Effects of State and Action Abstraction on Development of Controllers for Concurrent, Interfering, Non-Episodic Tasks

Brent E. Eskridge

Robotics, Evolution, Adaptation, and Learning Lab
School of Computer Science
University of Oklahoma

May 4, 2008

Outline

1. Summary of motivations & problem domain
2. Background of architecture & existing RL techniques
3. Summary of contributions of this thesis
4. Overview of experimental results
5. Discussion & analysis of results
6. Conclusions & future work

Summary of motivations & problem domain

Motivations

- Develop controllers for autonomous agents
- Authentic agent problems

→ Complex tasks

- Authentic solutions

→ Combination of techniques to solve

Motivations

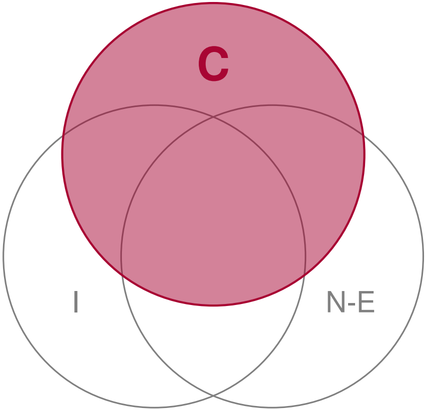
- Develop controllers for autonomous agents
- Authentic agent problems

→ Complex tasks

- Authentic solutions

→ Combination of techniques to solve

Complex CINE Tasks



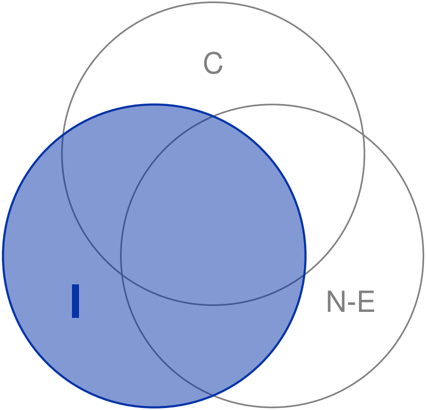
CINE

- Concurrent
- Interfering
- Non-Episodic

Details

Multiple tasks actively being addressed

Complex CINE Tasks



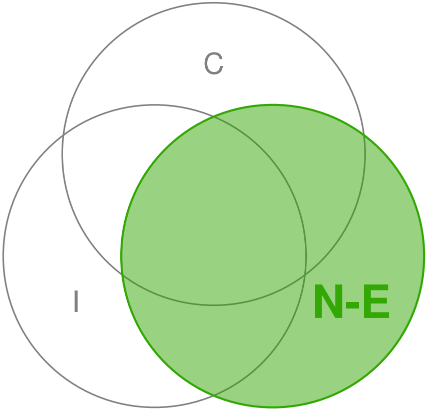
CINE

- Concurrent
- Interfering
- Non-Episodic

Details

Tasks have competing goals and share the same action space

Complex CINE Tasks



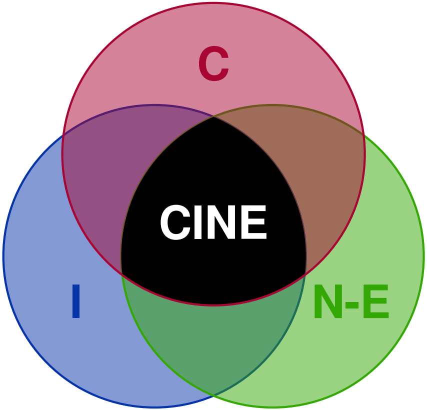
CINE

- Concurrent
- Interfering
- Non-Episodic

Details

Tasks do not terminate and are always active

Complex CINE Tasks



CINE

- Concurrent
- Interfering
- Non-Episodic

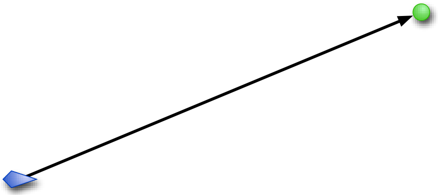
Details

Tasks in the intersection are the most difficult

Complex CINE Tasks: Examples

Active Tasks

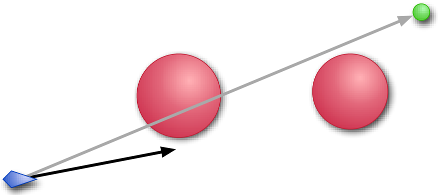
- GoalSeek
- CollisionAvoidance
- RunAway
- Flocking
  - Alignment
  - Cohesion
  - Separation



Complex CINE Tasks: Examples

Active Tasks

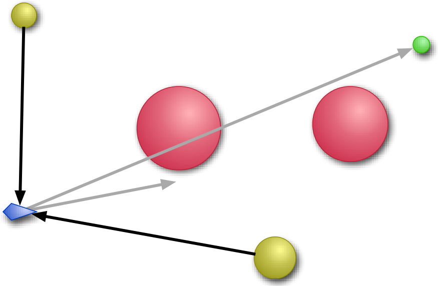
- GoalSeek
- CollisionAvoidance
- RunAway
- Flocking
  - Alignment
  - Cohesion
  - Separation



Complex CINE Tasks: Examples

Active Tasks

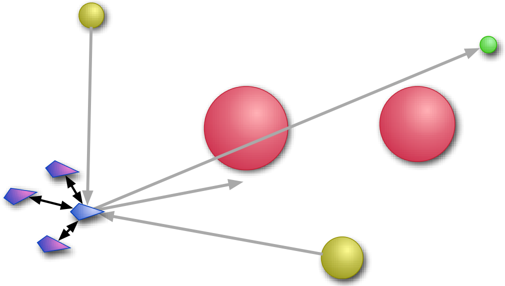
- GoalSeek
- CollisionAvoidance
- RunAway
- Flocking
  - Alignment
  - Cohesion
  - Separation



Complex CINE Tasks: Examples

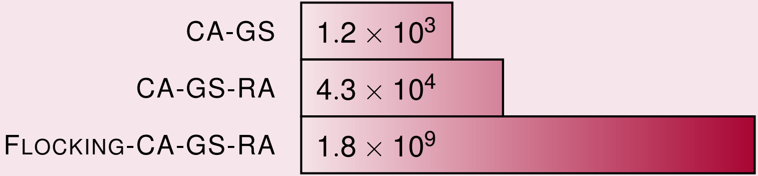
Active Tasks

- GoalSeek
- CollisionAvoidance
- RunAway
- Flocking
  - Alignment
  - Cohesion
  - Separation



Research Motivations

Log comparison of state space sizes



- Developing controllers for these tasks is difficult
- Need to make development of controllers practical
- State and action abstraction can help, but
- What are the benefits/costs of abstraction?

Experimental questions

Given a composite task in which the subtasks are, in general, concurrent, interfering, and non-episodic (CINE),

1. what are the effects of abstracting state information on the performance and development rate of controllers, and
2. what are the effects of abstracting actions on the performance and development rate of controllers?

Why Is This Significant?

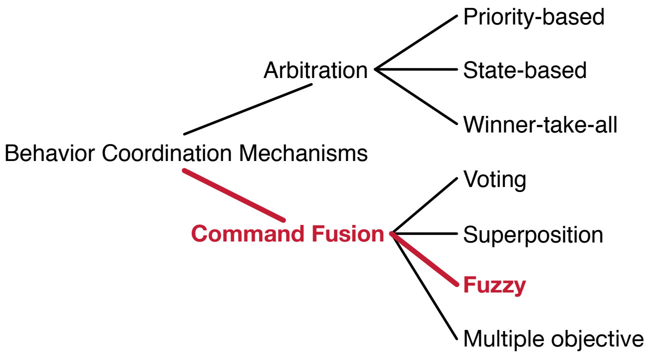
- Answering these questions is difficult
- Can't just perform an experiment
- Existing techniques are insufficient
- To even answer the question, we need:
  - An effective architecture
  - An effective RL approach
  - Set of standard state abstractions

Major Contributions

- Investigated effects of state and action abstraction in the development of controllers for complex CINE tasks
- Demonstrated action abstraction is more beneficial than state abstraction
- Abstracted state into an adaptive, dynamic priority
- Extended adaptive fuzzy behavior hierarchies to allow for more complex hierarchies
- Developed composite reinforcement learning
- Reused existing behaviors in new composite tasks without modification

Background of architecture & existing RL techniques

Behavior-Based Command Fusion

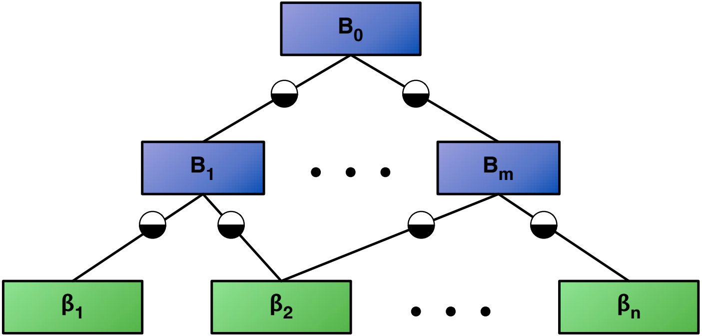


- Associate a behavior with each primitive task
- Use fuzzy command fusion for coordination
- Focus only on reactive tasks — no planning

Adaptive Fuzzy Behavior Hierarchies

- Organize behaviors into hierarchy
- Separate high-level coordination from low-level control
- High-level coordination abstracts action space
- No need for full state space in coordination
- State abstraction is now useful

Hierarchical Decomposition of Behavior



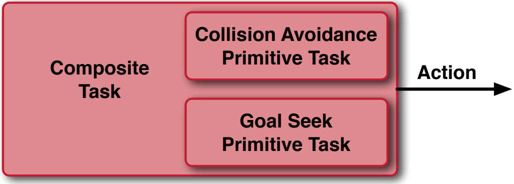
Primitive behaviors produce low-level actions

Composite behaviors modulate lower-level behaviors

Applicable RL Techniques

- Most RL techniques have task restrictions
- Such as sequential or non-interfering
- Restrictions exclude CINE tasks
- Two known techniques can be used:
  1. Standard monolithic RL
  2. Modular RL

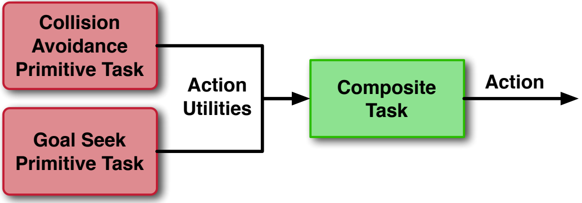
Monolithic RL



Implementation details

- No abstraction used
- Joint state space of all primitive tasks
- Low-level control action space
- Does not scale well!

Modular RL



Implementation details

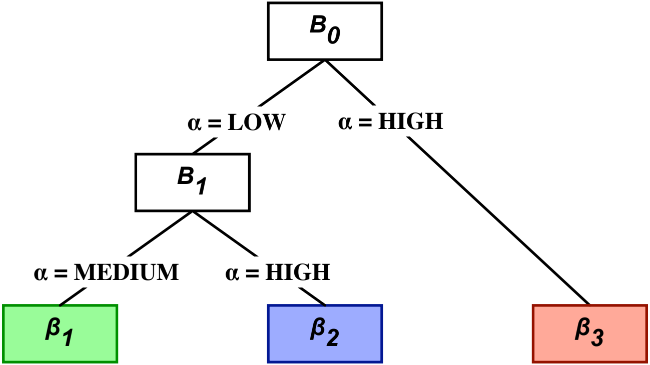
- Designed for concurrent & interfering tasks
- Q-values for primitive tasks are separate
- Action with largest sum of Q-values is taken
- Q-values for each task are learned simultaneously
- Q-values biased towards current composite task

Summary of contributions of this thesis

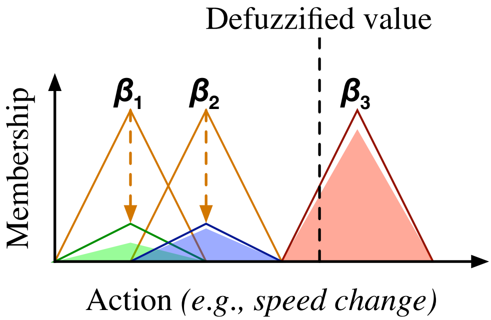
Extending Adaptive Fuzzy Behavior Hierarchies

- Original implementation is effective, but
- Limited to a single composite behavior
- Multiple levels of modulation don't work
- Extended modulation to work as expected
- Can now use with arbitrarily deep hierarchy

Sample Three-level Hierarchy



Expected Defuzzification

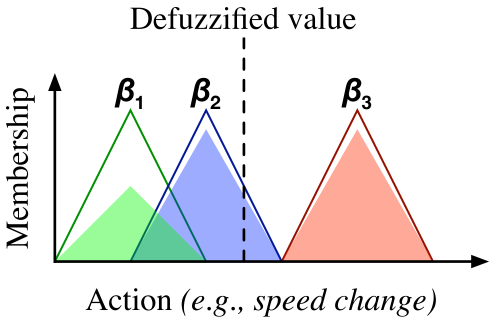


β_1 is a combination of `LOW` and `MEDIUM`

β_2 is a combination of `LOW` and `HIGH`

β_3 is `HIGH`

Actual Defuzzification



β_1 is `MEDIUM`

β_2 is `HIGH`

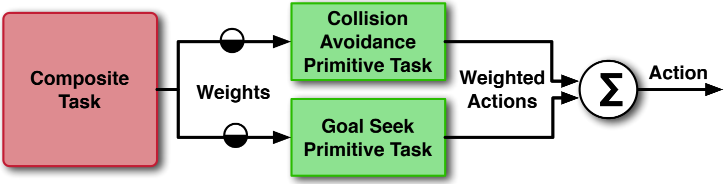
β_3 is `HIGH`

Extension details

Adaptive, Dynamic Priorities

- How far can we push state abstraction in coordination?
- Minimum per primitive task is a single value
- Can be calculated using task state information
- Behavior's opinion of its current priority
- Same or better performance than other abstractions
  - One exception will be discussed

Composite Reinforcement Learning



Implementation details

- Reuses existing behaviors
- Only learns to effectively coordinate (or modulate) lower behaviors
- Limited by effectiveness of lower behaviors, but
- Results show it significantly outperforms alternatives

Behavior Reuse

- Would like to reuse behaviors once developed
- Best if done without modification
- Could significantly reduce development effort
- Composite RL can reuse behaviors
- Other approaches can not

Investigation of Abstraction Effects

- What are effects of abstraction in CINE tasks?
- No known investigation detailing effects
- Experiments investigate effects of abstraction:
  - State
  - Action
- Compare effects for single and multi-agent tasks
- Tasks range from simple to complex

State Abstraction

- Composite behaviors do not produce motor control actions
- Candidates for state abstraction
- Can abstract state to varying degrees
- Used 5 abstraction levels:
  - **Full**
  - **Large**
  - **Medium**
  - **Small**
  - **Minimal**
- Compared effects for same task

State Abstraction

- Composite behaviors do not produce motor control actions
- Candidates for state abstraction
- Can abstract state to varying degrees
- Used 5 abstraction levels:
  - **Full**
  - **Large**
  - **Medium** (3D only)
  - **Small**
  - **Minimal** (Deep hierarchies only)
- Compared effects for same task

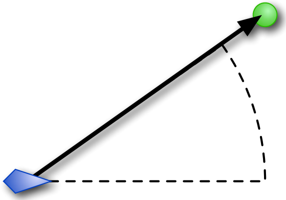
Abstraction Levels

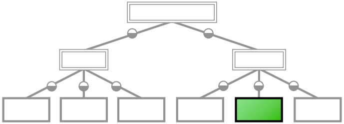
Full

Arrival Time
Theta Direction
Phi Direction

$-180 \le \text{theta} \le 180$

$90 \le \text{phi} \le 90$

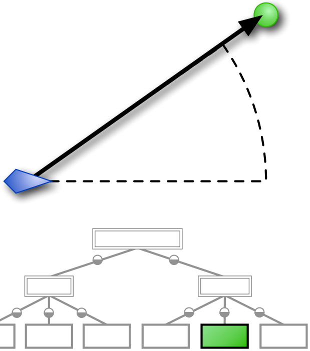




Abstraction Levels

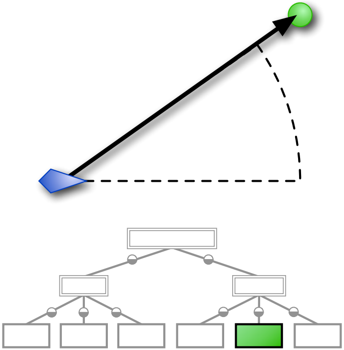
|theta| ≤ 180

|phi| ≤ 90



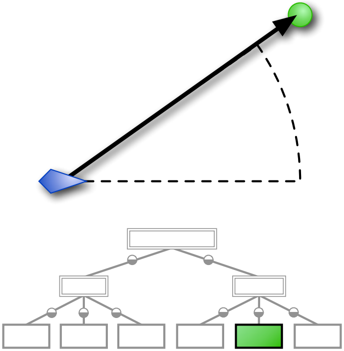
Abstraction Levels

max(| theta |, | phi |)

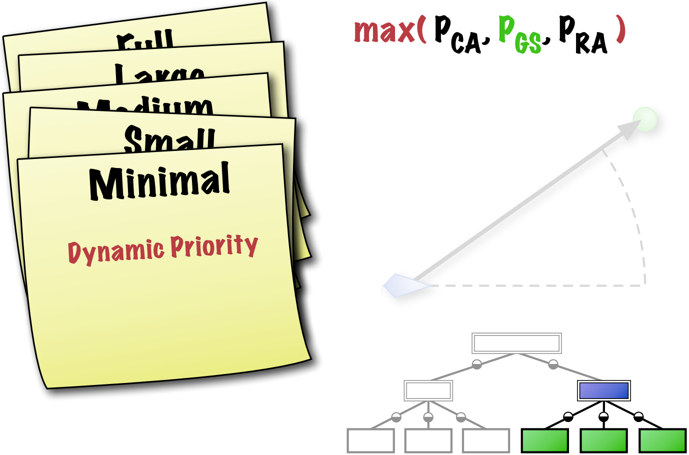


Abstraction Levels

P_{GS}(time, theta, phi)

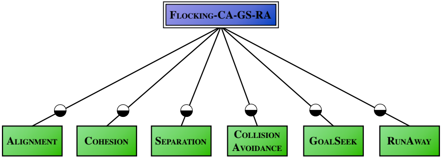


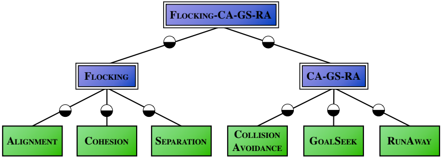
Abstraction Levels



Action Abstraction

- Can abstract actions using composite behaviors
- Deeper hierarchies offer more abstraction
- Fewer, more abstract actions
- Allows for reuse of composite behaviors



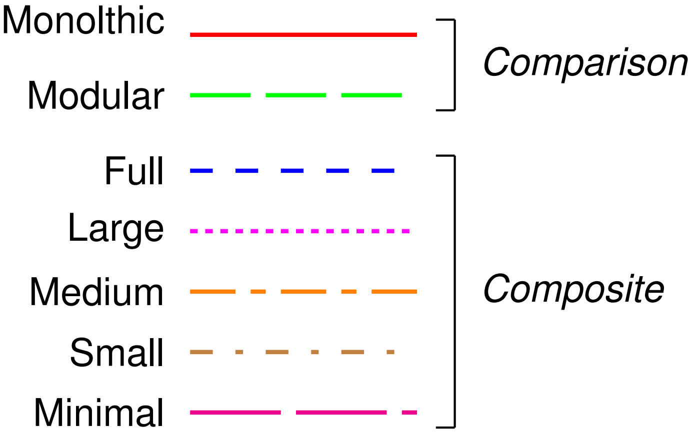


State vs. Action Abstraction

- Directly compared state and action abstraction
- Both make development of controllers more practical
- Action abstraction is comparatively more beneficial
- In complex tasks, action abstraction is essential
- Approaches not using it were ineffective

Overview of Experimental Results

RL Results Legend

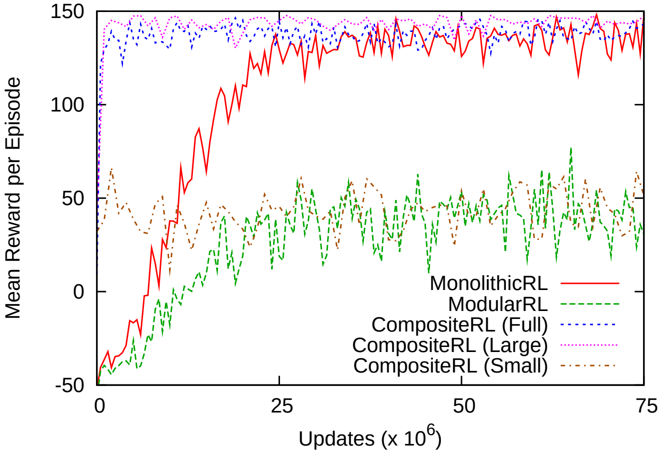


Single-Agent Tasks

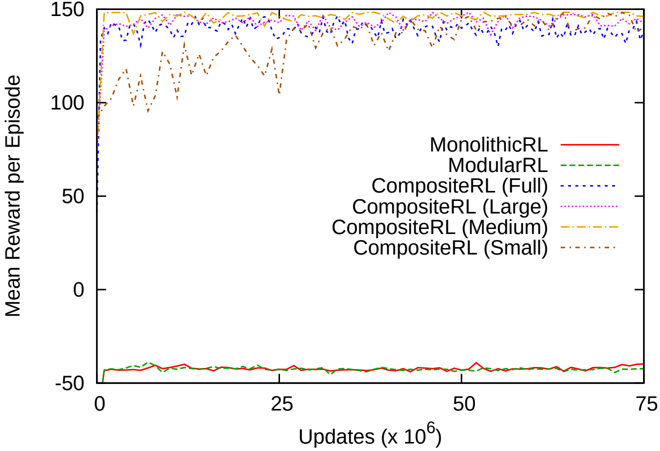
- 2 single agent tasks used:
  - CA-GS
  - CA-GS-RA
- Existing techniques could be effective
- Experiments using 2D and 3D

Task hierarchy
Task hierarchy

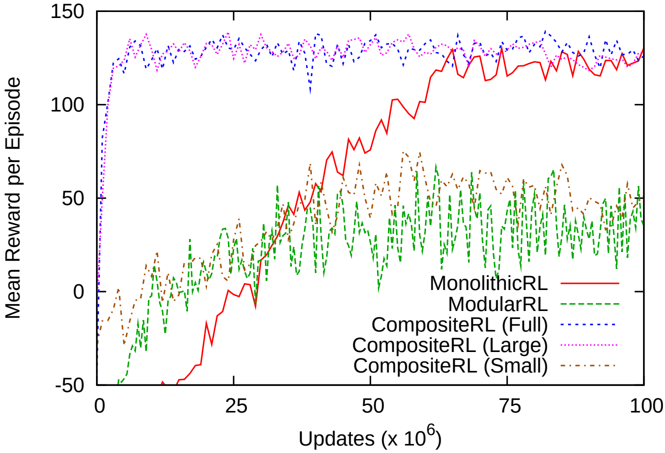
CAGS 2D RL Results



CAGS 3D RL Results



CAGSRA 2D RL Results

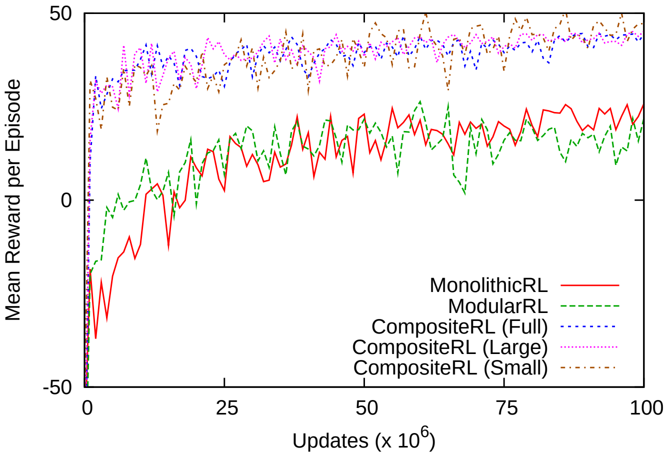


Multi-Agent Tasks

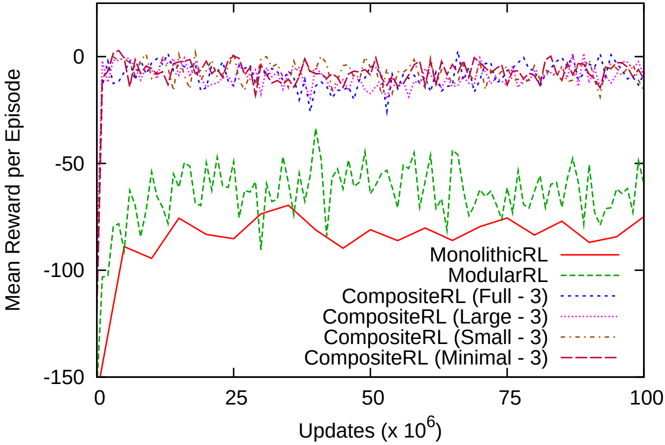
- 4 multi-agent tasks used:
  - FLOCKING
  - FLOCKING-CA
  - FLOCKING-CA-GS
  - FLOCKING-CA-GS-RA
- Push existing techniques to their limits
- Considerably more time to evaluate
- Much more storage required
- Experiments mainly in 2D

Task hierarchy
Task hierarchy
Task hierarchy
Task hierarchy

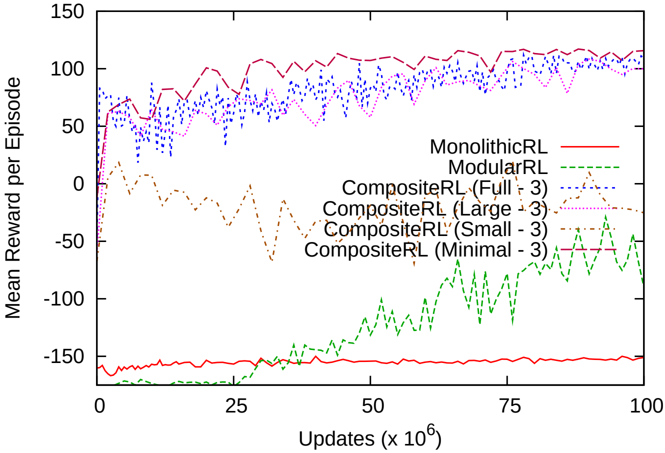
Flocking 2D RL Results



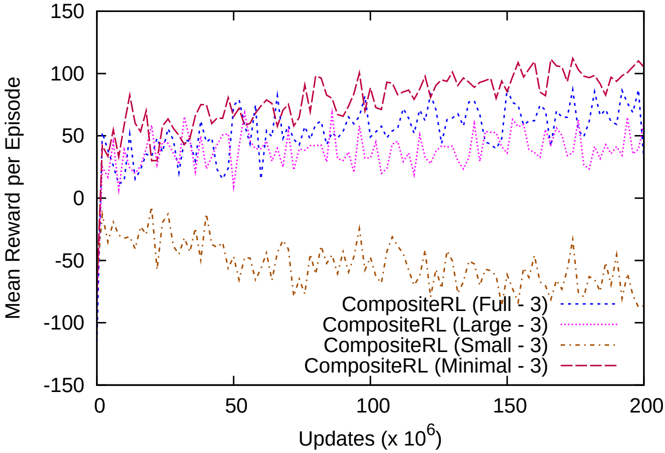
Flocking-CA 2D RL Results



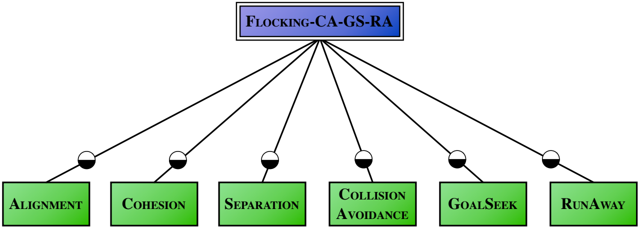
Flocking-CA-GS 2D RL Results



FLOCKING-CA-GS-RA 2D RL Results



Results For 2-Level Hierarchies



Where are they?

- Tried to use 2-level hierarchies
- Too many potential actions to evaluate agent
- 4× longer to evaluate
- Still didn't perform as well

Discussion & analysis of results

State and Action Abstraction

- In most tasks, state abstraction had little effect
- Using a hierarchy introduces action abstraction
  - Performed better even with larger action space
  - In most tasks, almost immediate success
- Action abstraction was essential in complex tasks
- 2-level hierarchies weren't effective or practical in complex tasks

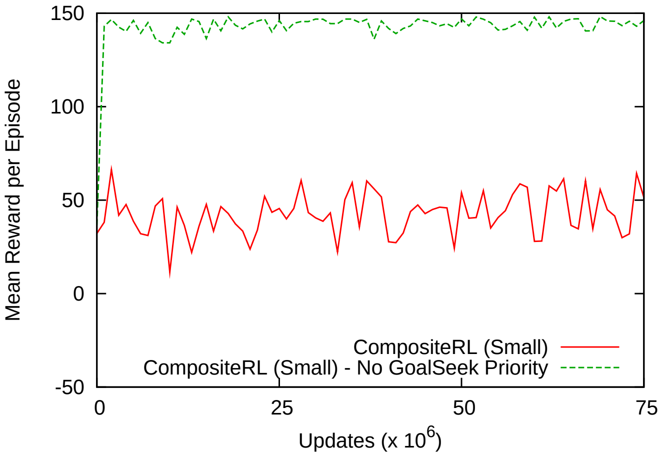
Adaptive, Dynamic Priorities

- Significantly reduces state size, but
- Dynamic priorities don't harm performance
  - GoalSeek priority is the exception
- In complex tasks, dynamic priorities have better performance
- Dependent on effective priority calculation

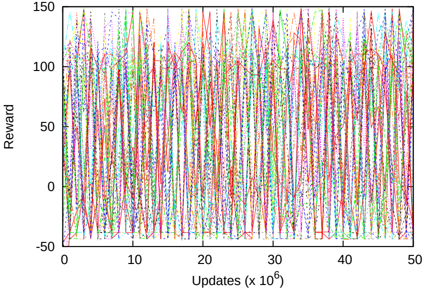
Small Abstraction Level

- **Small** abstraction level often performed poorly in RL
- Only happened with GoalSeek
- Tried using **Small** *without* GoalSeek priority
  - Replaced with full GoalSeek state information
- Significantly better performance
- Performance consistent with other RL results

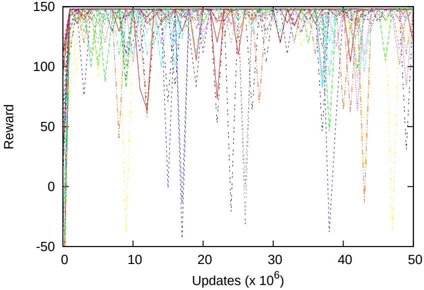
Performance Comparison for CA-GS 2D



Comparison of All Runs in CA-GS 2D



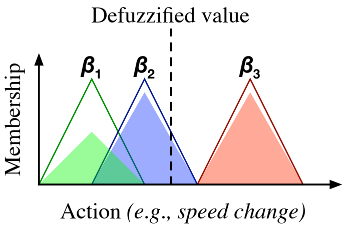
With GOALSEEK priority



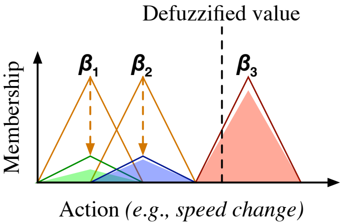
Replaced GOALSEEK priority

More results

Extending Behavior Modulation

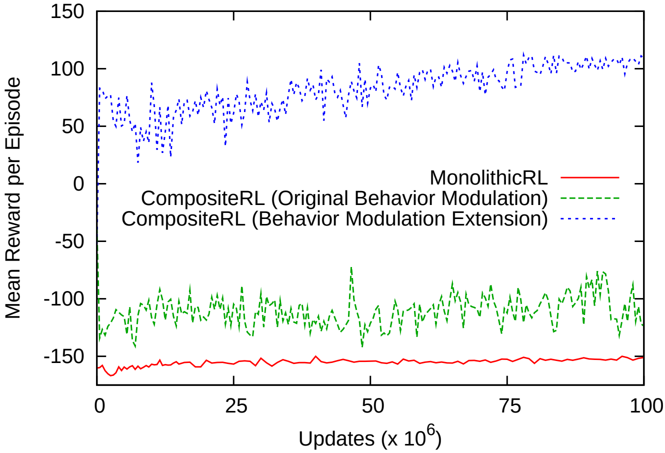


Original



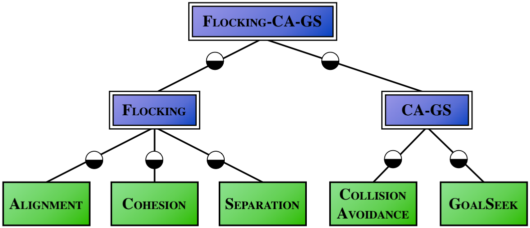
Extension

Extending Behavior Modulation: Flocking-CA-GS



Extending Behavior Modulation *(cont'd)*

- Original behavior modulation was not effective
- High-level modulation was ignored
- No coordination between FLOCKING and CA-GS
- Extension was able to effectively coordinate



Reinforcement Learning Approaches

- Monolithic and Modular RL both performed poorly
- Despite intent of Modular RL development
- Composite RL had almost immediate success
- In complex tasks, only Composite RL was effective
- Composite RL had less to learn
- Faster development even including estimated effort for learning reused behaviors

Behavior Reuse

- Was able to reuse existing behaviors without modification
- Sped up development of new controllers
- Much more practical
- Might limit performance potential, but
- Approaches that didn’t reuse weren’t effective

Conclusions & future work

Conclusions

- Only empirical investigation of state and action abstraction in CINE tasks of which we are aware
- Action abstraction is more beneficial than state abstraction
- Action abstraction essential in complex tasks
- Modulation extension necessary for complex hierarchies
- Composite RL was only effective RL approach
- Reused existing behaviors without modification

Future Work

- Investigation of the erratic behavior when using the GoalSeek adaptive priority
- Evaluate Composite RL with reused behaviors learned with RL
- Use fuzzy reinforcement learning
- Develop more complex controllers
- Combine adaptive fuzzy behavior hierarchies with multi-objective behavior coordination

Book Chapter Publications

Brent E. Eskridge and Dean F. Hougen.
Using State and Action Abstraction in Controllers for Concurrent, Interfering, Non-episodic Tasks.
I-Tech Education and Publishing, 2009. *To appear.*

Conference Publications

- Brent E. Eskridge and Dean F. Hougen.
  Prioritizing fuzzy behaviors in multi-robot pursuit teams.
  In *IEEE International Conference on Fuzzy Systems*, 2006.
- Brent E. Eskridge and Dean F. Hougen.
  Using priorities to simplify behavior coordination.
  In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007.
- Brent E. Eskridge and Dean F. Hougen.
  Using action abstraction to evolve effective controllers.
  In *Genetic and Evolutionary Computation Conference*. ACM, 2009. *To appear.*

Thank you!

- Maya, Connor, & Noah
- Family
- My advisor, Dean Hougen
- PhD committee members
- Members of the Artificial Intelligence Research (AIR) group
- Computing performed at OSCER

Questions?

Appendix

References

- Dimiter Driankov, Hans Hellendoorn, and Michael Reinfrank.
  An Introduction to Fuzzy Control.
  Springer-Verlag, second edition, 1996.

- Mark Humphrys.
  Action selection methods using reinforcement learning.
  In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 135–144. MIT Press, Bradford Books, 1996.

- Lionel Jouffe.
  Fuzzy inference system learning by reinforcement methods.
  IEEE Transactions on Systems, Man and Cybernetics, Part C, 28(3):338–355, 1998.

References (*cont'd*)

Jonas Karlsson.
Learning to Solve Multiple Goals.
PhD thesis, University of Rochester, Rochester, NY, USA, 1997.

Paolo Pirjanian.
Multiple Objective Action Selection & Behavior Fusion using Voting.
PhD dissertation, Aalborg University, Denmark, 1998.

Edward Tunstel.
Fuzzy-behavior synthesis, coordination, and evolution in an adaptive behavior hierarchy.
In *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, volume 61 of *Studies in Fuzziness and Soft Computing*, chapter 9, pages 205–234. Springer-Phisica Verlag, 2001.

Adaptive fuzzy behavior hierarchies

Fuzzy Rules for Primitive Behaviors

$$\text{IF } x \text{ is } \tilde{A}_i \text{ THEN } u \text{ is } \tilde{B}_i \tag{1}$$

Where

- x represents primitive task state information
- u represents motor command actions
- $\tilde{A}_i$ is a linguistic value corresponding to x
- $\tilde{B}_i$ is a linguistic value corresponding to u
- Antecedent can be compound
- Consequent can also be compound

Primitive Behavior Output

$$\tilde{\beta}_p = \bigcup_{i=1}^{M} \tilde{u}_i \tag{2}$$

Where

- M rules in the ruleset
- p is a primitive behavior
- $\tilde{u}_i$ output fuzzy set from rule i
- $\tilde{\beta}_p$ output fuzzy set

Fuzzy Rules for Composite Behaviors

$$\text{IF } x \text{ is } \tilde{A}_i \text{ THEN } \alpha \text{ is } \tilde{D}_i \tag{3}$$

Where

- x and $\tilde{A}_i$ are the same
- α is the scalar activation level of a given behavior
- $\tilde{D}_i$ is a linguistic value corresponding to α

Modulation of Primitive Behaviors

$$\alpha_p \cdot \tilde{\beta}_p \tag{4}$$

Where

- p is a modulated primitive behavior
- α_p is the activation level of$p$
- $\tilde{\beta}_p$ is the output of p

Output of Entire Behavior Hierarchy

$$\tilde{\beta}_H = \biguplus_{p \in P} \alpha_p \cdot \tilde{\beta}_p \qquad (5)$$

Where

- $\tilde{\beta}_H$ is the output fuzzy set
- P is the set of all primitive behaviors
- $\biguplus$ is the arithmetic sum of the fuzzy sets over all the primitive behaviors

Defuzzification of Behavior Hierarchy Output

$$u^* = \frac{\sum_{i=1}^{n} u_i \sum_{p \in P} \mu_{\tilde{\beta}_H}(u_i)}{\sum_{i=1}^{n} \sum_{p \in P} \mu_{\tilde{\beta}_H}(u_i)} \quad (6)$$

Where

- Center-of-Sums defuzzification
- u is the motor command output fuzzy variable
- μ is the membership function over all possible actions

Behavior modulation extension

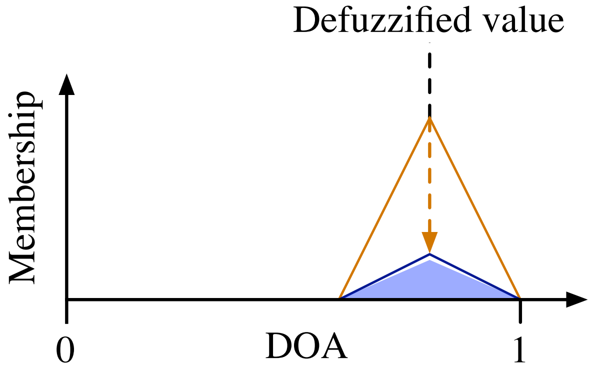
Modulation Conditions

Any modification to behavior modulation must:

1. Produce current correct modulation for a 2-level hierarchy
2. Account for more than one composite behavior modulating a given primitive behavior

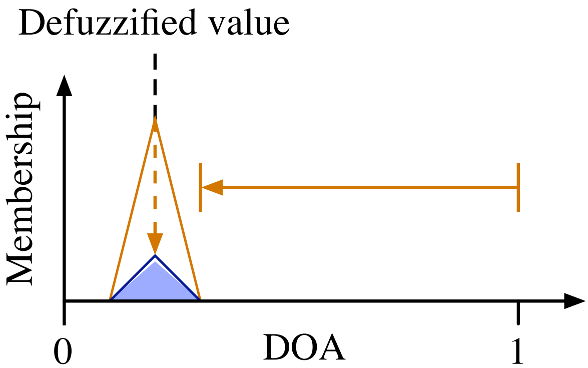
Return

DOA Calculation in Original



Return

DOA Calculation in Extension



Primitive Behavior DOA Calculation

$$\alpha_p = \frac{\int_{u \in U} u \sum_{c \in C} \alpha_c \cdot \mu_{\tilde{B}_c}(u)}{\int_{u \in U} \sum_{c \in C} \alpha_c \cdot \mu_{\tilde{B}_c}(u)} \qquad (7)$$

Where

- B_c is a composite behavior
- $\mu_{\tilde{B}_c}(u)$ is the DOA membership value assigned by B_c

Return

Simple Case

$$\alpha'_p = \alpha_p \cdot \alpha_c \tag{8}$$

Where

- α_p is the primitive behavior's calculated DOA
- α_c is the composite behavior's DOA
- α'_p is the final DOA of the primitive behavior

Return

Scaling Factor Calculation

$$S_p = \frac{\sum_{c \in C} \alpha_c \int_{u \in U} u \cdot \mu_{\tilde{B}_c}(u)}{\sum_{c \in C} \int_{u \in U} u \cdot \mu_{\tilde{B}_c}(u)} \quad (9)$$

Where

- S_p is the scaling factor for primitive behavior p
- S_p equals α_c in the simple case

Return

Final DOA Calculation for a Primitive Behavior

$$\alpha'_p = \alpha_p \cdot S_p \tag{10}$$

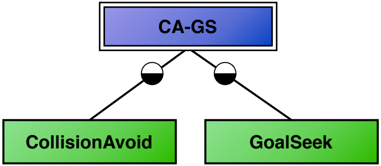
In the simple case, we recover Equation 8:

$$\alpha'_p = \alpha_p \cdot S_p = \alpha_p \cdot \alpha_c$$

Return

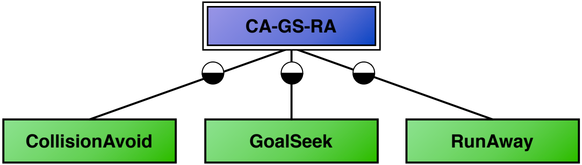
Task behavior hierarchies

CA-GS Composite Task



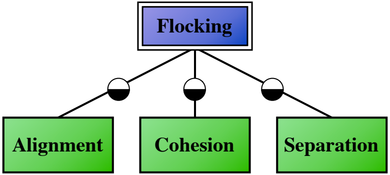
Return

CA-GS-RA Composite Task



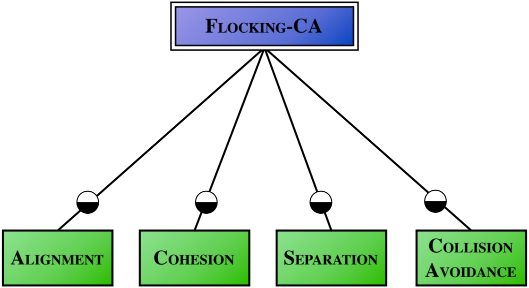
Return

FLOCKING Composite Task



Return

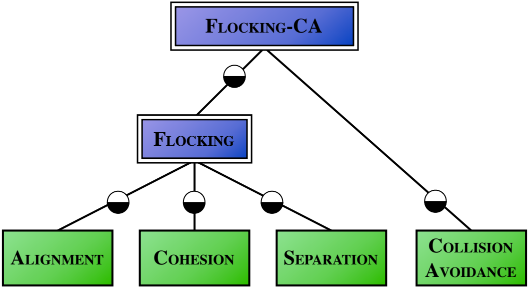
Flocking-CA Composite Task



2-Level Hierarchy

Return

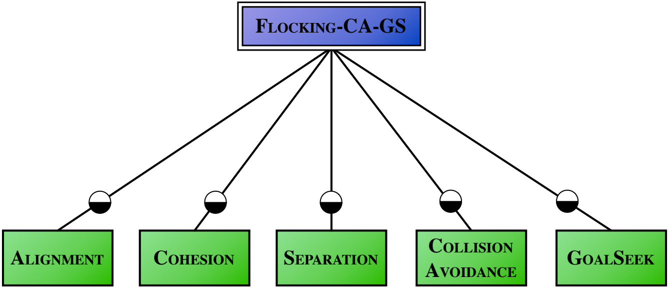
FLOCKING-CA Composite Task



3-Level Hierarchy

Return

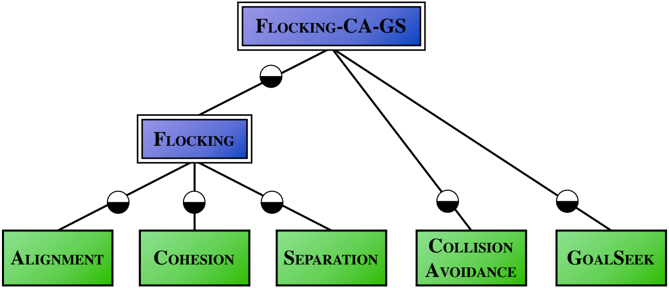
FLOCKING-CAGS Composite Task



2-Level Hierarchy

Return

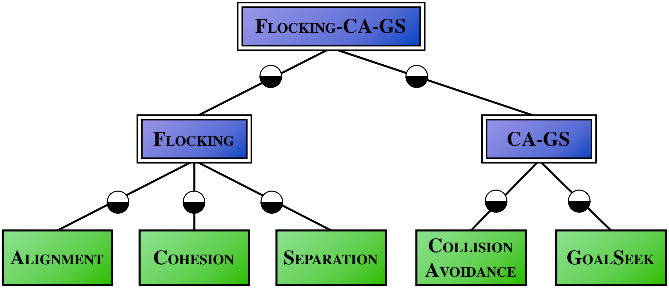
FLOCKING-CAGS Composite Task



3-Level Hierarchy

Return

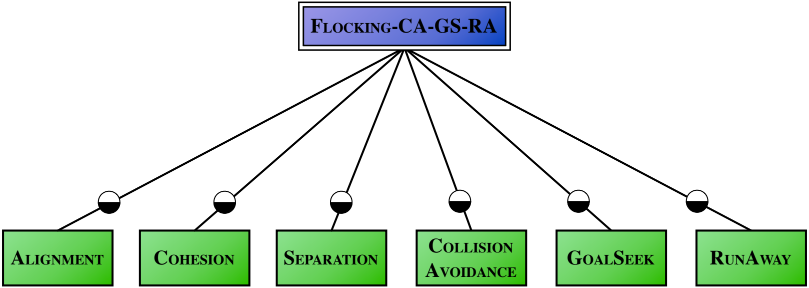
FLOCKING-CAGS Composite Task



3-Level Hierarchy *(Alternative)*

Return

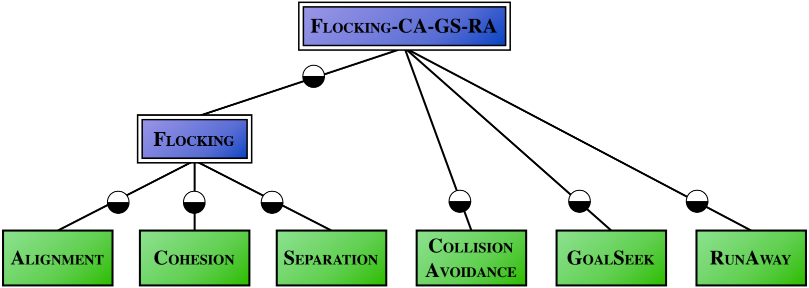
FLOCKING-CAGSRA Composite Task



2-Level Hierarchy

Return

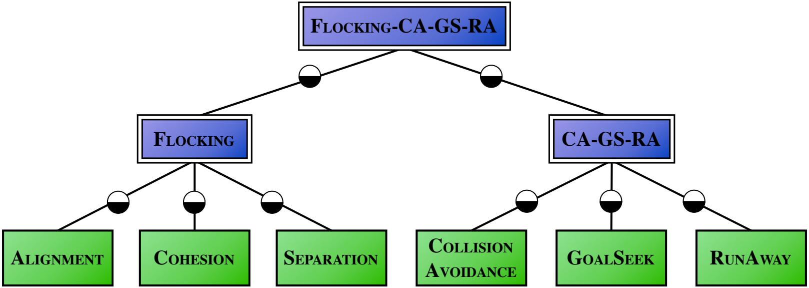
FLOCKING-CAGSRA Composite Task



3-Level Hierarchy

Return

FLOCKING-CAGSRA Composite Task

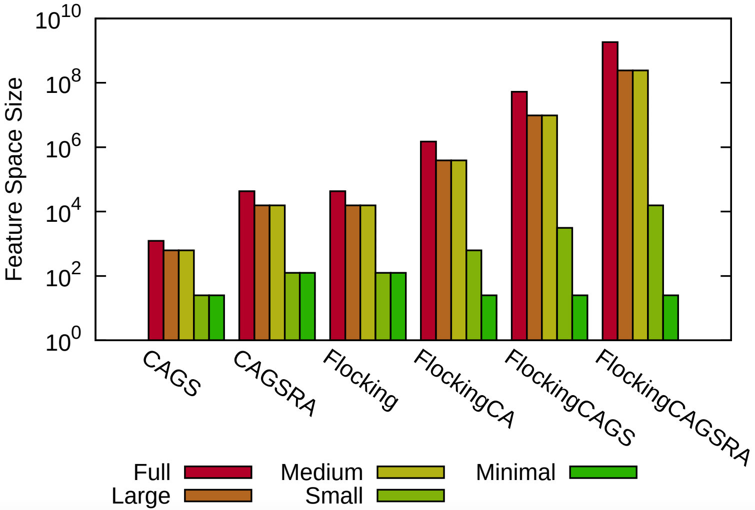


3-Level Hierarchy *(Alternative)*

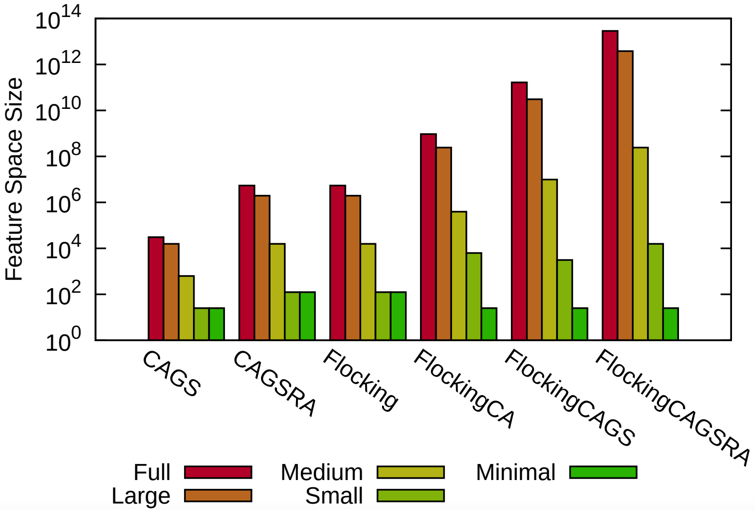
Return

State space sizes

2D State Space Sizes

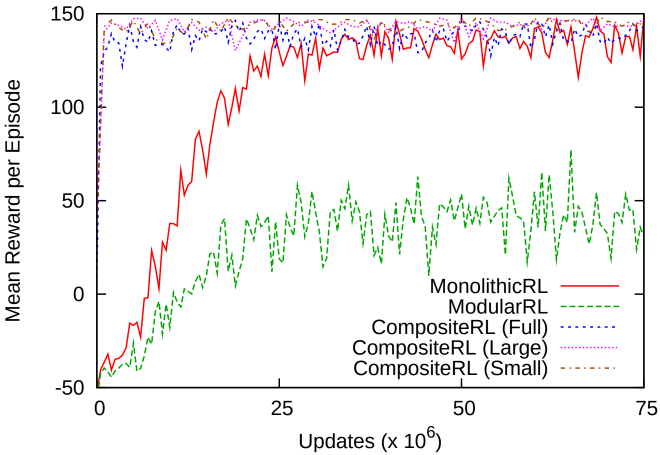


3D State Space Sizes



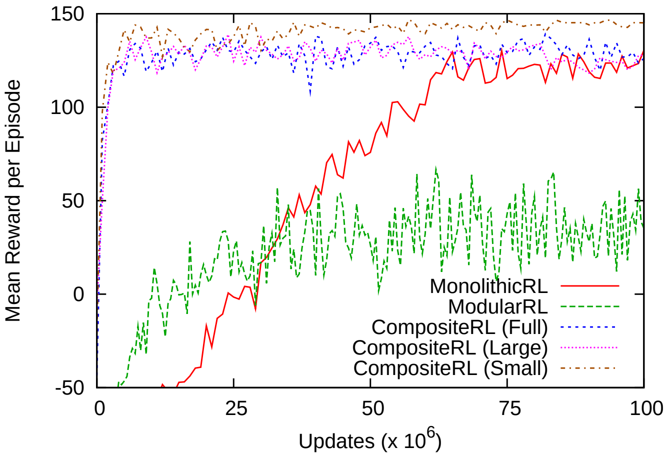
Small results

CA-GS 2D Results - Modified **Small**

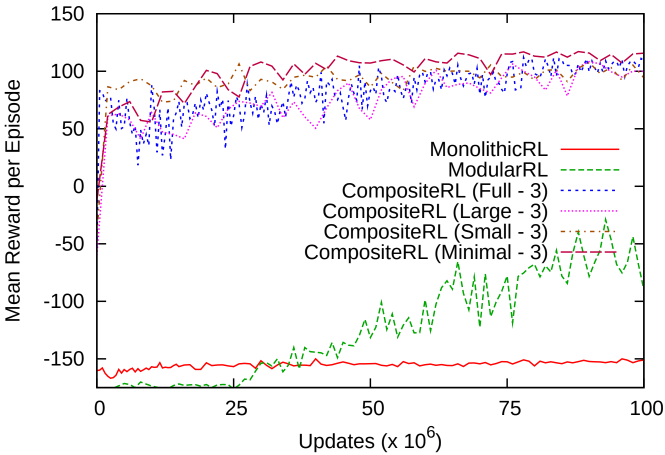


Return

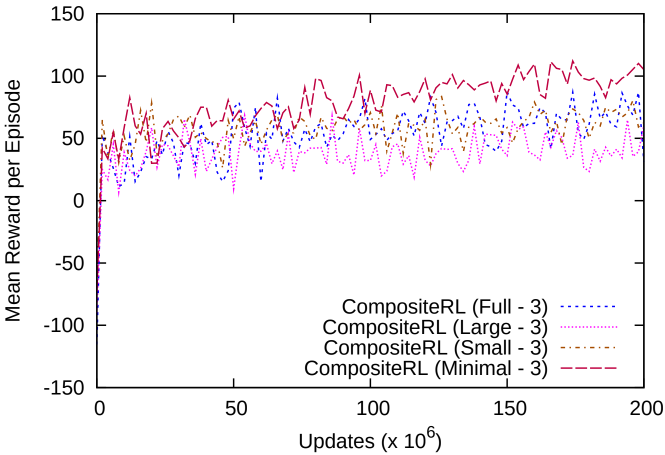
CA-GS-RA 2D Results - Modified **Small**



FLOCKING-CA-GS 2D Results - Modified Small



Flocking-CA-GS-RA 2D Results - Modified **Small**



Return