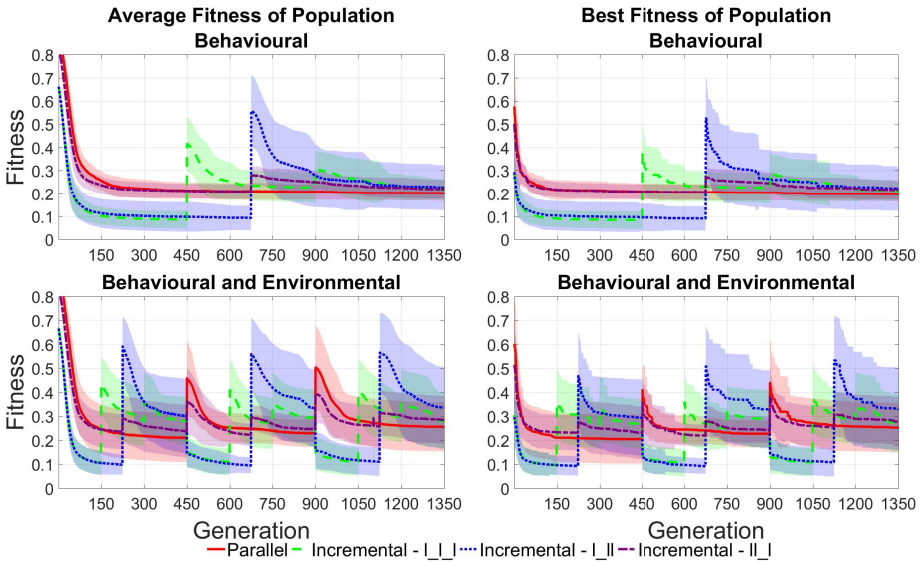


(b) Unordered : Restricted Modules

Figure 8: Evolution results for the task with unordered subgoals based on the four methods using autonomous behaviour module discovery (with unrestricted and restricted modules) and environmental scaffolding abstraction techniques. The left column illustrates the average fitness progression of the solution space and the right column illustrates the evolution of the best solution through 1350 generations. The experimental results are averaged across 30 runs and the shaded area depicts the SD.

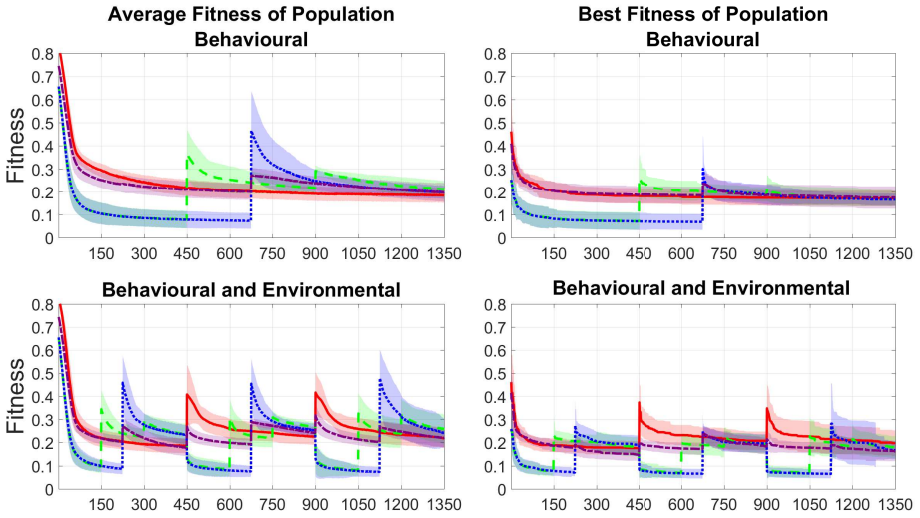


(a) Sequential : Unrestricted Modules

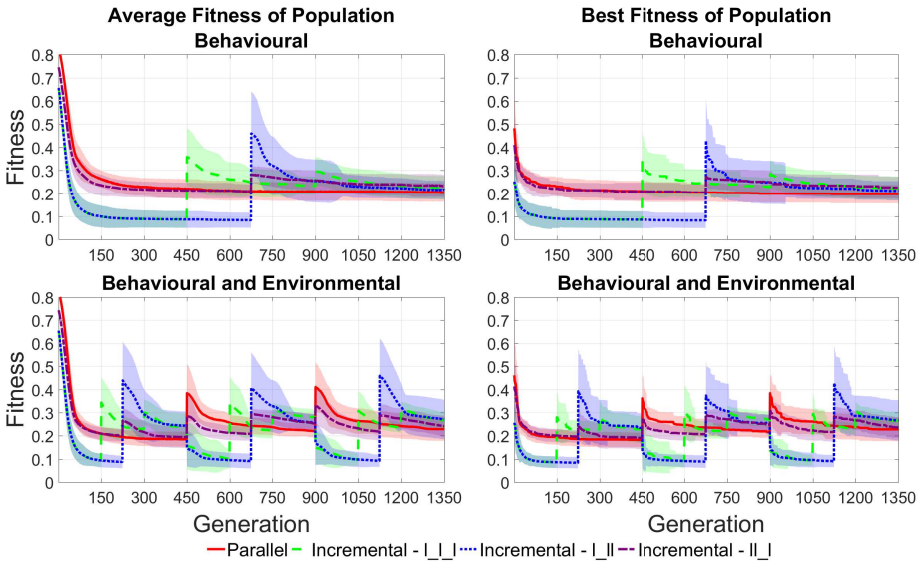


(b) Sequential : Restricted Modules

Figure 9: Evolution results for the task with sequential subgoals based on the four methods using autonomous behaviour module discovery (with unrestricted and restricted modules) and environmental scaffolding abstraction techniques. The left column illustrates the average fitness progression of the solution space and the right column illustrates the evolution of the best solution through 1350 generations. The experimental results are averaged across 30 runs and the shaded area depicts the SD.



(a) Hierarchical : Unrestricted Modules



(b) Hierarchical : Restricted Modules

Figure 10: Evolution results for the task with hierarchical subgoals based on the four methods using autonomous behaviour module discovery (with unrestricted and restricted modules) and environmental scaffolding abstraction techniques. The left column illustrates the average fitness progression of the solution space and the right column illustrates the evolution of the best solution through 1350 generations. The experimental results are averaged across 30 runs and the shaded area depicts the SD.

Table 4: Statistical Results Summary for the Performance of Autonomous Abstraction Architectures with both Behavioural Decomposition and Environmental Scaffolding

Method	Unrestricted		Restricted	
	K-W H-test (p-value)	M-W U-test (p-value)	K-W H-test (p-value)	M-W U-test (p-value)
Unordered	0.18		0.39	
PL - INC I_I_I		**0.03**		0.39
PL - INC I_II		0.08		0.45
PL - INC II_I		0.22		0.11
INC I_I_I - INC I_II		0.79		0.85
INC I_I_I - INC II_I		0.63		0.19
INC I_II - INC II_I		0.68		0.49
Sequential	0.58		0.10	
PL - INC I_I_I		0.33		0.51
PL - INC I_II		0.82		0.06
PL - INC II_I		1.00		**0.03**
INC I_I_I - INC I_II		0.14		0.20
INC I_I_I - INC II_I		0.38		0.19
INC I_II - INC II_I		0.97		0.86
Hierarchical	0.06		0.02	
PL - INC I_I_I		0.18		**0.01**
PL - INC I_II		**0.03**		0.07
PL - INC II_I		**0.02**		0.60
INC I_I_I - INC I_II		0.30		0.43
INC I_I_I - INC II_I		0.25		**0.02**
INC I_II - INC II_I		0.80		0.17

(Mann-Whitney U $p < 0.05$). However, this is not evident in the environments with sequential and hierarchical subgoals as the learning architectures demonstrate no statistically significant difference in their performance (Mann-Whitney U $p > 0.05$). When environmental scaffoldings are used, the significance of the difference in the performance levels is further reduced. Further, these results also demonstrate that no relationship exists between the nature of the task (in terms of the organisation of its subgoals) and the type of learning architecture used.

The robustness of the algorithms across the three tasks is examined in Table 5 for the unrestricted modules version [1]. The performance variability is analysed in terms of minimum and maximum fitness values achieved, starndard deviation (SD) of the performance across multiple runs and the number of runs within

[1]The respective results for the experiments using restricted modules are presented and discussed in the supplementary file.

Table 5: Robustness of the Algorithms with Unrestricted Modules

Algorithm	Minimum	Maximum	Mean	SD	Runs within (mean ± 2SD)
Unordered					
Behavioural					
PL	0.07	0.36	0.17	0.06	28/30
INC I_I_I	0.06	0.38	0.24	0.08	29/30
INC I_II	0.07	0.37	0.22	0.08	29/30
INC II_I	0.11	0.45	0.25	0.09	29/30
Behavioural and Environmental					
PL	0.06	0.40	0.18	0.09	28/30
INC I_I_I	0.07	0.68	0.24	0.13	28/30
INC I_II	0.06	0.64	0.21	0.11	29/30
INC II_I	0.06	0.66	0.22	0.13	29/30
Sequential					
Behavioural					
PL	0.12	0.24	0.18	0.03	29/30
INC I_I_I	0.11	0.67	0.20	0.13	28/30
INC I_II	0.14	0.67	0.21	0.10	28/30
INC II_I	0.12	0.26	0.19	0.04	30/30
Behavioural and Environmental					
PL	0.12	0.76	0.19	0.11	29/30
INC I_I_I	0.11	0.29	0.17	0.05	25/30
INC I_II	0.11	0.76	0.20	0.11	29/30
INC II_I	0.11	0.76	0.20	0.11	30/30
Hierarchical					
Behavioural					
PL	0.1186	0.27	0.17	0.03	28/30
INC I_I_I	0.12	0.24	0.17	0.03	30/30
INC I_II	0.12	0.23	0.17	0.03	29/30
INC II_I	0.11	0.27	0.18	0.05	29/30
Behavioural and Environmental					
PL	0.10	0.32	0.20	0.06	29/30
INC I_I_I	0.10	0.28	0.18	0.05	27/30
INC I_II	0.08	0.27	0.17	0.04	27/30
INC II_I	0.09	0.25	0.16	0.04	29/30

± 2SD of the mean for the three tasks. The learning architectures were able to achieve an improvement in performance using environmental scaffoldings in 10 out of 12 experiments (for four methods with the three tasks). However, the SD also generally increases with environmental scaffoldings (10 out of 12 experiments). As such, it can be deduced that the environmental scaffolding technique is slightly less robust than when both behaviour modules and environmental scaffoldings are used, but it still has more potential to generate highly fit behaviours. In general, however, all four methods with and without environmental scaffoldings maintain a lower SD of < 0.15 and manage to generate behaviours with fitness values within ± 2SD of the mean for most of the cases (least is 25 out of 30).

Therefore, it can be concluded that both parallel and incremental architectures are capable of generating complex behaviours with autonomously derived behaviour modules with and without the help of environmental scaffoldings, in restricted and unrestricted conditions, for the number of behaviour modules allowed. Both architectures are capable of generating robust solutions, while in comparison, using environmental scaffoldings slightly reduces the robustness but increases the chances of achieving a higher performance.

6.5. Comparison against the Generic Evolutionary Model

In this section, the performance of the two approaches, with restricted and unrestricted modules, are compared against each other and against two other models:

- Generic Evolution - Generic GE based generation of behaviours without using the genome partitioning or abstraction learning techniques.
- Partitioned Genome - Generic GE based model, but adopting the genome partitioning technique to evolve rule components independently but without using abstraction learning architectures.

The comparison results for parallel, incremental - I_I_I, incremental - I_II and incremental - II_I methods with restricted and unrestricted modules against the generic evolutionary models are presented in Figure 11. Comparisons are illustrated for experiments with and without using environmental scaffoldings. Both restricted and unrestricted versions of abstraction architectures are capable of generating significantly higher performance in comparison to the two generic evolutionary models with both the tasks with unordered and sequential subgoals (Mann-Whitney U $p < 0.05$).

Table 6 demonstrates the statistical results of comparisons of the two versions using the four learning methods. According to the results, the unrestricted version is capable of outperforming the restricted version (Mann-Whitney U $p < 0.05$) in nine out of 12 experiments (except in incremental I_I_I: unordered - Mann-Whitney U $p = 0.13$, incremental II_I: unordered - Mann-Whitney U $p = 0.28$, and incremental I_II: sequential - Mann-Whitney U $p = 0.06$) without environmental scaffoldings, and in all 12 experiments with environmental scaffoldings. The significance is higher in experiments where scaffoldings are used.

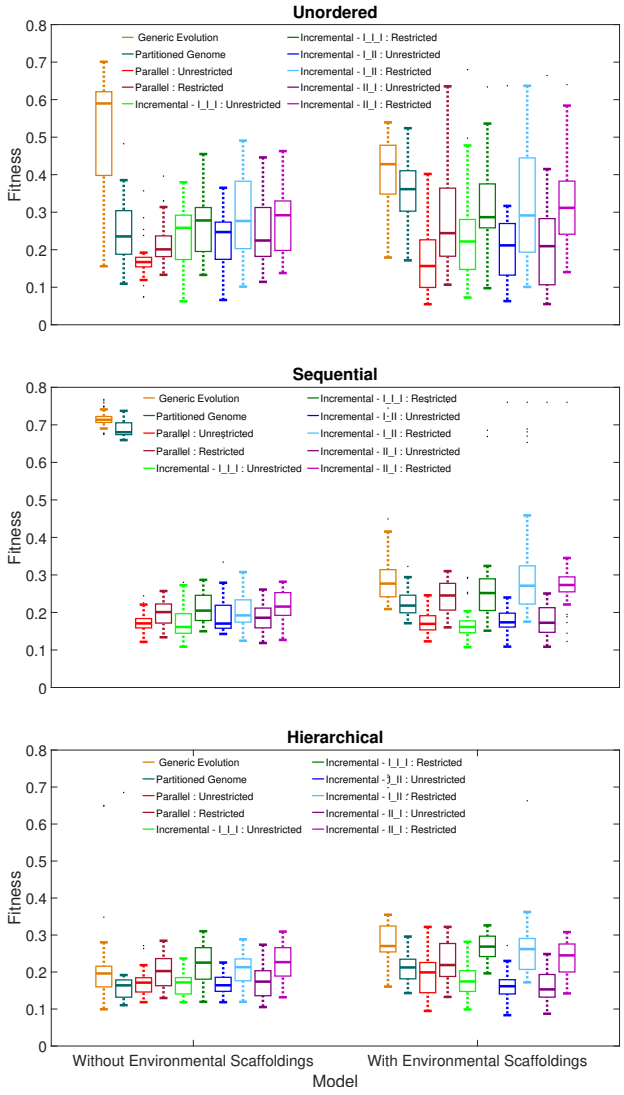


Figure 11: Comparison of performance of the best solutions for the four abstraction models against the generic evolutionary models across 30 runs. The bottom edge, central line and top edge indicate the 25th percentile, median result and 75th percentile, respectively. The extensions of whiskers run up to the extreme results not deemed outliers. The same evolutionary attributes (including crossover and mutation rates, the number of generations and number of wraps) used for the abstraction models are used for the compared generic models.

As such, it can be deduced that restrictions on algorithm search spaces in terms of the number of behaviour modules can have an impact on the performance, which in this case was negative with 10 modules. This provides evidence to support further investigations on constraining algorithms to understand their impact and correlation the behaviour modules have with the nature of the task.

6.6. Analysis of Abstraction Techniques

The use of autonomously discovered behaviour modules and environmental scaffoldings in the learning architectures, their impact on the performance and other implications are discussed in this section. As demonstrated in the previous section, the unrestricted version outperforms the restricted version. Therefore the rest of the analyses are mainly focused on the abstraction learning version where the number of modules that can be derived is unrestricted. The results with the restricted version are presented in the supplementary file for reference.

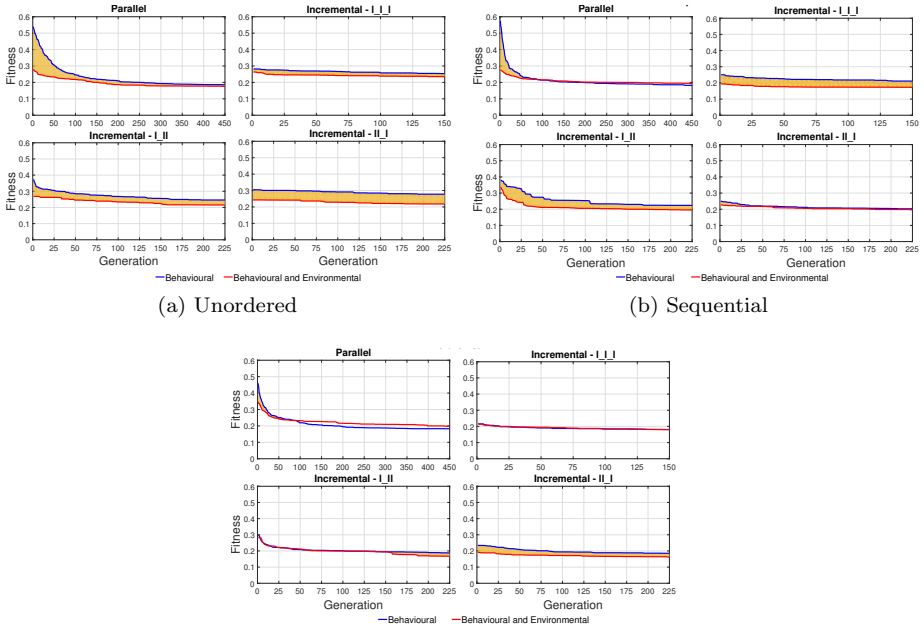
6.6.1. Impact of Environmental Scaffoldings on Performance

As identified in Section 6.4, using environmental scaffoldings helps improve mean performance, although it also slightly reduces the robustness of the algorithms. To further investigate their impact, Figure 12 illustrates the difference in fitness values obtained by the models when environmental scaffoldings are used versus when they are not used for the version of experiments with an unrestricted number of modules [1]. The difference is illustrated after introducing the last environment of the scaffoldings to the models and all three subgoals are being evaluated.

It is evident that a majority of the experiments (nine out of 12) show a statistically significant improvement in performance, when environmental scaffoldings are used (Mann-Whitney U $p < 0.05$). The percentage difference in

Table 6: Statistical Results Summary for the Autonomous Abstraction Architectures with Restricted and Unrestricted Modules

Method	Mann-Whitney U-test (p-value)		
	Unordered	Sequential	Hierarchical
Behavioural			
PL	0.00	0.01	0.01
INC I_I_I	0.13	0.00	0.00
INC I_II	0.01	0.06	0.00
INC II_I	0.28	0.01	0.00
Behavioural and Environmental			
PL	0.00	0.00	0.03
INC I_I_I	0.01	0.00	0.00
INC I_II	0.01	0.00	0.00
INC II_I	0.00	0.00	0.00



(a) Unordered

(b) Sequential

(c) Hierarchical

Figure 12: The difference between fitness values when environmental scaffoldings are used versus when they are not used for the parallel, incremental- I_I_I, incremental - I_II and incremental - II_I methods with unrestricted modules version for the three tasks with unordered, sequential and hierarchical subgoals. The graphs illustrate the fitness progression across the final generations where all three subgoals are being learned within the environment, with respect to the learning architecture used. The coloured regions correspond to an improvement in fitness when environmental scaffoldings are used in comparison to when they are not used. The results are averaged across 30 runs.

performance for each of the three tasks observed are: unordered - 5.46% (parallel), 9.93% (I_I_I), 17.29% (I_II), 32.52% (II_I); sequential - (-)6.60% (parallel), 21.17% (I_I_I), 15.49% (I_II), 3.41% (II_I); and hierarchical - (-)8.65% (parallel), (-)0.02% (I_I_I), 11.49% (I_II), 12.89% (II_I). The task with hierarchical subgoals shows the least impact as only incremental - I_II and incremental - II_I show an improvement in fitness, while parallel and incremental - I_I_I do not show an improvement. Further research into environmental scaffoldings with a higher number of generations and environments may help improve the results and more clearly delineate the impact of this technique on performance.

6.6.2. Analysis of Autonomously Discovered Behaviour Modules

In this section, the variation of the number of autonomously discovered modules and fitness over the generations is analysed within and across populations for experiments with unrestricted modules [1].

To investigate the relationship between the number of modules and fitness,

Figures 13, 14 and 15 represent the distribution variation of the number of behaviour modules with respect to the average fitness of the tasks with unordered, sequential and hierarchical subgoals, respectively. This illustrates how the number of behaviour modules vary across generations with respect to fitness in a typical group of individuals. The fitness averaged across 30 runs is shown in red, and the area is coloured based on the average number of behaviour modules in each run over 1350 generations. It can be observed that in all four methods in all three types of tasks, the first few generations vary within 4-20 modules and gradually start exploring even more modules (up to > 50). The peaks in fitness distribution (corresponding to the introduction of a new environment or a new subgoal fitness component to the fitness function) also correspond to a variation in the proportion of number of behaviour modules. More trenches and crests are observed in other methods in comparison to parallel learning without environmental scaffoldings which show relatively less significant variations. Further, it can also be observed that the experiments using environmental scaffoldings tend to explore higher numbers of behaviour modules than their counterparts that do not use scaffoldings. This is observable in all three types of tasks with scaffoldings, as towards the last generations, relatively more runs consist of higher numbers of modules.

To further understand the nature of the distribution of number of modules, Figures 16, 17 and 18 depict how the behaviour modules vary in a population of individuals with respect to the number of behaviour modules of the individual with the best performance. Based on these three figures, it is evident that the number of behaviour modules in the population generally varies in relation to that of the best individual in the population. The general variation is within $\pm$ 20 modules from the best individual. When environmental scaffoldings are not used, the deviation is much smaller from the best individual compared to when scaffoldings are used in all three tasks. Figures 19, 20, 21 and 22 explain the phenomenon in relation to a single population. The figures represent the distribution of fitness and the number of behaviour modules within the best run for each experiment with parallel and incremental (I_I_I) learning architectures with and without environmental scaffoldings. Based on these figures, it is evident that when the best individual of the population converges towards a number of modules, the rest of the population follows through. Further, the variations in the number of modules within the population is much higher when environmental scaffoldings are used.

In conclusion, the number of behaviour modules explored with the proposed methods vary from 1-60, with the most common numbers being 4-20. Introduction of new environments and fitness components to the fitness function for evaluation impacts the search space of behaviour modules. The use of environmental scaffoldings expands the search space of behaviour modules compared to when they are not used, and the populations generally converge towards the same number of behaviour modules employed by the best individual of the population, similar to how the fitness of the population converges.

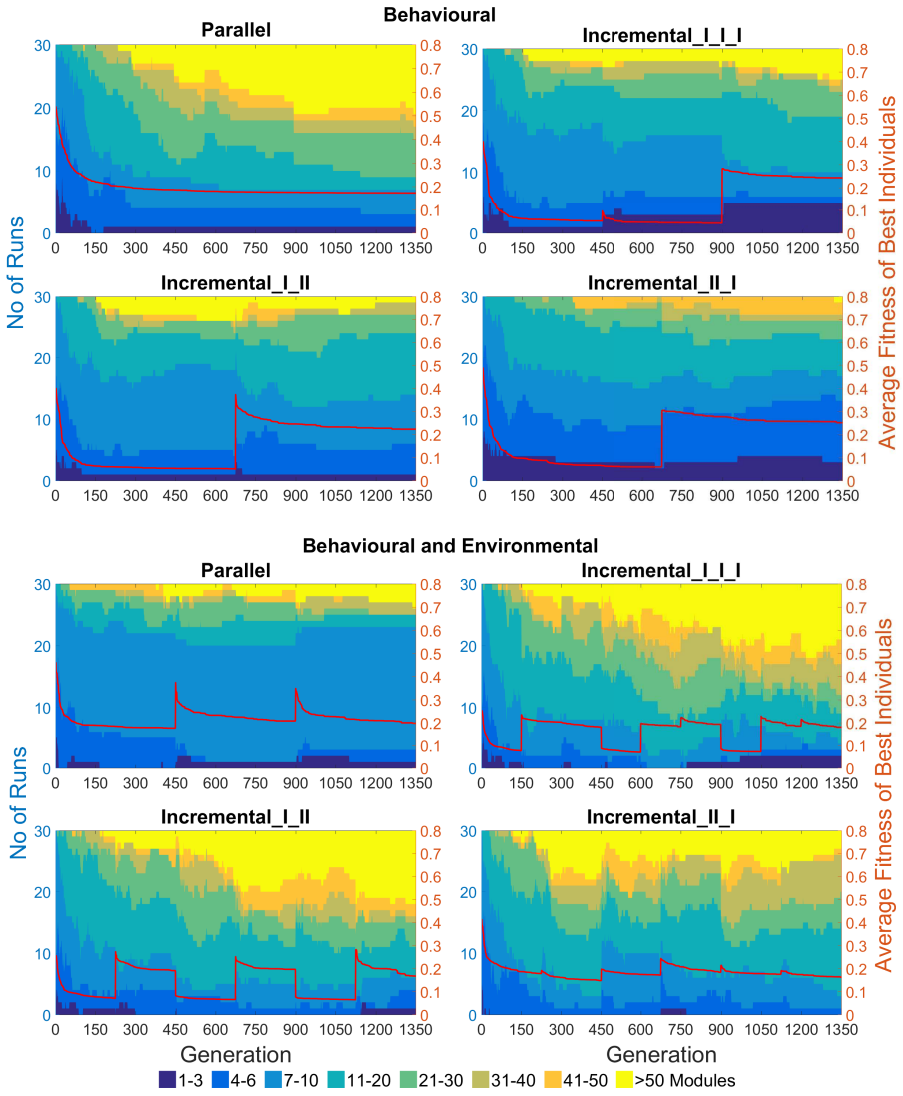


Figure 13: Distribution variation of the number of behaviour modules with respect to fitness in the task with unordered subgoals for all four methods with and without using environmental scaffoldings. The red line plot shows the variation of fitness of the best individual over 1350 generations averaged across 30 runs each. The area graph shows the average number of modules in each run, coloured based on respective range into which the number falls.

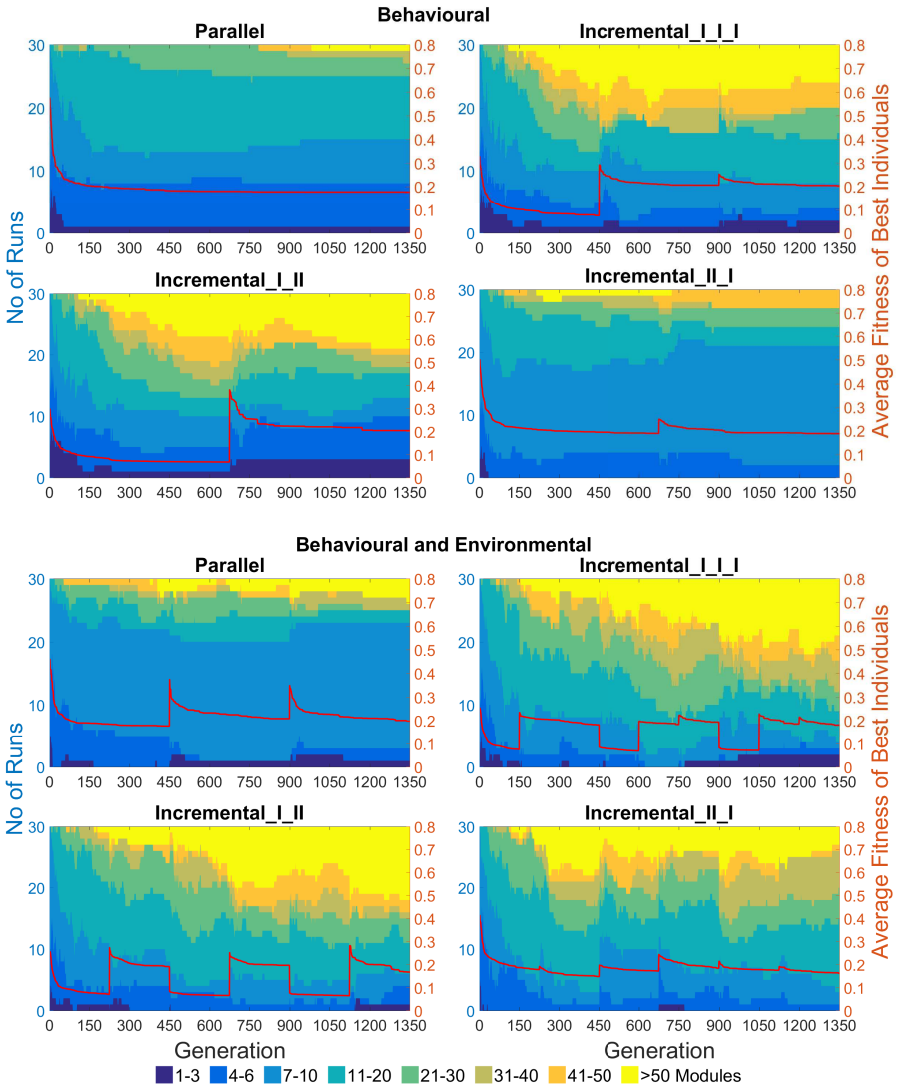


Figure 14: Distribution variation of the number of behaviour modules with respect to fitness in the task with sequential subgoals for all four methods with and without using environmental scaffoldings. The red line plot shows the variation of fitness of the best individual 1350 generations averaged across 30 runs each. The area graph shows the average number of modules in each run, coloured based on respective range into which the number falls.

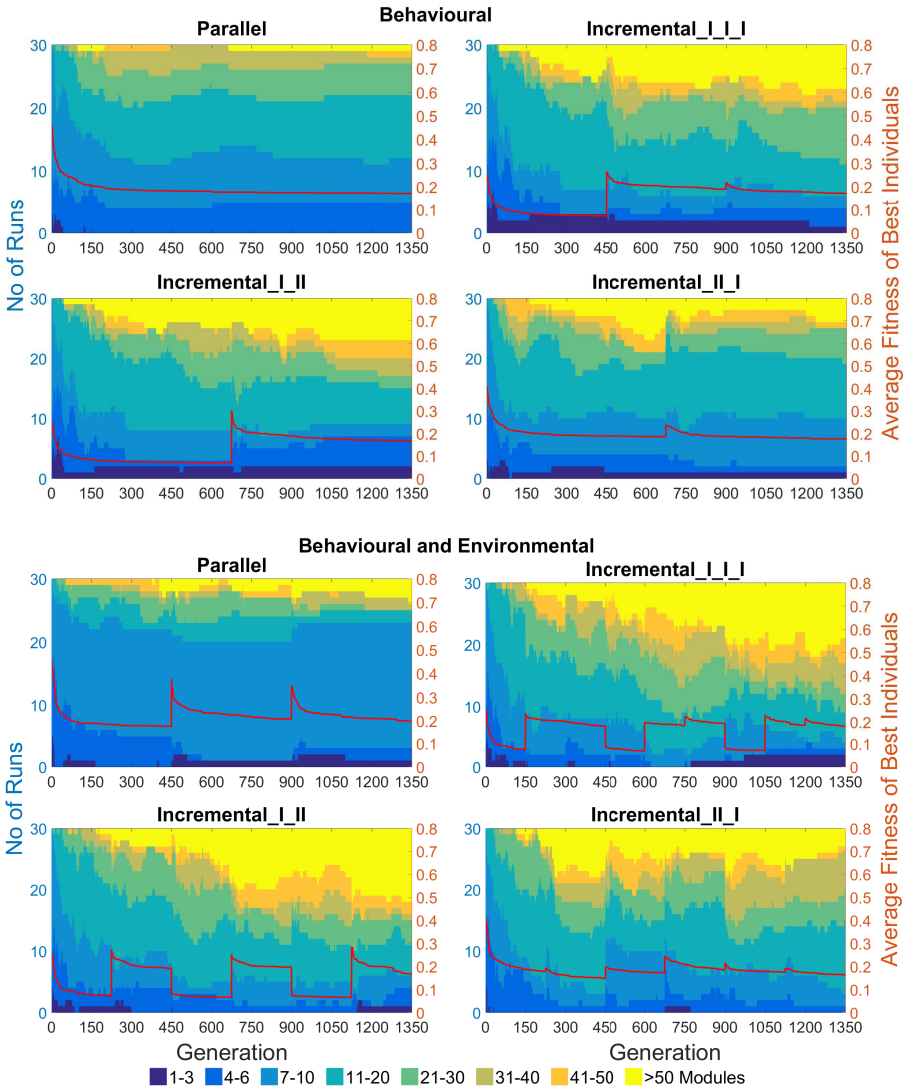


Figure 15: Distribution variation of the number of behaviour modules with respect to fitness in the task with hierarchical subgoals for all four methods with and without using environmental scaffoldings. The red line plot shows the variation of fitness of the best individual over 1350 generations averaged across 30 runs each. The area graph shows the average number of modules in each run, coloured based on respective range into which the number falls.

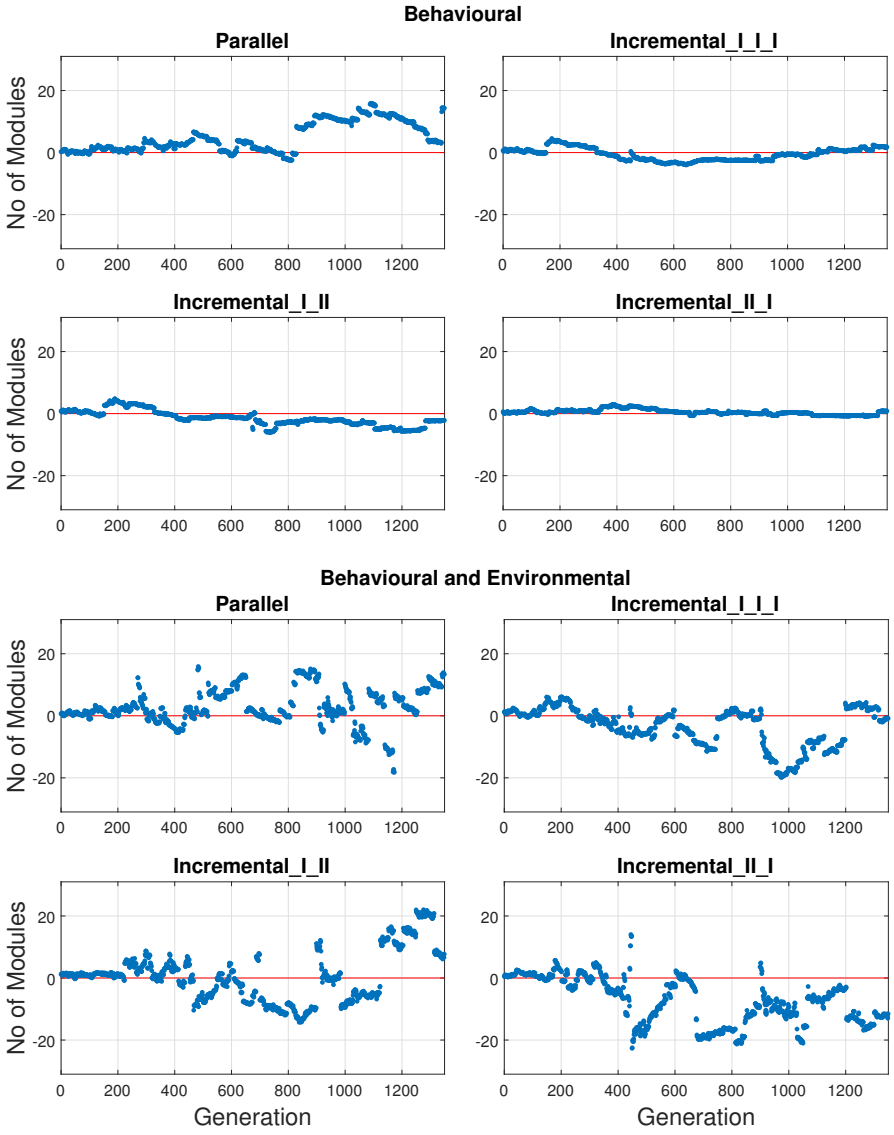


Figure 16: Deviation of number of behaviour modules in the population with respect to the best individual in the task with unordered subgoals. The horizontal red line corresponds to the normalised number of behaviour modules in the best individual averaged across 30 runs. The deviation of the number of modules within the rest of the population averaged across 30 runs is scattered around the best individual through 1350 generations.

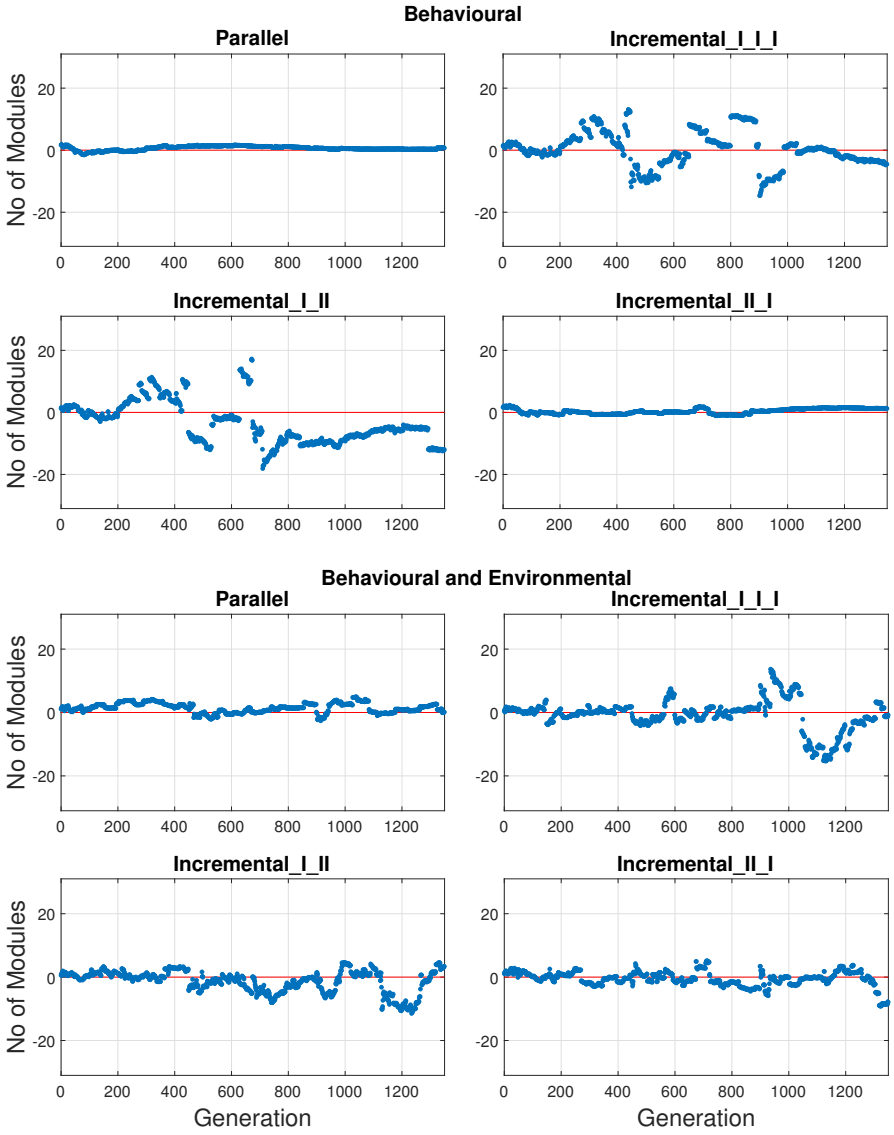


Figure 17: Deviation of number of behaviour modules in the population with respect to the best individual in the task with sequential subgoals. The horizontal red line corresponds to the normalised number of behaviour modules in the best individual averaged across 30 runs. The deviation of the number of modules within the rest of the population averaged across 30 runs is scattered around the best individual through 1350 generations.

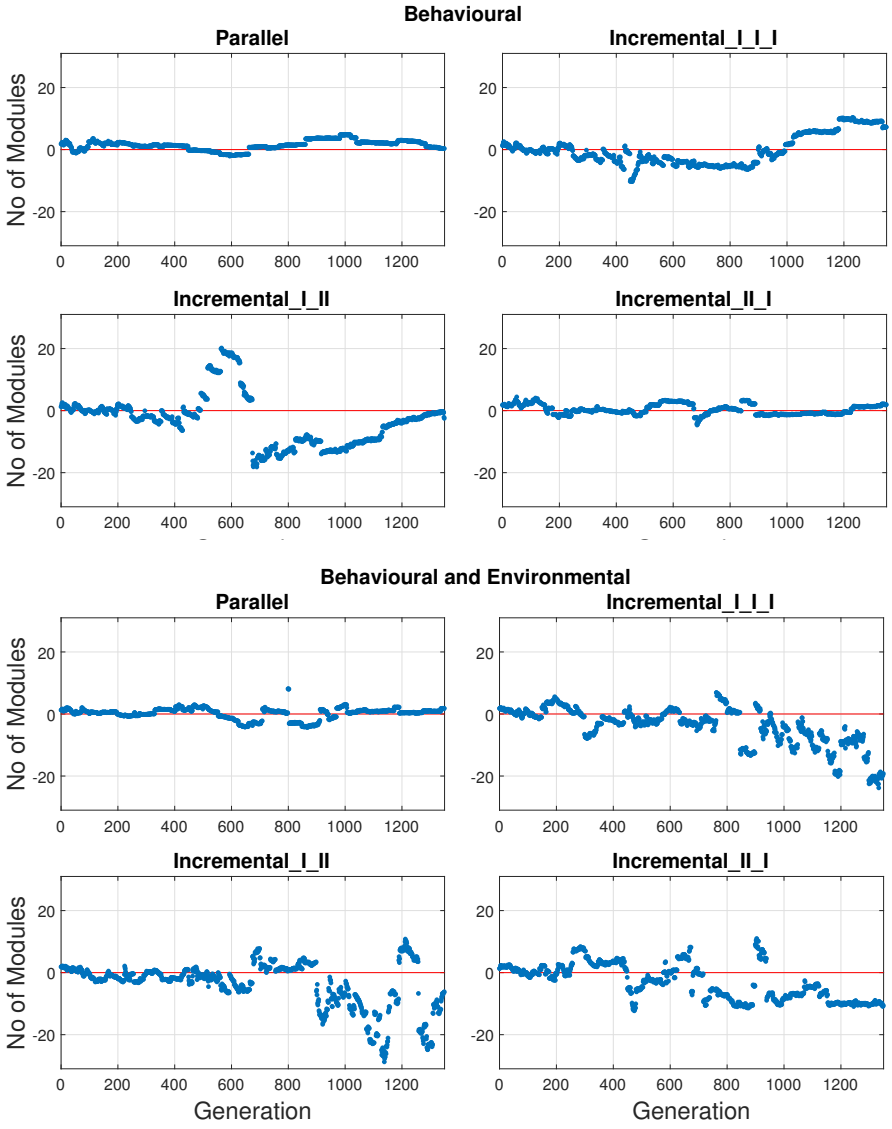
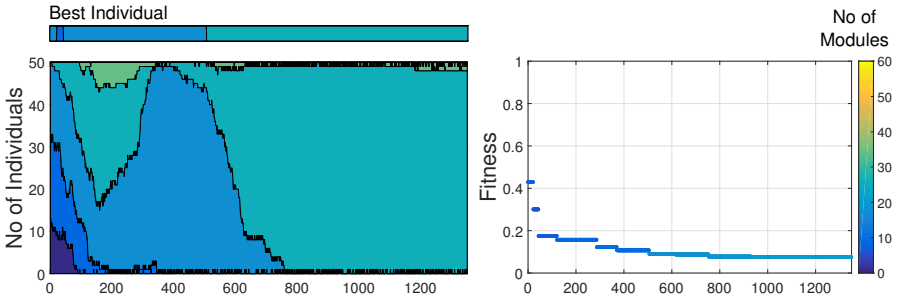
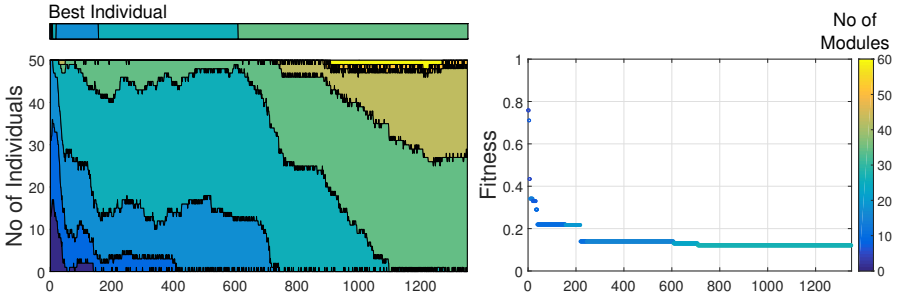


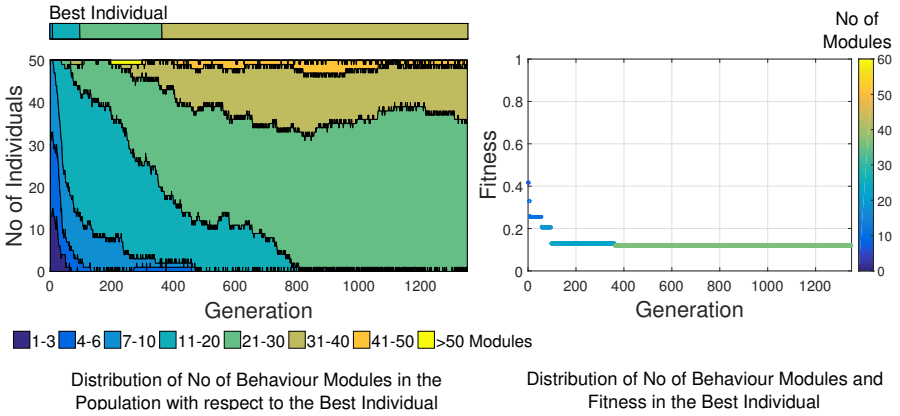
Figure 18: Deviation of number of behaviour modules in the population with respect to the best individual in the task with hierarchical subgoals. The horizontal red line corresponds to the normalised number of behaviour modules in the best individual averaged across 30 runs. The deviation of the number of modules within the rest of the population averaged across 30 runs is scattered around the best individual through 1350 generations.



(a) Parallel without Environmental Scaffoldings : Unordered

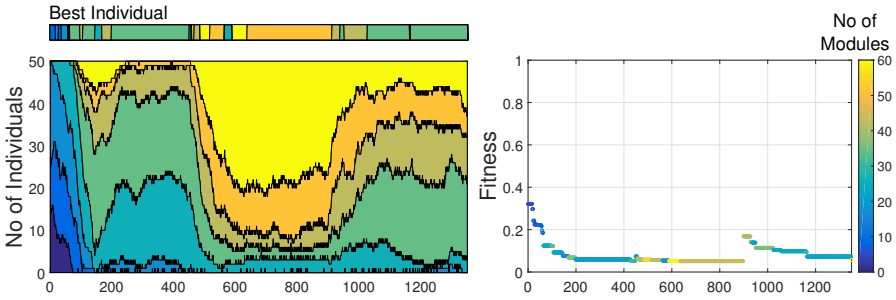


(b) Parallel without Environmental Scaffoldings : Sequential

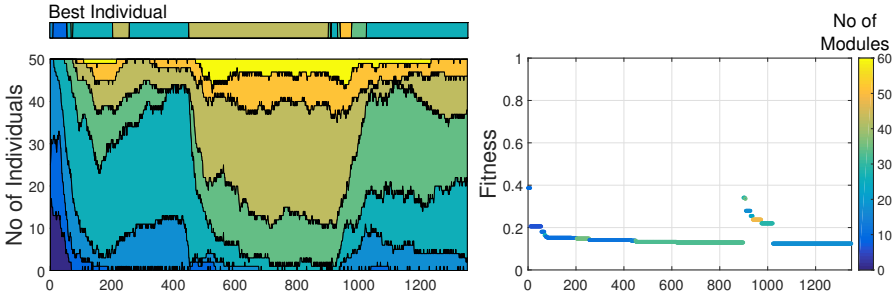


(c) Parallel without Environmental Scaffoldings : Hierarchical

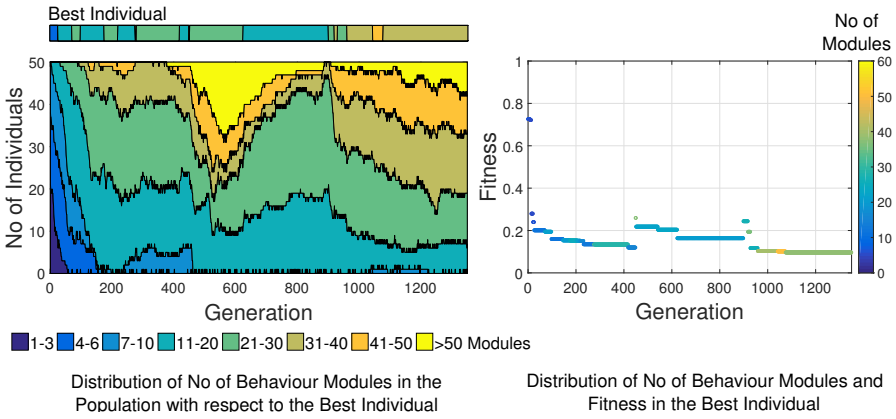
Figure 19: Distribution of the number of behaviour modules and fitness within a population in the best run out of 30 with parallel learning architecture without environmental scaffoldings for the three types of tasks. Left: Variation of the number of behaviour modules of the population during the evolutionary process in relation to the number of modules of the best-performing individual of the population. Right: Distribution of the number of behaviour modules and fitness of the best individual within the population.



(a) Parallel with Environmental Scaffoldings : Unordered

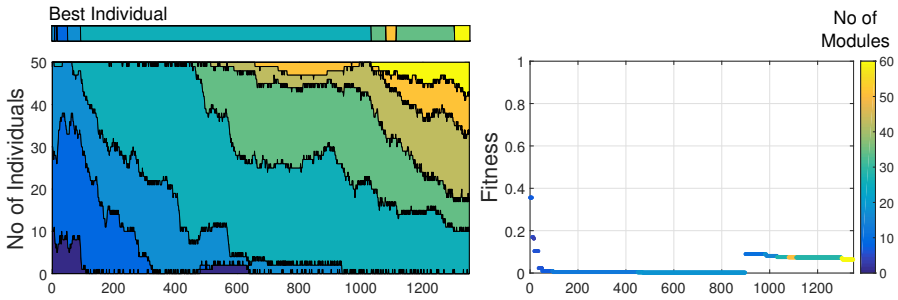


(b) Parallel with Environmental Scaffoldings : Sequential

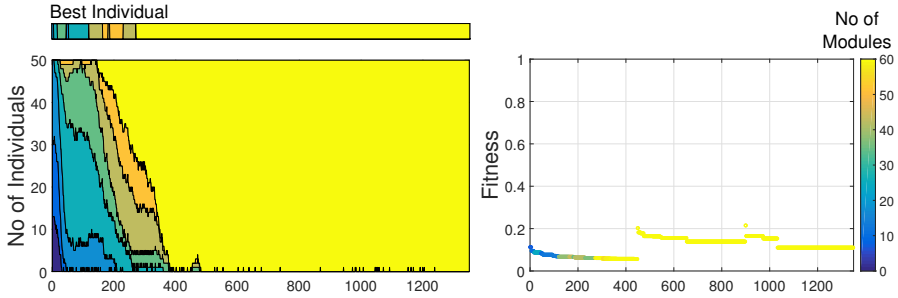


(c) Parallel with Environmental Scaffoldings : Hierarchical

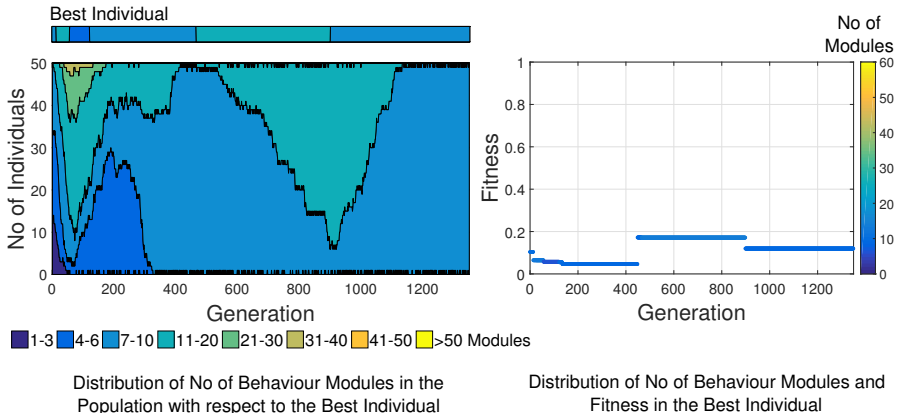
Figure 20: Distribution of the number of behaviour modules and fitness within a population in the best run out of 30 with parallel learning architecture with environmental scaffoldings for the three types of tasks. Left: Variation of the number of behaviour modules of the population during the evolutionary process in relation to the number of modules of the best-performing individual of the population. Right: Distribution of the number of behaviour modules and fitness of the best individual within the population.



(a) Incremental I_I_I without Environmental Scaffoldings : Unordered

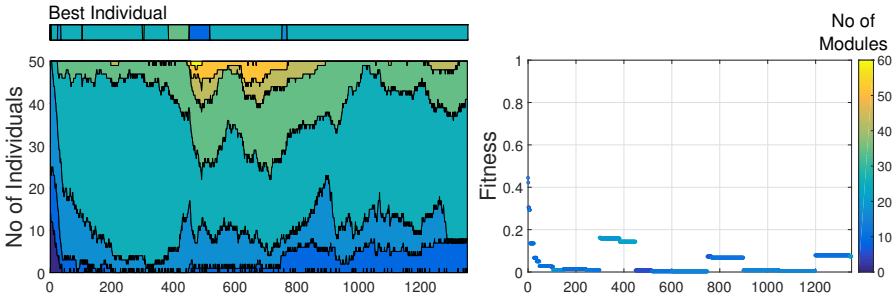


(b) Incremental I_I_I without Environmental Scaffoldings : Sequential

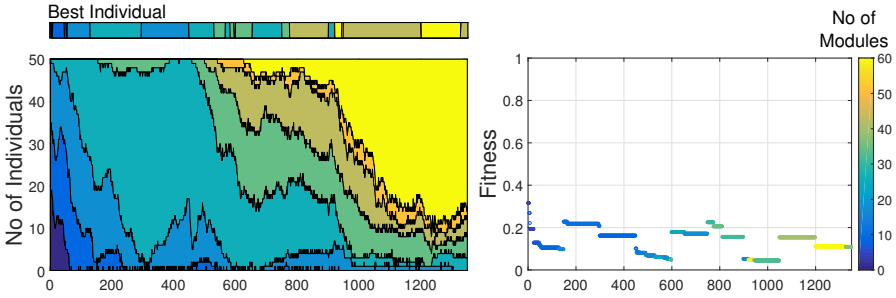


(c) Incremental I_I_I without Environmental Scaffoldings : Hierarchical

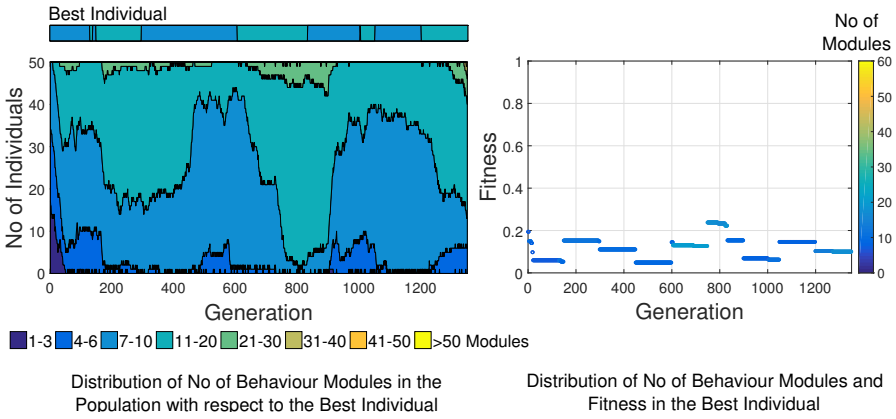
Figure 21: Distribution of the number of behaviour modules and fitness within a population in the best run out of 30 with incremental (I_I_I) learning architecture without environmental scaffoldings for the three types of tasks. Left: Variation of the number of behaviour modules of the population during the evolutionary process in relation to the number of modules of the best-performing individual of the population. Right: Distribution of the number of behaviour modules and fitness of the best individual within the population.



(a) Incremental I_I_I with Environmental Scaffoldings : Unordered



(b) Incremental I_I_I with Environmental Scaffoldings : Sequential



(c) Incremental I_I_I with Environmental Scaffoldings : Hierarchical

Figure 22: Distribution of the number of behaviour modules and fitness within a population in the best run out of 30 with incremental (I_I_I) learning architecture with environmental scaffoldings for the three types of tasks. Left: Variation of the number of behaviour modules of the population during the evolutionary process in relation to the number of modules of the best-performing individual of the population. Right: Distribution of the number of behaviour modules and fitness of the best individual within the population.

6.7. Rule Complexity

The cyclomatic complexity (CC) [51], a common quantitative metric used to determine the number of independent paths through a programme and number of rules is used as a measure to determine the complexity of the evolved rules. Figure 23 illustrates the CC for the models with and without the use of environmental scaffoldings in comparison to the two generic evolutionary models for the experiments using unrestricted modules [1].

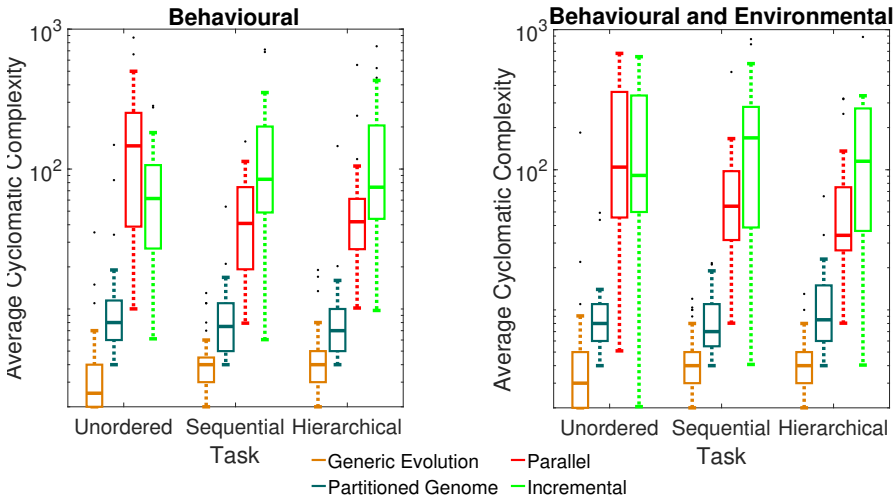
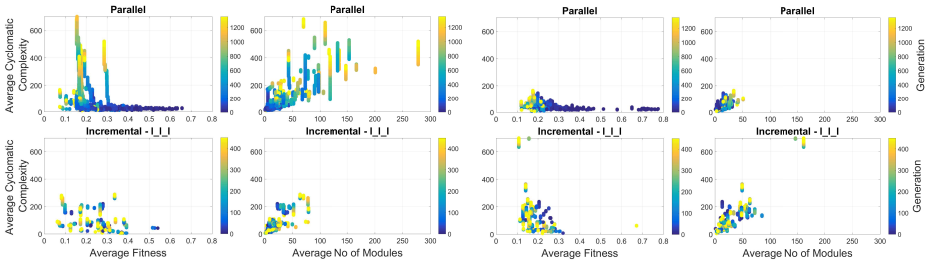


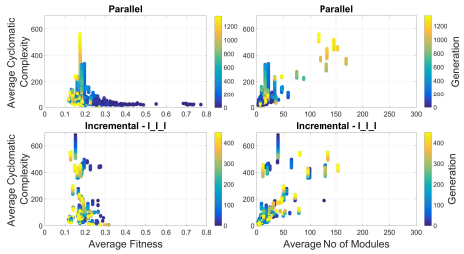
Figure 23: Complexity analysis of the evolved rules with the parallel and incremental architectures with automatically derived modules compared against the generic evolutionary models across 30 runs for the three tasks. The CC of the rules averaged across all agent behaviours for 10 agents is presented. Right: Without using environmental scaffoldings. Left: Using environmental scaffoldings. The bottom edge, central line and top edge indicate the 25th percentile, median result and 75th percentile, respectively. The extensions of whiskers run up to the extreme results not deemed outliers. The same evolutionary attributes (including crossover and mutation rates, the number of generations and number of wraps) used in the abstraction models are used for the compared generic models.

From the comparison results, it is evident that the rule structures derived from the abstraction learning architectures where modules are automatically derived are much more complex than the rest of the architectures. The CC of the generic evolutionary model generally averages to a value less than five. However, the average complexity of the rule structures evolved with architectures using automatically generated behaviour modules typically vary in the range of 100-300 and can go up to around 700. This is a significantly higher figure than the other architectures. However, a significant difference between the complexity of the rule structures evolved with and without the help of environmental scaffoldings cannot be observed from the comparisons (Mann-Whitney U $p > 0.05$).

To further investigate the reasons for evolving complex rule structures with the proposed architectures, the variation of CC is analysed in relation to the



(a) Unordered (b) Sequential

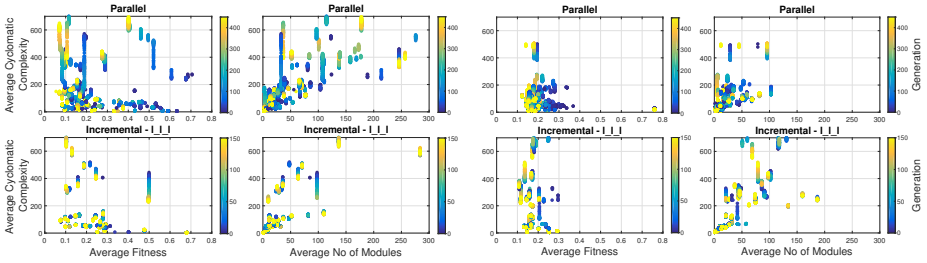


(c) Hierarchical

Figure 24: Variation of CC in relation to the fitness of the rule structures and the number of behaviour modules derived for 30 evolutionary runs with the parallel and incremental architectures without using environmental scaffoldings for the three tasks. The values are averaged across all agents of a group of 10 for each generation. The individual values are colour coded based on the generation.

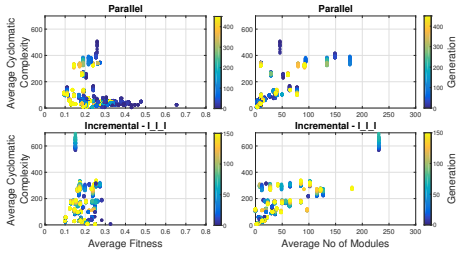
fitness and number of behaviour modules. Figures 24 and 25 depict these variations for both parallel and incremental architectures, with and without environmental scaffoldings. The number of generations depicted in the figures for each architecture are the final generations where the last environment has been introduced (with experiments using environmental scaffoldings) and all three fitness components for the subgoals are actively being evaluated with respect to the learning architecture used.

The results identify no evidence to suggest a correlation between the fitness and the complexity of the rules (-0.23 $<$ Pearson's r $>$ 0.27 for all three tasks with all methods without scaffoldings, -0.15 $<$ Pearson's r $>$ 0.00 for all three tasks with all methods with scaffoldings). The models explore a wide range of rule structures with diverse complexities throughout the evolutionary process in both types of experiments, with and without environmental scaffoldings. However, a correlation exists between the number of modules and the complexity of the rules, as the complexity increases along with an increasing number of modules, which is evident in all three tasks, and more prominent when scaffoldings are not used (Pearson's r $>$ 0.75 for all three tasks with all methods without scaffoldings, Pearson's r $>$ 0.5 for all three tasks with all methods with scaffoldings).



(a) Unordered

(b) Sequential



(c) Hierarchical

Figure 25: Variation of CC in relation to the fitness of the rule structures and the number of behaviour modules derived for 30 evolutionary runs with the parallel and incremental architectures using environmental scaffoldings for the three tasks. The values are averaged across all agents of a group of 10 for each generation. The individual values are colour coded based on the generation.

It is evident from Figure 25 that a wider range of modules, mostly within a higher complexity range, are explored with the models when environmental scaffoldings are used rather than when they are not, as depicted in Figure 24.

7. Discussion and Conclusion

This paper explored a novel mechanism for autonomous discovery of abstractions based on a grammar-based evolutionary approach. In contrast to other multi-agent automation techniques, GE has the unique ability to evolve complete rule structures reducing human intervention in the rule generation process. A combination of this ability and the proposed modifications to the evolutionary model and proposed abstraction learning architectures has shown potential to autonomously identify suitable behaviour modules to learn a complex task in stages solving it with a higher performance.

Parallel and incremental learning architectures were developed to facilitate autonomous derivation of appropriate behaviour modules. Each behaviour module was evolved in a separate genome partition by adopting the proposed genome partitioning technique. To investigate the capacity of the learning architectures to adapt when restrictions are introduced to the abstractions allowed, two techniques were investigated. In one approach, the number of behaviour modules

allowed to address a particular complex task was confined to a maximum of 10. In the next approach, no such restrictions were imposed. The learning architectures were experimented with the use of behavioural decomposition and environmental scaffoldings to investigate their impact on guiding the performance of the models.

The results showed that both restricted and unrestricted versions of abstraction learning architectures perform with significantly higher fitness than the generic evolutionary models that do not use abstractions. However, the unrestricted version is able to achieve an even significant fitness improvement than the restricted version. This suggests that the impact on performance can change based on constraints introduced on the abstractions. Further, both parallel and incremental learning architectures demonstrated equal competency in generating complex behaviours. No relationship between the organisation of subgoals within a task and the type of learning architecture was observed.

The models are more robust when environmental scaffoldings are not used to support the evolution process. However, they can achieve improved fitness with the help of scaffoldings. On further analysis of the scaffolding technique, it was also observed that the evolutionary models tend to explore a wider range of rule structures with diverse and higher complexity levels when scaffoldings are used. The number of behaviour modules derived by the models is primarily guided by the fitness of the individual rules. It was observed that the populations tend to converge towards similar number of behaviour modules, which the individual with the best fitness in the population has.

According to the rule complexity analysis, the rule structures evolved with the autonomous discovery technique of abstractions are much more complex than the generic evolutionary models that do not use abstractions. The complexity is also correlated with the number of behaviour modules, since it was observed that the complexity increases when the number of behaviour modules in the rule structure increases. On the other hand, there exists no correlation between the fitness and the complexity, nor a difference in complexity between the two architectures, incremental and parallel learning.

In conclusion, the proposed abstraction learning architectures exploring autonomous means to discover appropriate abstractions for a given complex task show potential in deriving successful multi-agent behaviours to address dynamic multifaceted problems in the real world. It should also be noted that several limitations and potential for future expansions of the proposed model exist in relation to its application in real-world domains. As discussed above, it was discovered that the model with an unrestricted number of abstractions is more complex than when restrictions are introduced on the number of abstractions that a genome can support through partitions. While the unrestricted version facilitates a higher performance level, it could also be hindered by limitations in computational resources that can be provided in an application environment. As such, the constraints applied on the automatic derivation mechanism can be further investigated by an analysis of more restrictions. Experimenting with more values for the number of maximum modules allowed and different types of constraints such as restricting the rule depth and complexity may provide more

insights on the application of automatic abstractions in real-world domains. It could further enhance the understanding on achieving a balance between complexity and performance that would suit more application domains. Secondly, the fitness measures that guide the evolution are currently derived based on subgoals. Investigating fitness function design techniques independent of subgoals would also strengthen the automatic abstraction learning architectures. Adding another evolutionary layer to generate the fitness functions themselves using a rule space as proposed to determine the appropriate structure of the functions is a possible alternative. Finally, the proposed mechanism is currently limited to homogeneous MASs. However, certain real-world applications may involve heterogeneous systems where agents possess different skills and attributes from one another. There exists potential for future research to expand the proposed model towards more diverse agent systems and task requirements. The proposed architectures can be experimented in heterogeneous as well as hybrid multi-agent architectures in a co-evolutionary environment where agents have diverse morphological and/or behavioural structures. For example, the abstraction learning architectures can be combined with our previously proposed GE model for heterogeneous MASs [49] to investigate the capacity of the model in more dynamic and complex environments. Finally, the presented investigations are conducted with simulations rather than physical robots. There exists potential to expand the research work towards real-world applications through physical agents which could present new research avenues associated with practical limitations of implementation.

References

[1] J. Schmidhuber, A general method for incremental self-improvement and multi-agent learning in unrestricted environments, in: Evolutionary Computation: Theory and Applications. Scientific Publ. Co., Singapore. In, press, 1996, pp. 81–123.

[2] W. H. Hsu, S. M. Gustafson, Genetic programming and multi-agent layered learning by reinforcements, in: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, GECCO'02, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002, p. 764–771.

[3] D. Samarasinghe, M. Barlow, E. Lakshika, K. Kasmarik, Exploiting abstractions for grammar-based learning of complex multi-agent behaviours, International Journal of Intelligent Systems 36 (11) (2021) 6273–6311. doi:https://doi.org/10.1002/int.22550.

[4] H. Tao, J. Li, Y. Chen, V. Stojanovic, H. Yang, Robust point-to-point iterative learning control with trial-varying initial conditions, IET Control Theory & Applications 14 (19) (2020) 3344–3350. doi:https://doi.org/10.1049/iet-cta.2020.0557.

[5] J. L. Elman, Incremental learning, or the importance of starting small, University of California, San Diego, 1991.

[6] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1345–1359. doi:10.1109/TKDE.2009.191.

[7] S. Arimoto, S. Kawamura, F. Miyazaki, Bettering operation of robots by learning, Journal of Robotic Systems 1 (2) (1984) 123–140. doi:https://doi.org/10.1002/rob.4620010203.

[8] T. Peirelinck, H. Kazmi, B. V. Mbuwir, C. Hermans, F. Spiessens, J. Suykens, G. Deconinck, Transfer learning in demand response: A review of algorithms for data-efficient modelling and control, Energy and AI 7 (2022) 100126.

[9] H. Tao, X. Li, W. Paszke, V. Stojanovic, H. Yang, Robust pd-type iterative learning control for discrete systems with multiple time-delays subjected to polytopic uncertainty and restricted frequency-domain, Multidimensional Systems and Signal Processing 32 (2) (2021) 671–692.

[10] R. W. Longman, Iterative learning control and repetitive control for engineering practice, International journal of control 73 (10) (2000) 930–954.

[11] E. A. Mcgovern, A. G. Barto, Autonomous discovery of temporal abstractions from interaction with an environment, Ph.D. thesis, University of Massachusetts at Amherst (2002).

[12] A. L. Christensen, M. Dorigo, Incremental evolution of robot controllers for a highly integrated task, in: S. e. a. Nolfi (Ed.), From Animals to Animats 9, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 473–484.

[13] C. Ryan, J. Collins, M. O. Neill, Grammatical evolution: Evolving programs for an arbitrary language, in: W. Banzhaf, R. Poli, M. Schoenauer, T. C. Fogarty (Eds.), Genetic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 83–96.

[14] O. Buffet, A. Dutech, F. Charpillet, Incremental reinforcement learning for designing multi-agent systems, in: Proceedings of the Fifth International Conference on Autonomous Agents, AGENTS '01, Association for Computing Machinery, New York, NY, USA, 2001, p. 31–32. doi:10.1145/375735.375826.
URL https://doi.org/10.1145/375735.375826

[15] Y. Cai, D. Wu, Y. Zhou, S. Fu, H. Tian, Y. Du, Self-organizing neighborhood-based differential evolution for global optimization, Swarm and Evolutionary Computation 56 (2020) 100699. doi:https://doi.org/10.1016/j.swevo.2020.100699.
URL http://www.sciencedirect.com/science/article/pii/S2210650219303669

[16] J. Xu, F. Zhong, Y. Wang, Learning multi-agent coordination for enhancing target coverage in directional sensor networks, Advances in Neural Information Processing Systems 33 (2020).

[17] A. Ma, M. Ouimet, J. Cortés, Hierarchical reinforcement learning via dynamic subspace search for multi-agent planning, Autonomous Robots 44 (3) (2020) 485–503.

[18] H. T. Nguyen, T. Nguyen, D. Nguyen, T. Le, A hierarchical deep deterministic policy gradients for swarm navigation, in: 2019 11th International Conference on Knowledge and Systems Engineering (KSE), 2019, pp. 1–7. doi:10.1109/KSE.2019.8919269.

[19] M. A. Montes de Oca, T. Stutzle, K. Van den Enden, M. Dorigo, Incremental social learning in particle swarms, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41 (2) (2011) 368–384. doi:10.1109/TSMCB.2010.2055848.

[20] T. Shibata, K. Ohkawa, K. Tanie, Sensor-based behavior using a neural network for incremental learning in family mobile robot system, in: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), Vol. 5, 1994, pp. 2825–2830 vol.5. doi:10.1109/ICNN.1994.374679.

[21] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 41–48. doi:10.1145/1553374.1553380.
URL https://doi.org/10.1145/1553374.1553380

[22] J. Pugh, A. Martinoli, Parallel learning in heterogeneous multi-robot swarms, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 3839–3846. doi:10.1109/CEC.2007.4424971.

[23] Şaban Gülcü, H. Kodaz, A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization, Engineering Applications of Artificial Intelligence 45 (2015) 33 – 45. doi:https://doi.org/10.1016/j.engappai.2015.06.013.
URL http://www.sciencedirect.com/science/article/pii/S0952197615001359

[24] M. A. Mirza, S. Li, L. Jin, Simultaneous learning and control of parallel stewart platforms with unknown parameters, Neurocomputing 266 (2017) 114 – 122. doi:https://doi.org/10.1016/j.neucom.2017.05.026.
URL http://www.sciencedirect.com/science/article/pii/S0925231217308639

[25] Y. Sun, J. Lai, L. Cao, X. Chen, Z. Xu, Y. Xu, A novel multi-agent parallel-critic network architecture for cooperative-competitive reinforcement learning, IEEE Access 8 (2020) 135605–135616. doi:10.1109/ACCESS.2020.3011670.

[26] G. Li, Q. Duan, Y. Shi, A parallel evolutionary algorithm with value decomposition for multi-agent problems, in: Y. Tan, Y. Shi, M. Tuba (Eds.), Advances in Swarm Intelligence, Springer International Publishing, Cham, 2020, pp. 616–627.

[27] R. de Boer, J. Kok, The incremental development of a synthetic multi-agent system: The uva trilearn 2001 robotic soccer simulation team, Ph.D. thesis, Master's thesis, University of Amsterdam, The Netherlands (2002).

[28] A. Stanton, A. Channon, Heterogeneous complexification strategies robustly outperform homogeneous strategies for incremental evolution, in: Artificial Life Conference Proceedings 13, MIT Press, 2013, pp. 973–980.

[29] A. Dutech, O. Buffet, F. Charpillet, Multi-agent systems by incremental gradient reinforcement learning, in: International Joint Conference on Artificial Intelligence, Vol. 17, Citeseer, 2001, pp. 833–838.

[30] A. Graves, M. G. Bellemare, J. Menick, R. Munos, K. Kavukcuoglu, Automated curriculum learning for neural networks, in: Proc. 34th Int. Conf. Machine Learning - Vol. 70, ICML'17, JMLR.org, 2017, p. 1311–1320.

[31] J.-B. Mouret, S. Doncieux, Incremental evolution of animats' behaviors as a multi-objective optimization, in: M. Asada, J. C. T. Hallam, J.-A. Meyer, J. Tani (Eds.), From Animals to Animats 10, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 210–219.

[32] K. Frans, J. Ho, X. Chen, P. Abbeel, J. Schulman, Meta Learning Shared Hierarchies, in: International Conference on Learning Representations, 2018, pp. 1–11.
URL https://openreview.net/forum?id=SyX0IeWAW

[33] D. Jardim, L. Nunes, S. Oliveira, Hierarchical reinforcement learning: Learning sub-goals and state-abstraction, in: 6th Iberian Conf. on Information Systems and Technologies (CISTI 2011), 2011, pp. 1–4.

[34] M. Minsky, Steps toward artificial intelligence, Proceedings of the IRE 49 (1) (1961) 8–30. doi:10.1109/JRPROC.1961.287775.

[35] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.

[36] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Vol. 1, MIT press, 1992.

[37] D. Yan, J. Weng, S. Huang, C. Li, Y. Zhou, H. Su, J. Zhu, Deep reinforcement learning with credit assignment for combinatorial optimization, Pattern Recognition 124 (2022) 108466. doi:https://doi.org/10.1016/j.patcog.2021.108466.
URL https://www.sciencedirect.com/science/article/pii/S0031320321006427

[38] A. Cahill, Catastrophic forgetting in reinforcement-learning environments, Master's thesis, University of Otago (2011).

[39] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Transactions on Evolutionary Computation 5 (4) (2001) 349–358. doi:10.1109/4235.942529.

[40] A. Hallawa, S. Schug, G. Iacca, G. Ascheid, Evolving instinctive behaviour in resource-constrained autonomous agents using grammatical evolution, in: P. A. Castillo, J. L. Jiménez Laredo, F. Fernández de Vega (Eds.), Applications of Evolutionary Computation, Springer International Publishing, Cham, 2020, pp. 369–383.

[41] E. Ferrante, E. Duéñez Guzmán, A. E. Turgut, T. Wenseleers, Geswarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 17–24. doi:10.1145/2463372.2463385.
URL https://doi.org/10.1145/2463372.2463385

[42] P. Urbano, E. Naredo, L. Trujillo, Generalization in maze navigation using grammatical evolution and novelty search, in: A.-H. Dediu, M. Lozano, C. Martín-Vide (Eds.), Theory and Practice of Natural Computing, Springer International Publishing, Cham, 2014, pp. 35–46.

[43] M. Nicolau, D. Perez-Liebana, M. O'Neill, A. Brabazon, Evolutionary behavior tree approaches for navigating platform games, IEEE Transactions on Computational Intelligence and AI in Games 9 (3) (2017) 227–238. doi:10.1109/TCIAIG.2016.2543661.

[44] E. Galván-López, D. Fagan, E. Murphy, J. M. Swafford, A. Agapitos, M. O'Neill, A. Brabazon, Comparing the performance of the evolvable πgrammatical evolution genotype-phenotype map to grammatical evolution in the dynamic ms. pac-man environment, in: IEEE Congress on Evolutionary Computation, 2010, pp. 1–8. doi:10.1109/CEC.2010.5586508.

[45] Z. Zhang, T. Tan, K. Huang, An extended grammar system for learning and recognizing complex visual events, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2) (2011) 240–255. doi:10.1109/TPAMI.2010.60.

[46] A. Neupane, M. A. Goodrich, Learning swarm behaviors using grammatical evolution and behavior trees., in: IJCAI, 2019, pp. 513–520.

[47] D. Perez, M. Nicolau, M. O'Neill, A. Brabazon, Evolving behaviour trees for the mario ai competition using grammatical evolution, in: European Conference on the Applications of Evolutionary Computation, Springer, 2011, pp. 123–132.

[48] D. Samarasinghe, E. Lakshika, M. Barlow, K. Kasmarik, Automatic synthesis of swarm behavioural rules from their atomic components, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, New York, NY, USA, 2018, pp. 133–140. doi:10.1145/3205455.3205546.

[49] D. Samarasinghe, M. Barlow, E. Lakshika, K. Kasmarik, Grammar-based cooperative learning for evolving collective behaviours in multi-agent systems, Swarm and Evolutionary Computation 69 (2022) 101017. doi:https://doi.org/10.1016/j.swevo.2021.101017.
URL https://www.sciencedirect.com/science/article/pii/S2210650221001796

[50] D. Samarasinghe, M. Barlow, E. Lakshika, K. Kasmarik, Task allocation in multi-agent systems with grammar-based evolution, in: Proceedings of the 21st ACM International Conference on Intelligent Virtual Agents, Association for Computing Machinery, New York, NY, USA, 2021, p. 175–182.

[51] T. McCabe, A complexity measure, IEEE Transactions on Software Engineering SE-2 (4) (1976) 308–320. doi:10.1109/TSE.1976.233837.