1 Experiment Detials

1.1 Platform

All the experiments are done in Ubuntu 16.04, Python3.6 and PonyGE2. The computation time is measured on a platform with Intel(R) Xeon(R) CPU, 16 cores, 2.10GHz.

1.2 Settings of House Cleaning Scenario

Component-Action Mapping

The component-action mapping in the house cleaning scenario is as follows:

Components	Actions
wheel2	Cross(spills,wheel2) , Wade(puddles,wheel2)
wheel3	Cross(spills,wheel3) , Wade(puddles,wheel3), RunOver(toys,wheel3)
scoop	Sweep(toys,scoop)

Table 1: The component-action mapping in the house cleaning scenario.

$wheel_1$ has the fastest speed but can not cross any obstacle. $wheel_2$ operates fine in wet condition $spills$ and $puddles$ but fails with $toys$. $wheel_3$ can overcome all these obstacles, but its speed is slow. $scoop$ enables the agent to sweep $toys$ in front of it to keep moving.

Action Effects

The action effects in the house cleaning scenario are as follows:

Action	Pre	Add	Del
Cross(spills,wheel2) Cross(spills,wheel3)	At(yard)	At(lounge)	At(yard)
Wade(puddles,wheel2) Wade(puddles,wheel3)	At(yard)	At(playroom)	At(yard)
RunOver(toys,wheel3)	At(lounge)	At(playroom)	At(lounge)
Sweep(toys,scoop)	At(lounge)	At(playroom) Sweeped(toys)	At(lounge)

Table 2: The action effects in the house cleaning scenario.

Joint Fitness Function

The joint fitness of $\mathcal{B}$ and $\mathcal{T}$ is the negative number of the total execution time. If a body-BT pair $\langle \mathcal{B}, \mathcal{T} \rangle$ can not finish the task, its fitness is set to be $-\infty$. Although motors do not provide any executable actions, they influence the executing effects of other components. $motor_1$ is light with limited power while $motor_2$ is heavy but has higher power. Table 3 shows how these two motors affect the execution time of each action.

Act	motor1	motor2
Cross(spills,wheel2)	-2	-1
Wade(puddles,wheel2)	-9	-7
Cross(spills,wheel3)	-	-4
Wade(puddles,wheel3)	-	-12
RunOver(toys,wheel3)	-	-3
Sweep(toys,scoop)	-4	-

Table 3: The negative number of execution time of each action under two different motors.

As can be seen from the above, the house cleaning scene we designed is relatively complex. In fact, it is close to the real life. We believe this scenario can be a good example to show the utility of component-action mapping as well as the modeling of componentized bodies and modular behavior for efficiently solving embodied intelligence problems.

Parameters	Values
Codon Size	2000000
Crossover	variable_onepoint
Crossover Probability	0.9
Max Genome Length	10000
Initialisation	uniform_tree
Mutation	int_flip_per_codon
Mutation Probability	0.1
Replacement	generational
Selection	truncation

Table 4: GE parameters for both body and BT evolution.

1.3 GE Parameters

Due to the randomness of grammar evolution process, all experiments in the paper were repeated 100 times, with the seed from 0 to 99. In all experiments, the GE parameters for both body and BT evolution are the same (Table 4).

Typically, HGE (General) uses a general BT grammar in the following forms:

$$
\begin{aligned}
&\langle s\rangle ::= \langle sequence\rangle \mid \langle selector\rangle\\
&\langle sequence\rangle ::= Sequence(\langle execution\rangle) \mid \langle s\rangle\langle s\rangle \mid \langle sequence\rangle\langle s\rangle\\
&\langle selector\rangle ::= \langle execution\rangle \mid \langle s\rangle\langle s\rangle \mid \langle selector\rangle\langle s\rangle\\
&\langle execution\rangle ::= \langle conditions\rangle\langle action\rangle\\
&\langle conditions\rangle ::= \langle condition\rangle\langle conditions\rangle \mid \langle condition\rangle\\
&\langle condition\rangle ::= c_1 \mid c_2 \mid ... \mid c_n\\
&\langle action\rangle ::= a_1 \mid a_2 \mid ... \mid a_m
\end{aligned}
$$

This a recursive grammar which leads to the infinite search space. So we need to limit the max tree depth in the production process (Table 5).

Parameters	Values
Max Init Tree Depth	10
Max Tree Depth	30

Table 5: Max tree depth limitation for HGE (General).

2 Application Domains

We believe the modeling of componentized body and modular behaviors can be applied to many interesting scenarios in the real world:

- Item Collection. Imagine that you are going on hiking. What should you bring to your experience or safety? In this scenario, you and the things you bring together form an embodied agent. Different items, such as the lighter and the pocket knife, can be used to perform different actions. So a component-action mapping can be established.
- Card Games. In a collectible card game, the cards are components. Each card usually has only one action: play this card. The players need to construct a card pile first under the limit of the number of cards, and then take a sensible strategy to play this deck to beat opponents. In this scenario, a card pile is a componentized body without considering the structure of the components, and your strategies are modular behaviors.
- Multi-agent Grouping. Broadly speaking, a group of agents can be seen as a super embodied agent. In this case, each agent is a component, and its behaviors are part of the group behaviors, which shows modularity.

In the above scenarios, we can use HGE to search for the optimal body-behavior pair, showing the bright application prospects of our methods.