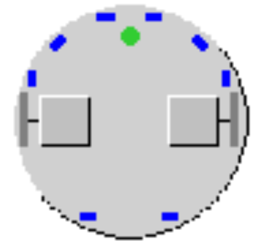
Evolving Multimodal Behavior Through Modular Multiobjective Neuroevolution

By Jacob Schrum

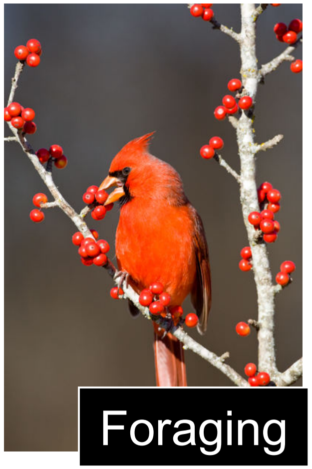
Introduction


- Challenge: Discover behavior automatically
  - Simulations, video games, robotics
- Why challenging?
  - Noisy sensors
  - Complex domains
  - Continuous states/actions
  - Multiple agents
  - Multiple objectives
  - ➢ Multimodal behavior required **(focus)**

Multimodal Behavior

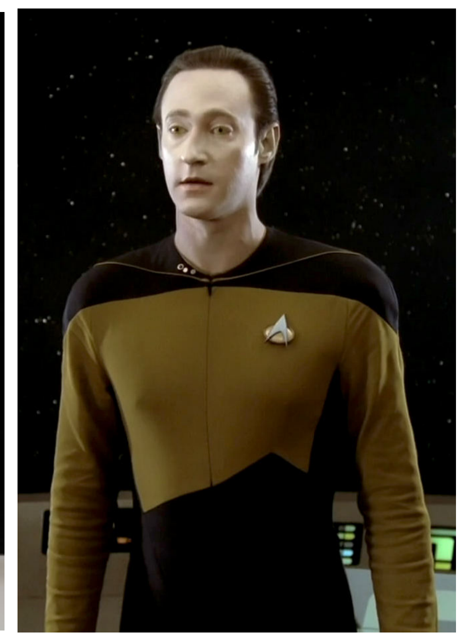
- Animals can perform many different tasks



- Imagine learning a monolithic policy as complex as a cardinal’s behavior: HOW?
- Problem more tractable if broken into component behaviors

Multimodal Assistants

- Consider all the things we would like computers/robots to eventually do for/with us
- We can program one behavior at a time, but how does it all combine in one brain?



Outline

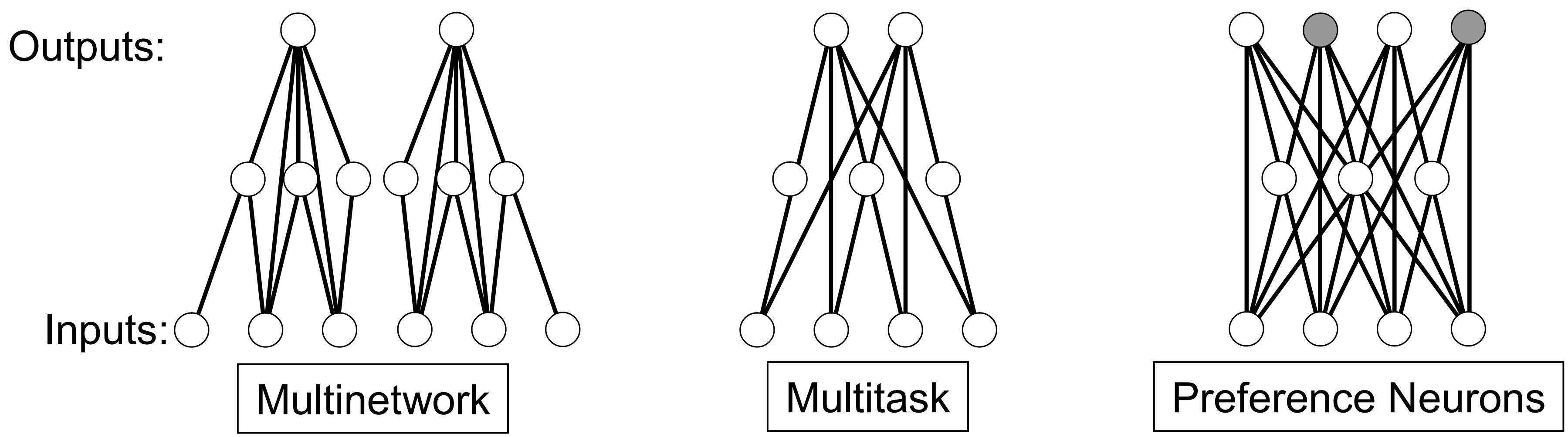
- Motivation
- Multimodal Behavior
  - What is it?
  - How to learn it?
- Methods
- Domains/Experiments
- Discussion/Conclusion

What is Multimodal Behavior?

- From Observing Agent Behavior:
  - Agent performs distinct tasks
  - Behavior very different in different tasks
    - Single function would have trouble generalizing
- Reinforcement Learning Perspective
  - Similar to Hierarchical Reinforcement Learning
  - A “mode” of behavior is like an “option”
    - A temporally extended action
    - A control policy that is only used in certain states
  - Policy for each mode must be learned as well
- Idea From Supervised Learning
  - Multitask Learning trains on multiple known tasks

Modular Policy

- One policy consisting of several policies/modules
  - Number preset, or learned
- Means of arbitration also needed
  - Human specified, or learned via preference neurons
- Separate behaviors easily represented
  - Sub-policies/modules can share components



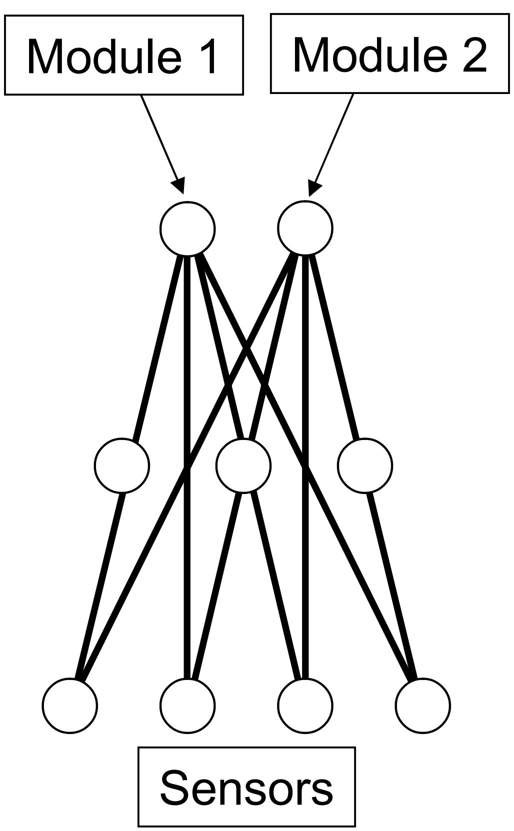
(Caruana 1997)

How to Learn Multimodal Behavior?

- Networks with multiple modules
  - Multitask: set the task division
  - Preference neurons: learn the task division
  - Module Mutation: learn number of modules as well
- Learning algorithm
  - Multiobjective: mode/objective correspondence
  - TUG: Where to focus evolutionary search
- Sensor design
  - Split sensors encourage a task division

Behavioral Modes vs. Network Modules

- Different behavioral modes
  - Determined via observation of behavior, subjective
  - Any net can exhibit multiple behavioral modes
- Different network modules
  - Determined by connectivity of network
  - Groups of “policy” outputs designated as modules (sub-policies)
  - Modules distinct even if behavior is same/unused
  - Network modules should help build behavioral modes



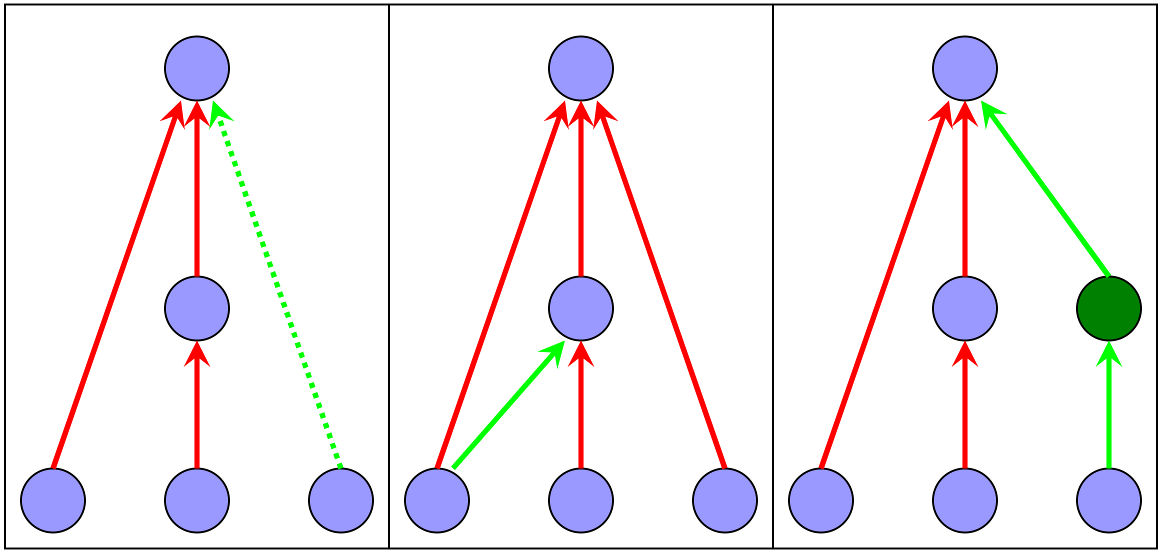
Outline

- Motivation
- Multimodal Behavior
- Methods
  - Neuroevolution
    - Module Mutation (Contribution)
  - Multiobjective optimization
    - TUG (Contribution)
- Domains/Experiments
- Discussion/Conclusion

Constructive Neuroevolution

- Genetic Algorithms + Neural Networks
- Build structure incrementally
- Good at generating control policies
- Three basic mutations (+ Crossover)
- Other structural mutations possible

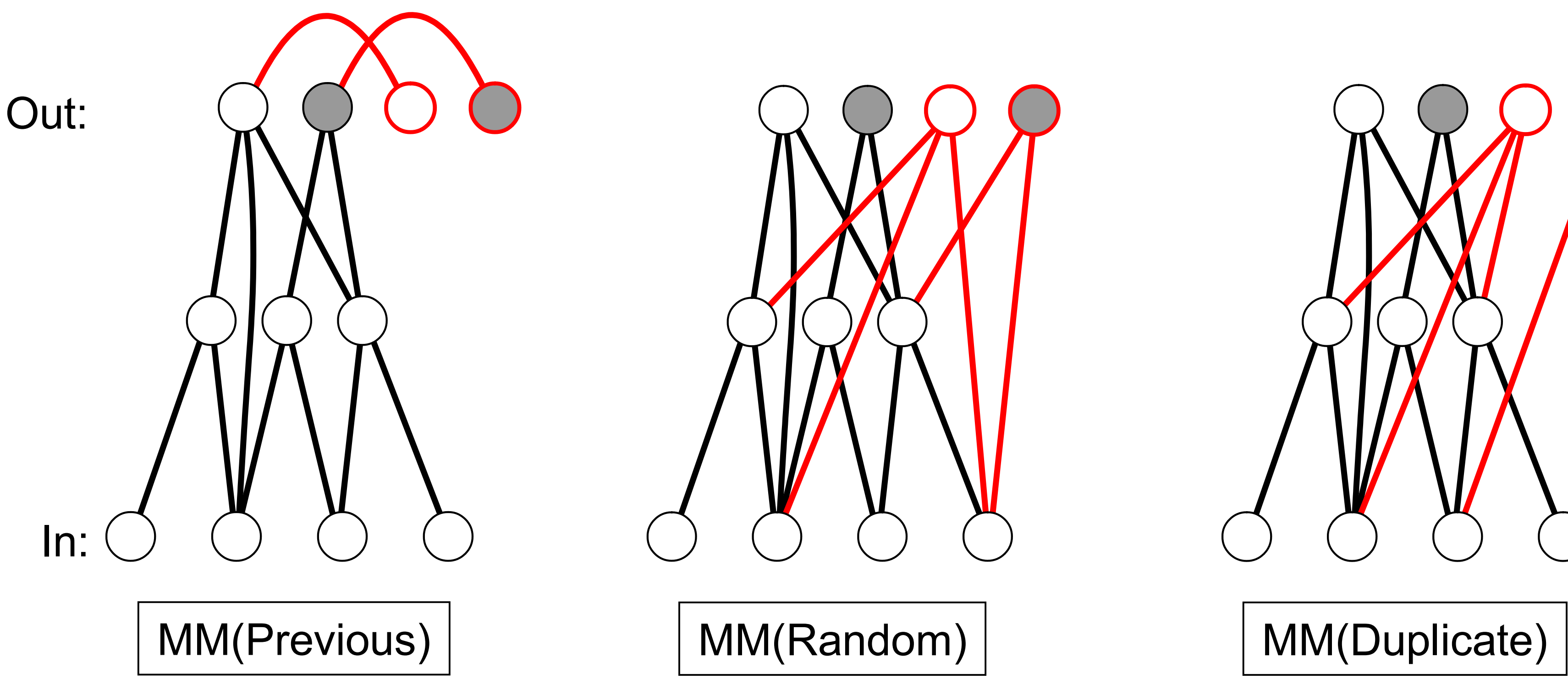
(cf NEAT by Stanley 2004)



Perturb Weight Add Connection Add Node

Module Mutation

- A mutation that adds a module
- Can be done in many different ways
- Can happen more than once for multiple modules



(Schrum and Miikkulainen 2009, 2011, 2012)

(cf Calabretta et al 2000)

Pareto-based Multiobjective Optimization

(Pareto 1890)

Imagine game with two objectives:

- Damage Dealt

- Health Remaining

Attack and retreat modes?

$\vec{v}$ dominates $\vec{u}$, i.e. $\vec{v} \succ \vec{u} \Leftrightarrow$

1. $\forall i \in \{1,\ldots,n\}(v_i \geq u_i)$ and

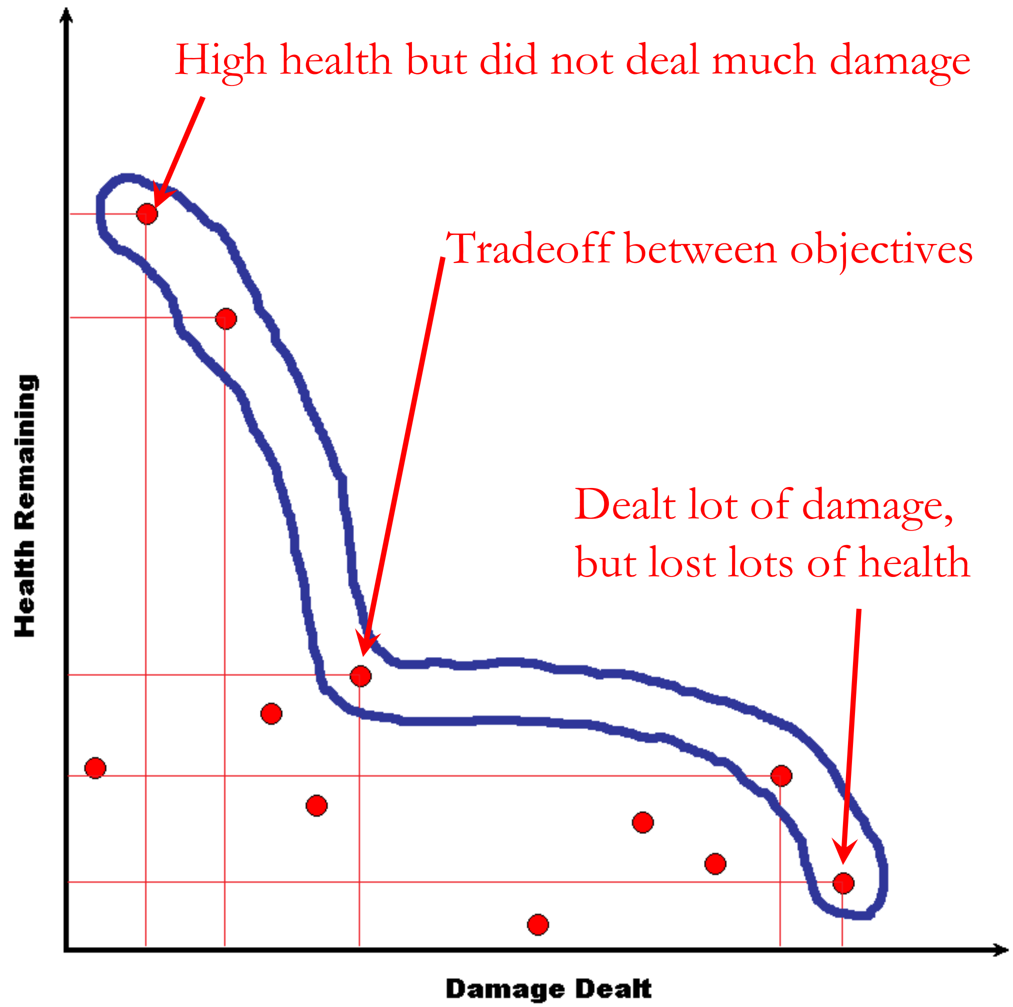
2. $\exists i \in \{1,\ldots,n\}(v_i > u_i)$

Non-dominated points best:

$A \subseteq F$ is Pareto optimal $\Leftrightarrow$

A contains all points in F s.t.

$\forall \vec{x} \in A \neg \exists \vec{y} \in F(\vec{y} \succ \vec{x})$

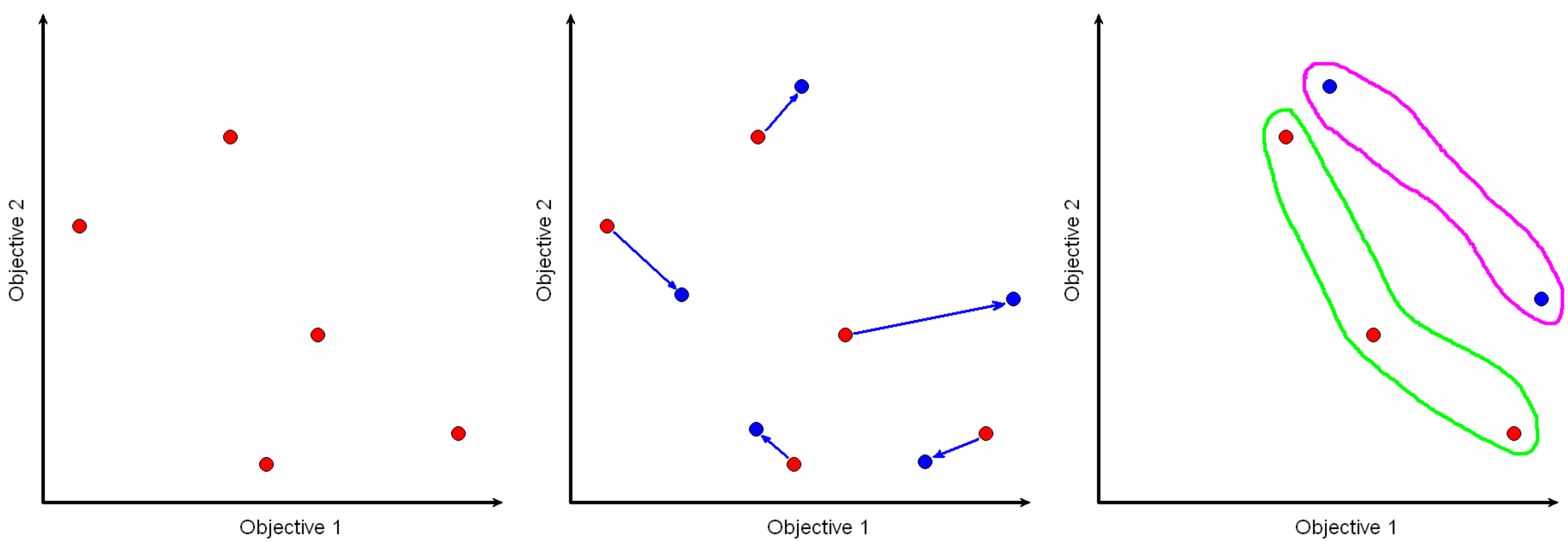


Useful if modes correspond to objectives

Non-dominated Sorting Genetic Algorithm II

(Deb et al. 2000)

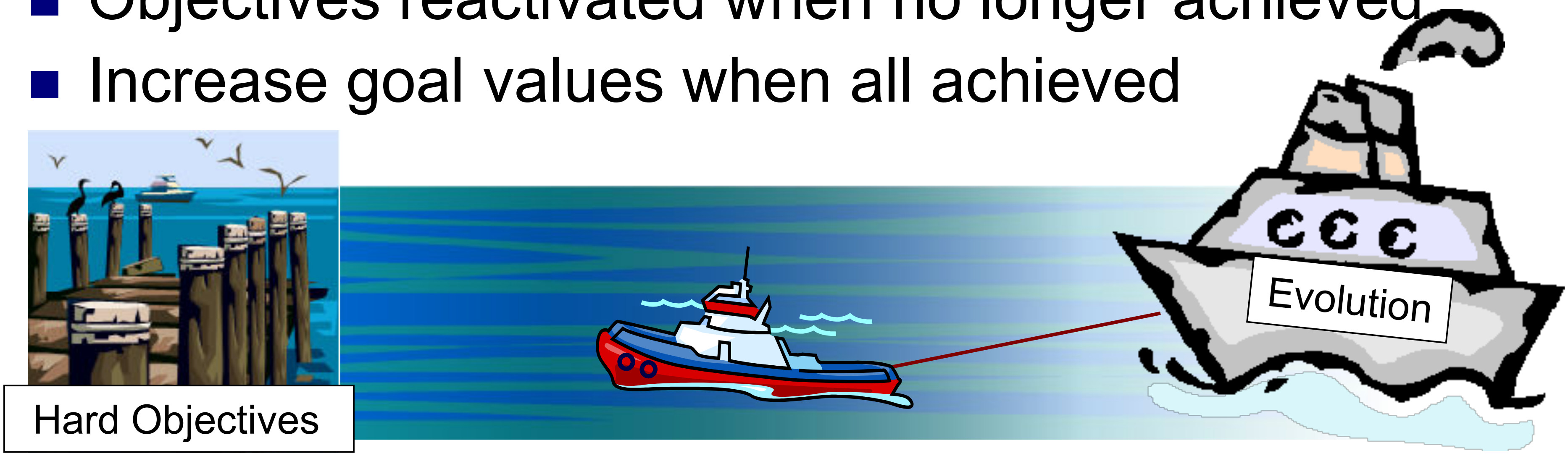
- Population P with size N; Evaluate P
- Use mutation (& crossover) to get P´ size N; Evaluate P´
- Calculate non-dominated fronts of P ∪ P´ size 2N
- New population size N from highest fronts of P ∪ P´



Targeting Unachieved Goals

(Schrum and Miikkulainen 2010)

- Main ideas:
  - Temporarily deactivate “easy” objectives
  - Focus on “hard” objectives
- “Hard” and “easy” defined in terms of goal values
  - Easy: average fitness “persists” above goal (achieved)
  - Hard: goal not yet achieved
- Objectives reactivated when no longer achieved
- Increase goal values when all achieved



TUG Goal Achievement

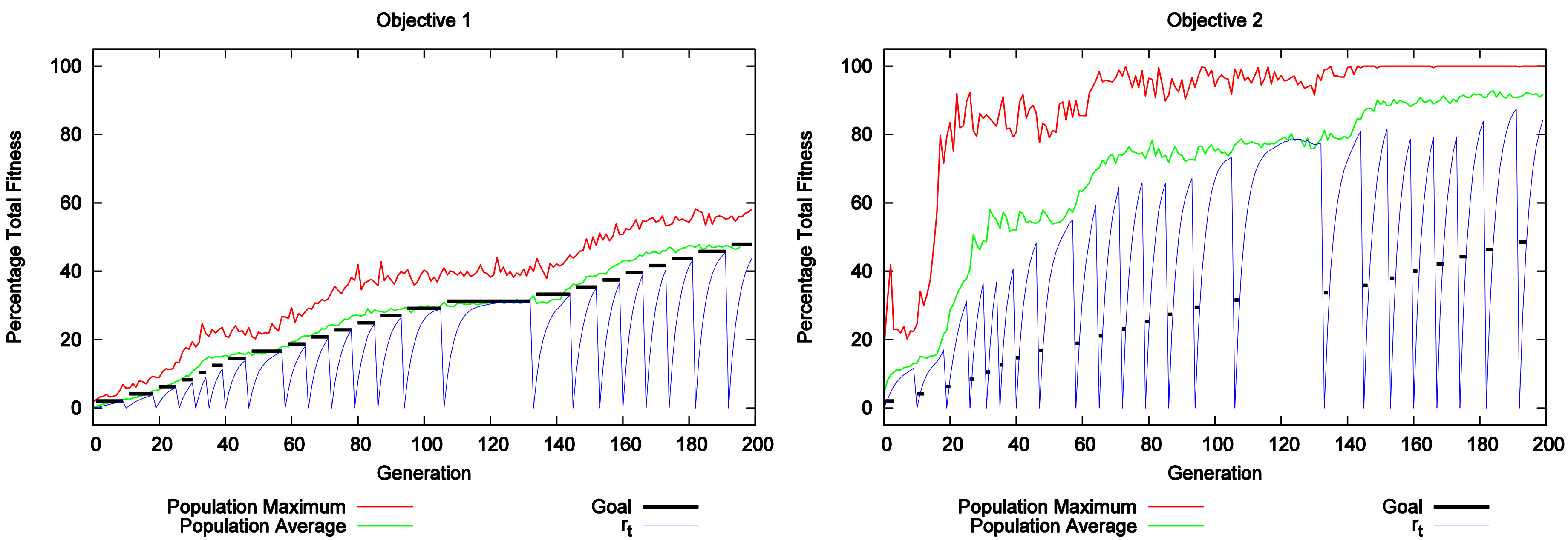
- Persistent goal achievement
  - Recency-weighted average catches up

$$r_t \leftarrow r_{t-1} + \alpha(\bar{x}_t - r_{t-1})$$

r_t : Recency - weighted average of average score on generation t

$\bar{x}_t$: Average population objective score on generation t

α : Step - size parameter (how quickly r_t catches up)



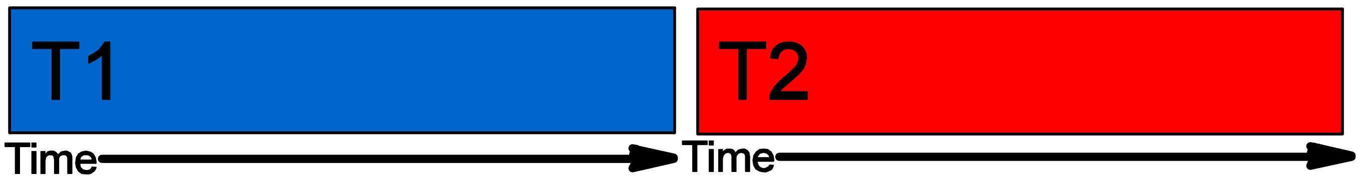
Outline

- Motivation
- Multimodal Behavior
- Methods
- Domains/Experiments
  - Types of divisions
  - Front/Back Ramming (constructed)
  - Predator/Prey (constructed)
  - Battle Domain (constructed)
  - Ms. Pac-Man (real)
- Discussion/Conclusion

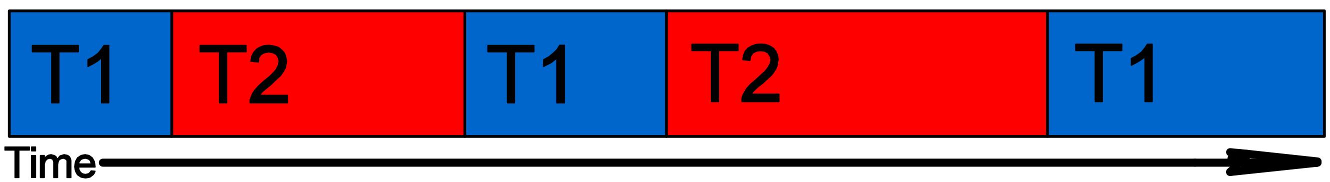
How will these methods work in domains with different types of task divisions?

Domains with Multiple Tasks

- Tasks can be completely isolated
  - Evaluation in one does not affect other



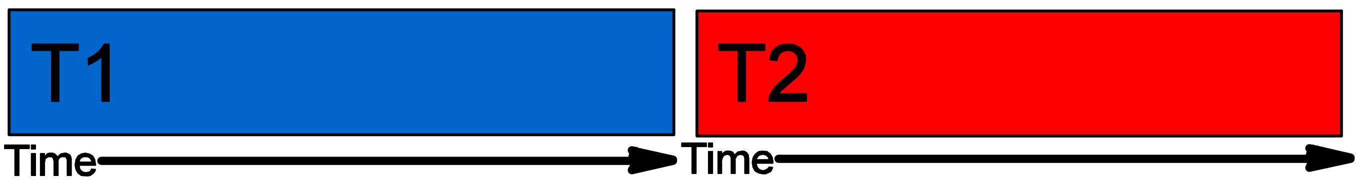
- Tasks may be interleaved
  - Alternates between tasks, but division is clear



- Division can be ambiguous, uncertain
  - Are tasks completely separate?

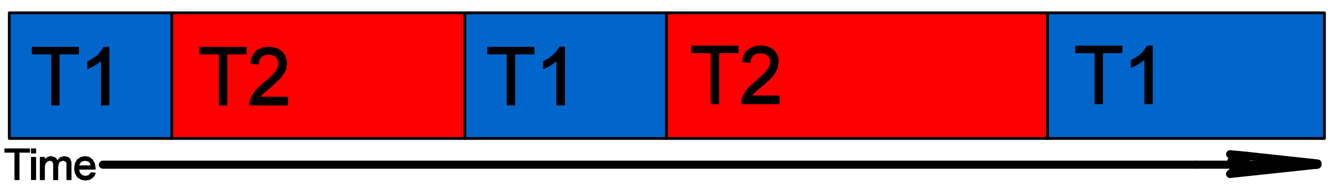
Domains with Multiple Tasks

- Tasks can be completely isolated
  - Evaluation in one does not affect other



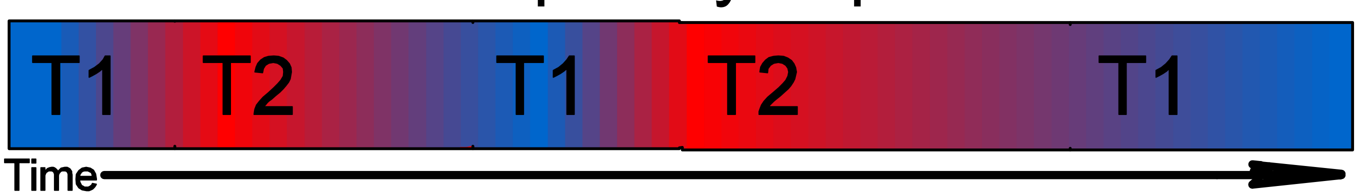
Front/Back Ram
Predator/Prey

- Tasks may be interleaved
  - Alternates between tasks, but division is clear



Imprison PM

- Division can be ambiguous, uncertain
  - Are tasks completely separate?



Battle Domain
Full PM

Outline

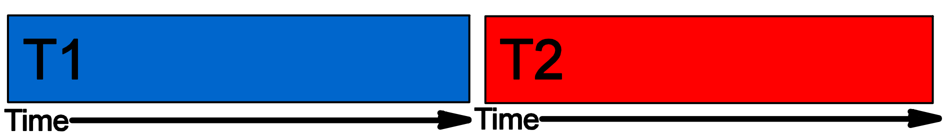
- Motivation
- Multimodal Behavior
- Methods
- Domains/Experiments
  - Types of divisions
  - Front/Back Ramming
  - Predator/Prey
  - Battle Domain
  - Ms. Pac-Man
- Discussion/Conclusion

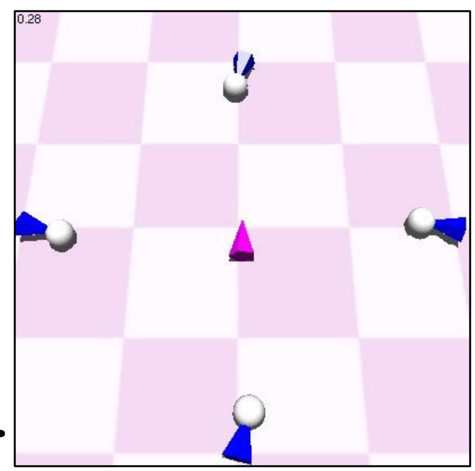
- Two isolated tasks
- Equal difficulty
- Multimodal behavior needed to succeed
  - Are network modules needed?

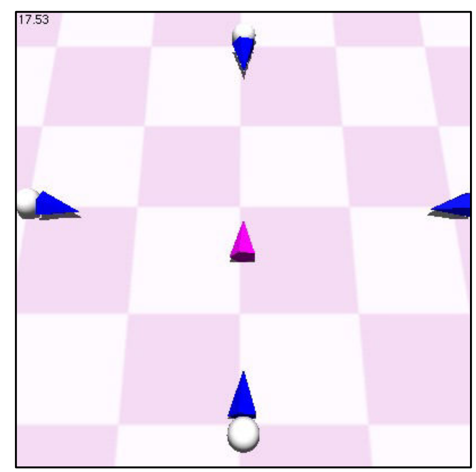
Front/Back Ramming

(Schrum and Miikkulainen 2011, 2012)

- Four evolved monsters surround bot
- Each has a spherical ram attached
  - Attached either on front or back of monster
- The ram can damage the bot
- Rest of body vulnerable to bot
- Monster goals: in each task
  - Damage bot
  - Avoid damage
  - Stay alive

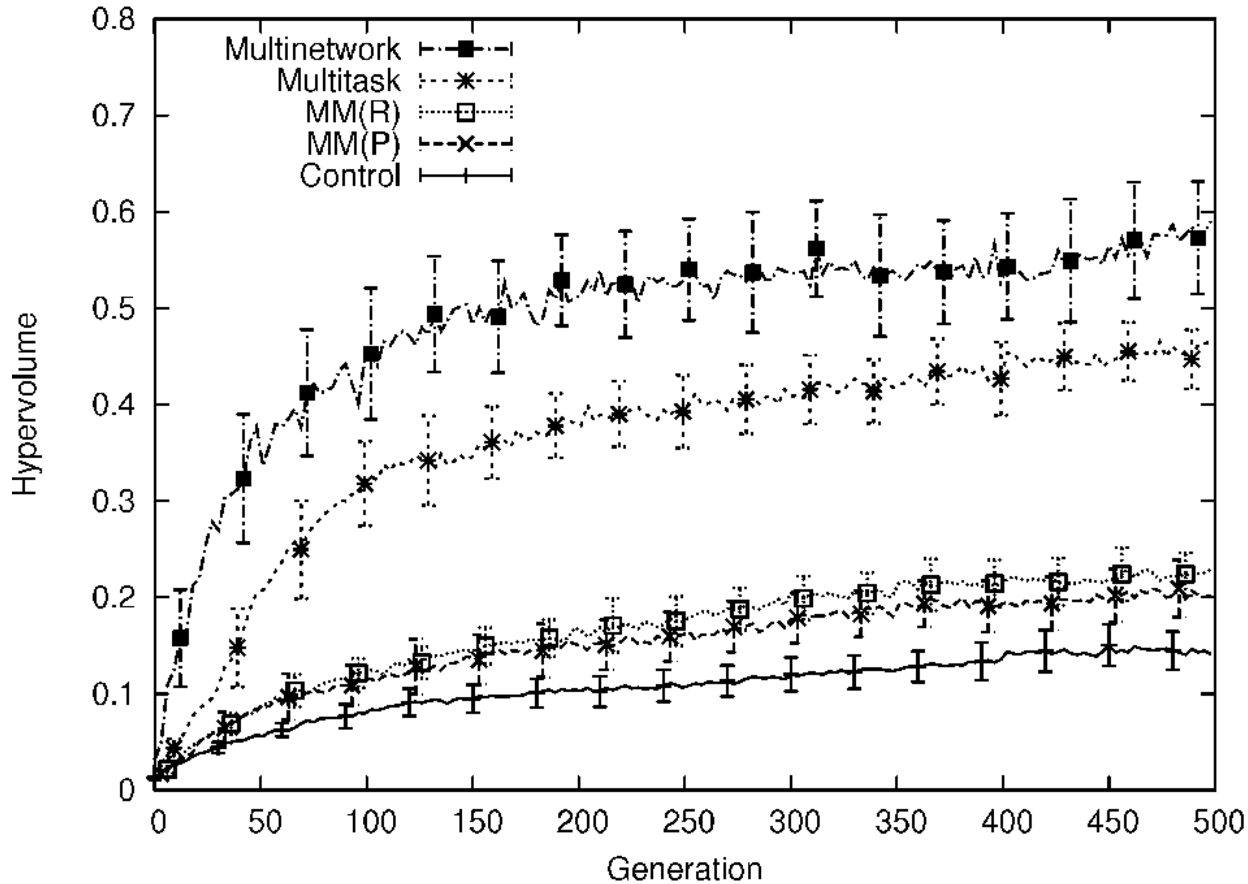






Front/Back Ramming Results

- Two complex tasks
  - Both similar
  - Equal difficulty
- Strong division best
  - Multitask
  - Multinetwork
- Middle division next
  - Module Mutation
    - Both tasks use multiple modules
    - One module helps determine current task
    - One module for retreating
    - One module for attacking



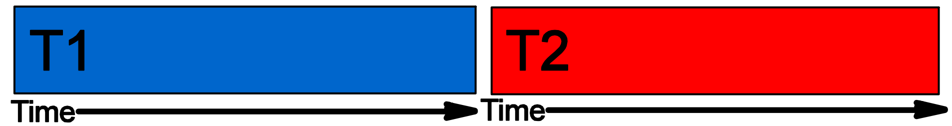
Outline

- Motivation
- Multimodal Behavior
- Methods
- Domains/Experiments
  - Types of divisions
  - Front/Back Ramming
  - Predator/Prey
  - Battle Domain
  - Ms. Pac-Man
- Discussion/Conclusion

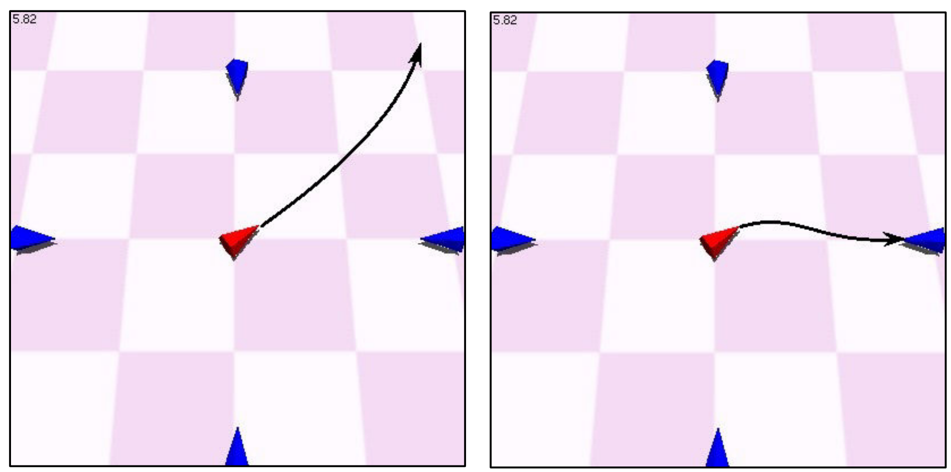
- Two isolated tasks
- Skewed difficulty
- Multimodal behavior needed to succeed
  - How will it differ?

Predator/Prey

(Schrum and Miikkulainen 2011, 2012)

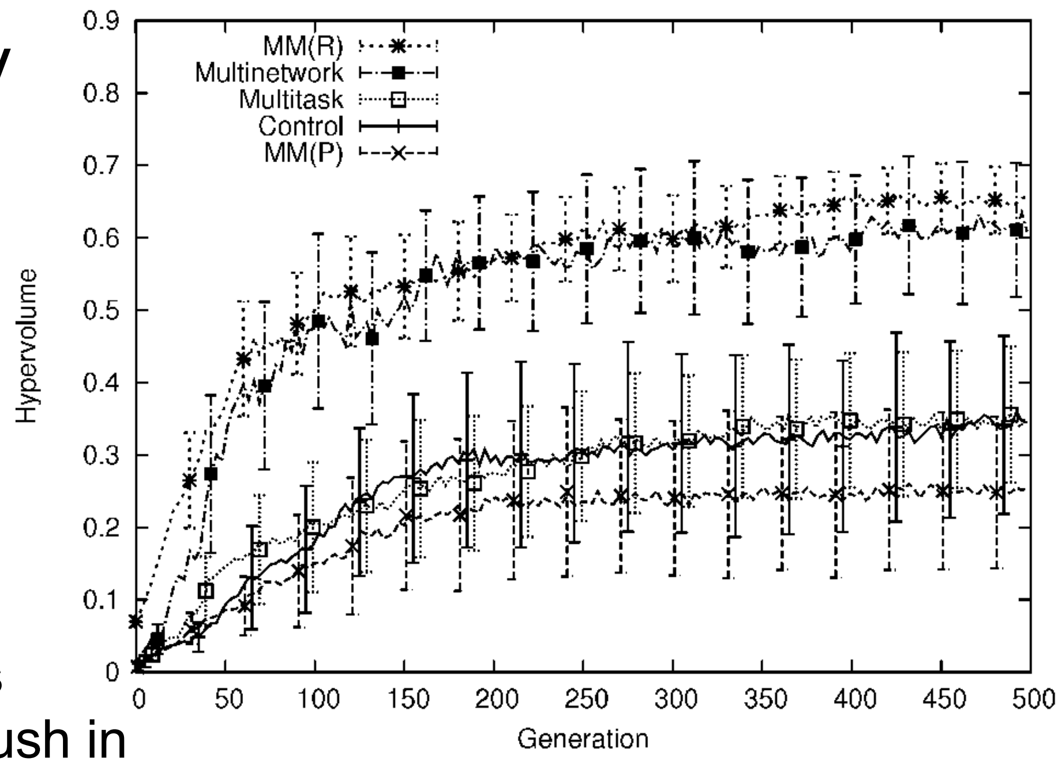


- Four evolved monsters surround bot
  - In Predator evaluation, monster deal damage
    - Bot is safe after escaping ring of monsters
  - In Prey evaluation, bot damages monsters
- Clear division, but not equal in difficulty
  - Predator task harder: attack and confine
- Predator goals
  - Damage bot
- Prey goals
  - Avoid damage
  - Stay alive



Predator/Prey Results

- Surprisingly, Multitask performs poorly
  - Modules interfering with each other
- But Multinetwork performs well
  - The task division does work
- MM(P) performs poorly
- MM(R) works well
  - Multiple modules used
  - One module favored
  - Unexpected division
    - Retreating and attacking both in one module
    - Second module restrains teammates so one can rush in



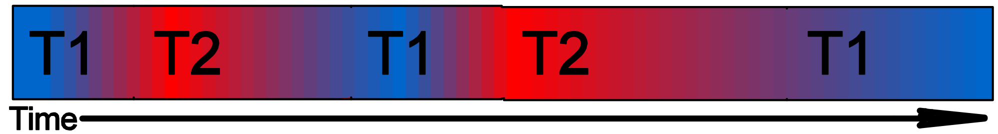
Outline

- Motivation
- Multimodal Behavior
- Methods
- Domains/Experiments
  - Types of divisions
  - Front/Back Ramming
  - Predator/Prey
  - Battle Domain
  - Ms. Pac-Man
- Discussion/Conclusion

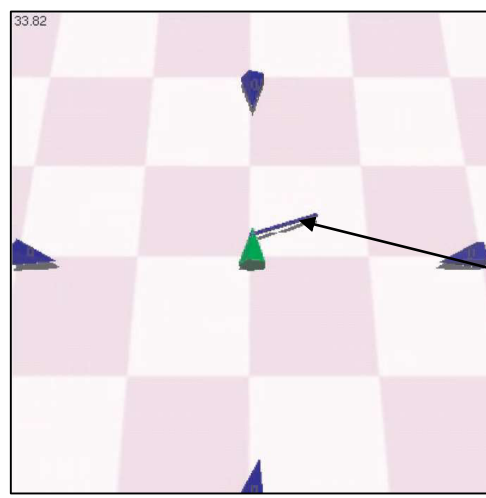
- Two blended tasks
- Evaluate TUG
- Multimodal behavior needed to succeed
  - Importance of timing

Battle Domain

(Schrum and Miikkulainen 2010)



- Four evolved monsters surround opponent
- Bot chases nearest monster
  - Repeatedly wings damaging bat
  - Short time between swings
  - Body vulnerable to monsters
- Offensive and defensive tasks blended
- Monster goals
  - Damage bot
  - Avoid damage
  - Stay alive

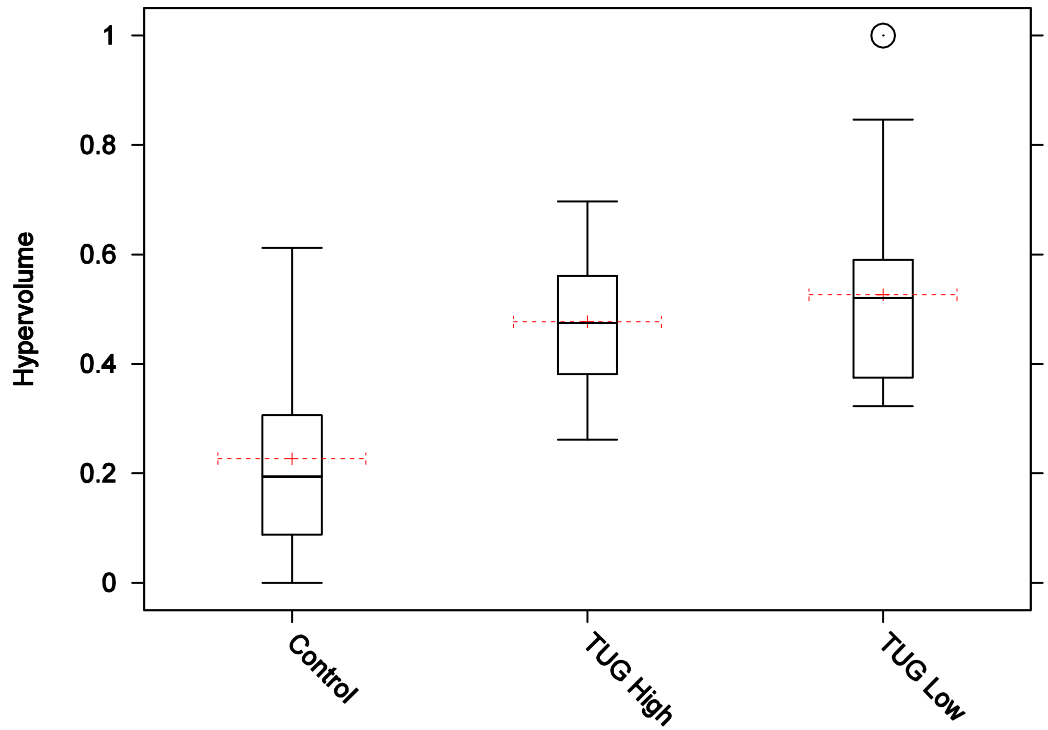


Monster must time their attacks to avoid the bot's bat

Bat

Battle Domain Results

- TUG outperforms plain NSGA-II
- Learns multimodal behavior
  - Precise timing of retreat and attack
  - Trading roles between teammates
  - Baiting
- Different initial goals successful

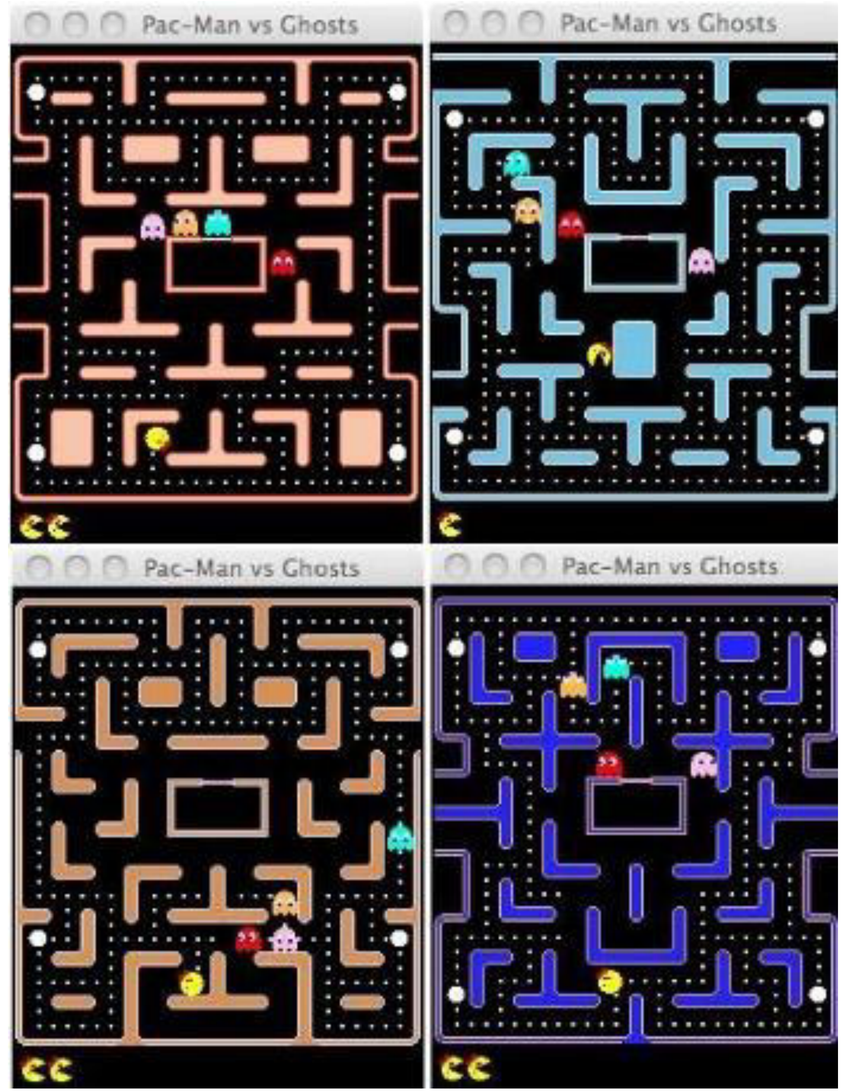


Outline

- Motivation
- Multimodal Behavior
- Methods
- Domains/Experiments
  - Types of divisions
  - Front/Back Ramming
  - Predator/Prey
  - Battle Domain
  - Ms. Pac-Man
    - Blended tasks
    - Scale to real game
    - Compare with others
- Discussion/Conclusion

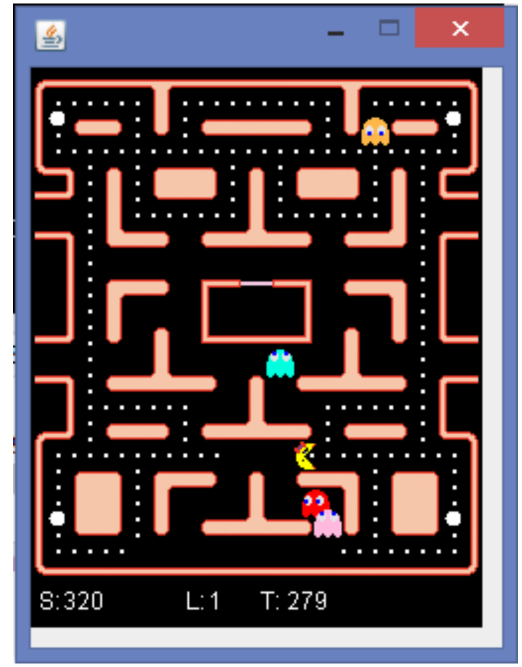
Ms. Pac-Man

- Domain needs multimodal behavior to succeed
- Classic, well-known game
  - Lots of previous work
- Predator/prey variant
  - Pac-Man takes on both roles
- Goals: Maximize score by
  - Eating all pills in each level
  - Avoiding threatening ghosts
  - Eating ghosts (after power pill)
- Non-deterministic
  - Very noisy evaluations
- Four mazes
  - Behavior must generalize



Task Overlap

- Distinct behavioral modes
  - Eating edible ghosts
  - Clearing levels of pills
  - More?
- Are ghosts currently edible?
  - Possible some are and some are not
  - Task division is blended
- Test One Life and Multiple Lives
- Compare with scores from literature

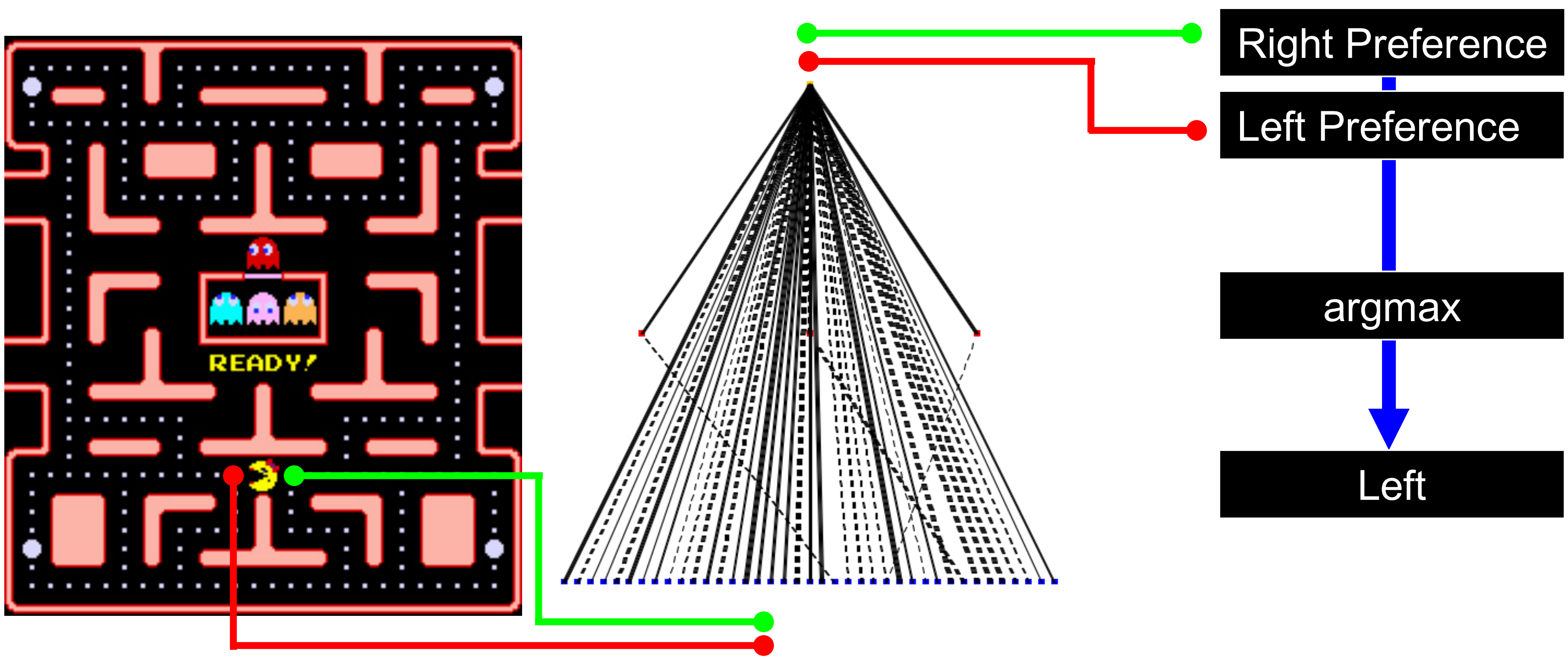


Previous Work in Pac-Man

- Custom Simulators
  - Genetic Programming: Koza 1992
  - Neuroevolution: Gallagher & Ledwich 2007, Burrow & Lucas 2009, Tan et al. 2011
  - Reinforcement Learning: Burrow & Lucas 2009, Subramanian et al. 2011, Bom 2013
  - Alpha-Beta Tree Search: Robles & Lucas 2009
- Screen Capture Competition: Requires Image Processing
  - Evolution & Fuzzy Logic: Handa & Isozaki 2008
  - Influence Map: Wirth & Gallagher 2008
  - Ant Colony Optimization: Emilio et al. 2010
  - Monte-Carlo Tree Search: Ikehata & Ito 2011
  - Decision Trees: Foderaro et al. 2012
- Pac-Man vs. Ghosts Competition: Pac-Man
  - Genetic Programming: **Alhejali & Lucas 2010, 2011, 2013, Brandstetter & Ahmadi 2012**
  - Monte-Carlo Tree Search: Samothrakis et al. 2010, **Alhejali & Lucas 2013**
  - Influence Map: Svensson & Johansson 2012
  - Ant Colony Optimization: **Recio et al. 2012**
- Pac-Man vs. Ghosts Competition: Ghosts
  - Neuroevolution: Wittkamp et al. 2008
  - Evolved Rule Set: Gagne & Congdon 2012
  - Monte-Carlo Tree Search: Nguyen & Thawonmos 2013

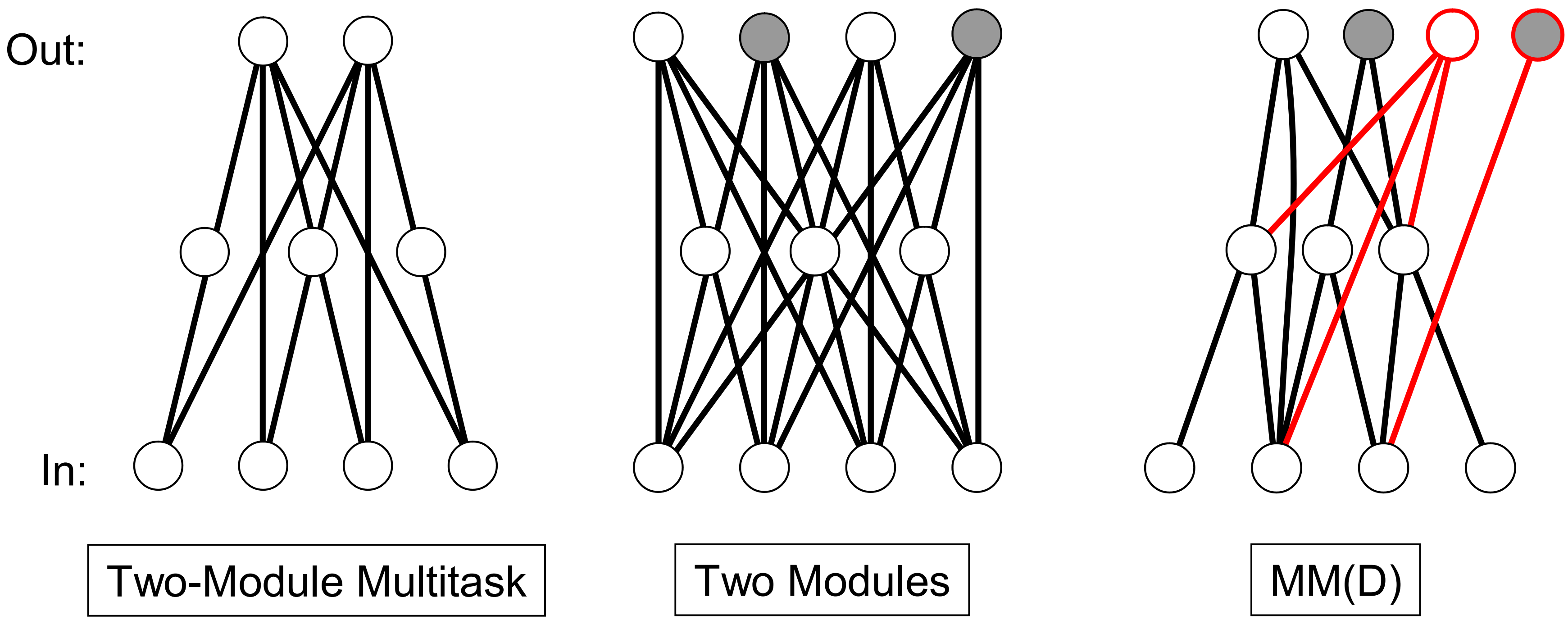
Evolved Direction Evaluator

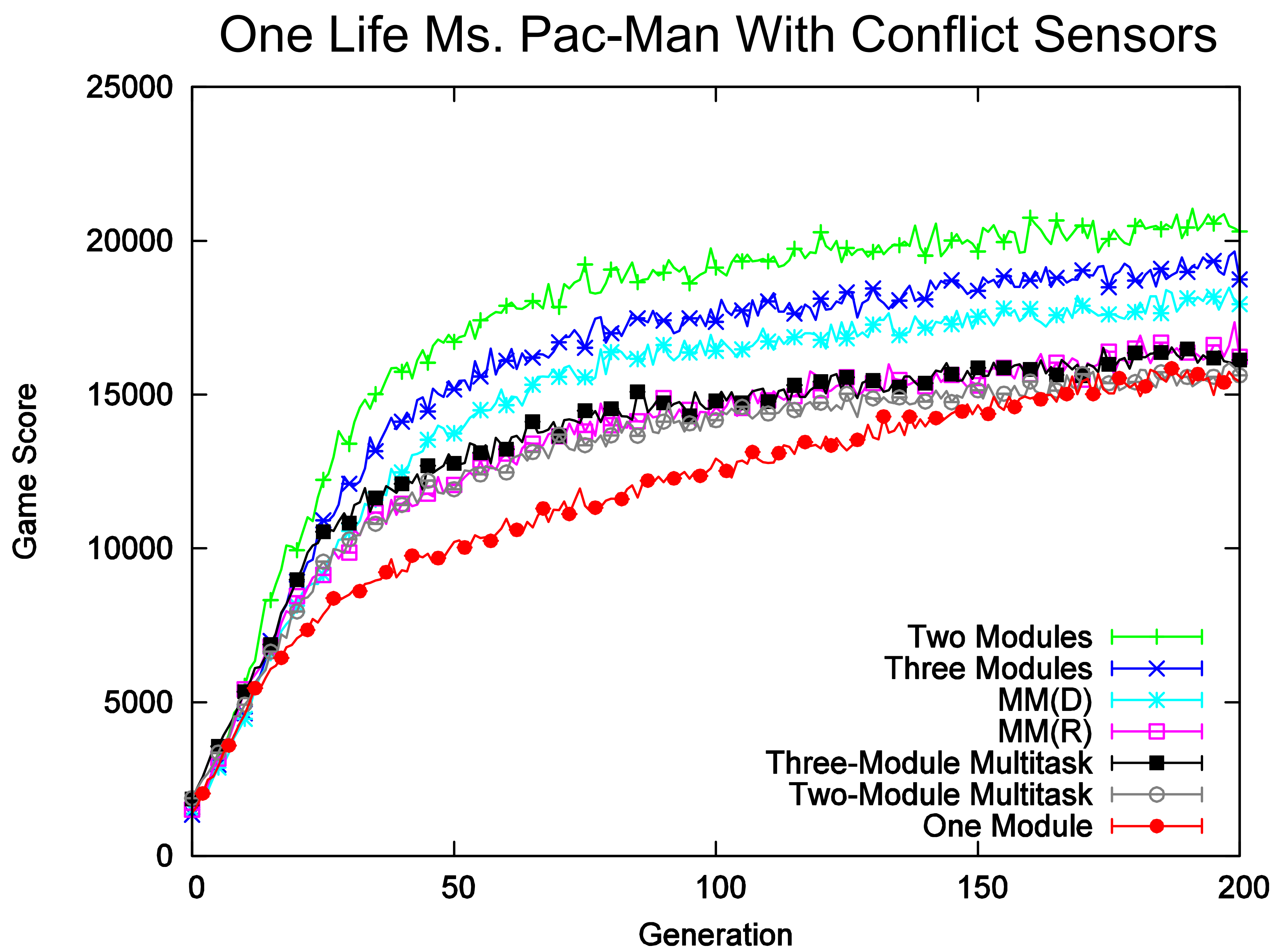
- Inspired by Brandstetter and Ahmadi (CIG 2012)
- Net with single output and direction-relative sensors
- Each time step, run net for each available direction
- Pick direction with highest net output



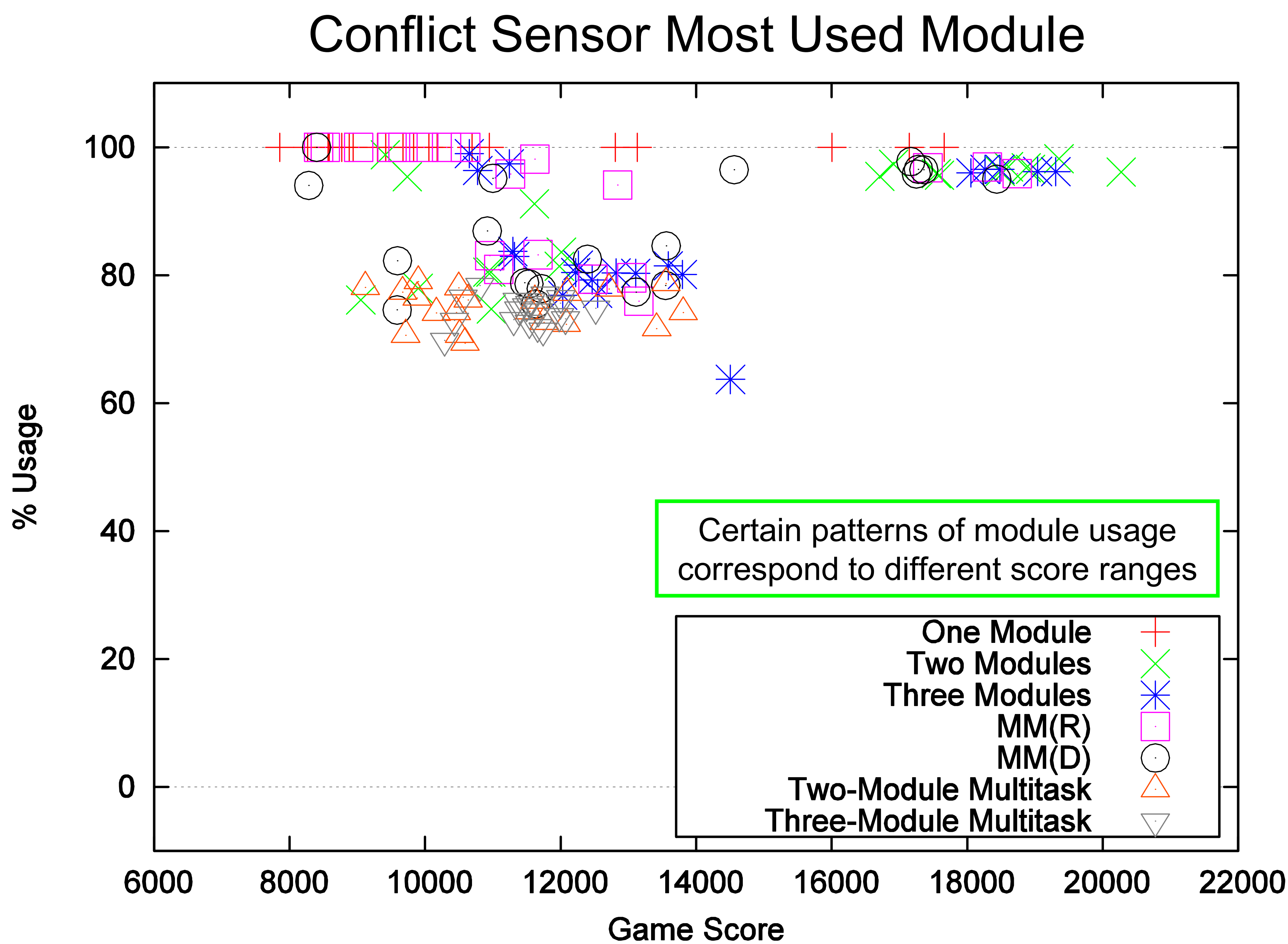
Module Setups

- Manually divide domain with Multitask
  - Two-Module: Threat/Any Edible
  - Three-Module: All Threat/All Edible/Mixed
- Discover new divisions with preference nodes
  - Two Modules, Three Modules, MM(R), MM(D)

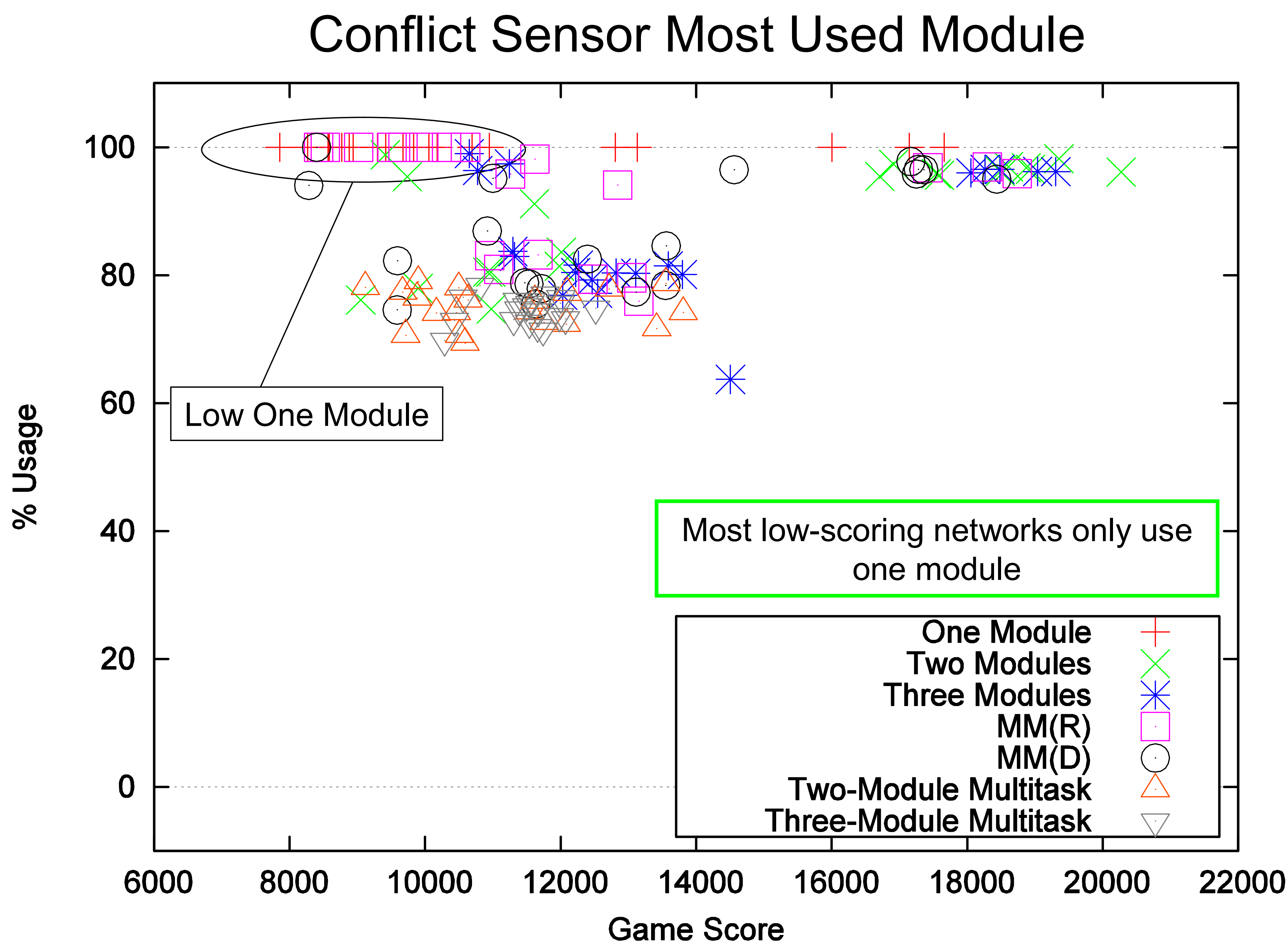




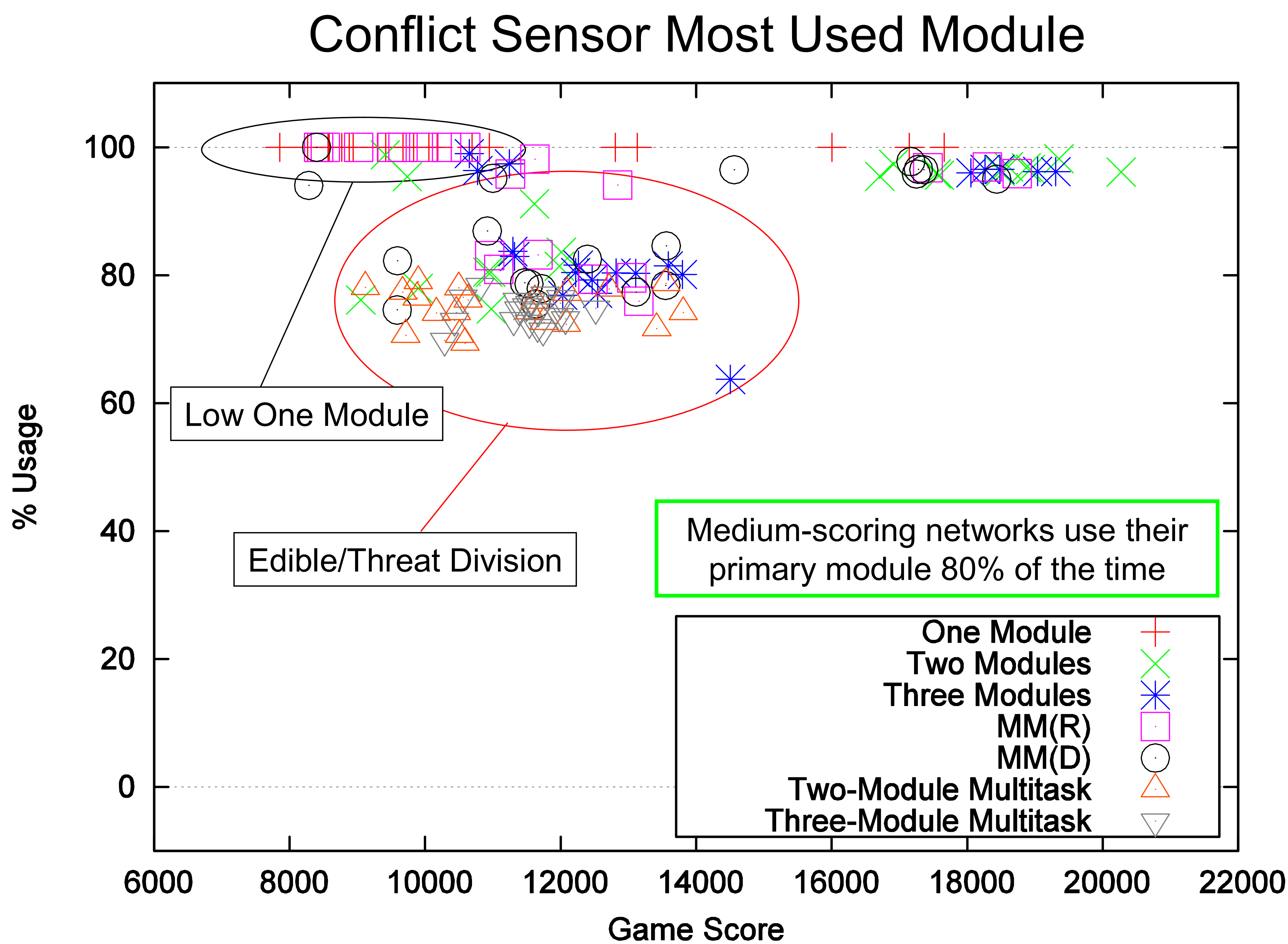
One Life Ms. Pac-Man With Conflict Sensors
Game Score
Generation
0
5000
10000
15000
20000
25000
0
50
100
150
200
Two Modules
Three Modules
MM(D)
MM(R)
Three-Module Multitask
Two-Module Multitask
One Module



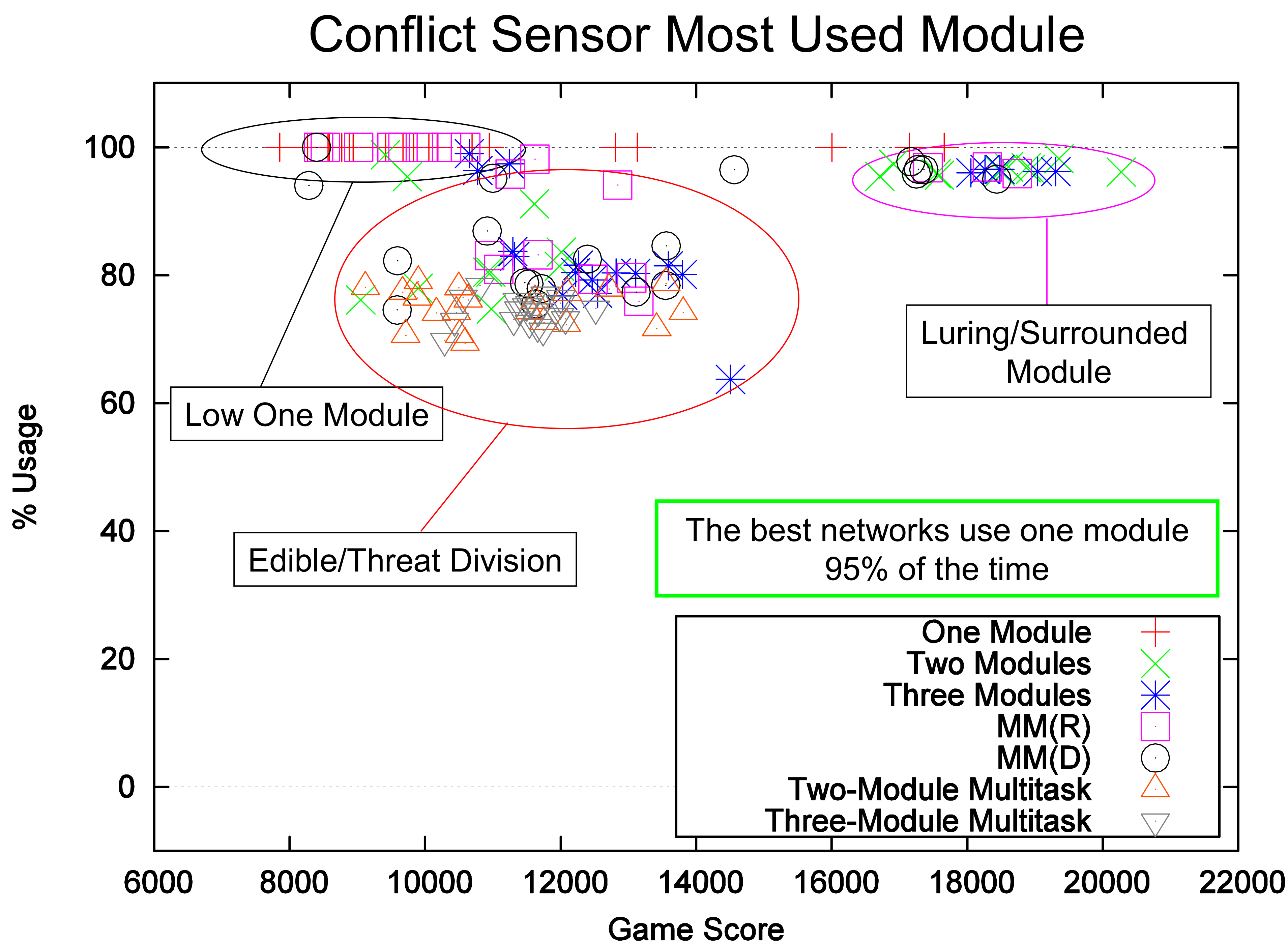
Conflict Sensor Most Used Module
% Usage
Game Score
100
80
60
40
20
0
6000
8000
10000
12000
14000
16000
18000
20000
22000
Certain patterns of module usage correspond to different score ranges
One Module
Two Modules
Three Modules
MM(R)
MM(D)
Two-Module Multitask
Three-Module Multitask



Conflict Sensor Most Used Module
% Usage
Game Score
100
80
60
40
20
0
6000
8000
10000
12000
14000
16000
18000
20000
22000
Low One Module
Most low-scoring networks only use one module
One Module
Two Modules
Three Modules
MM(R)
MM(D)
Two-Module Multitask
Three-Module Multitask



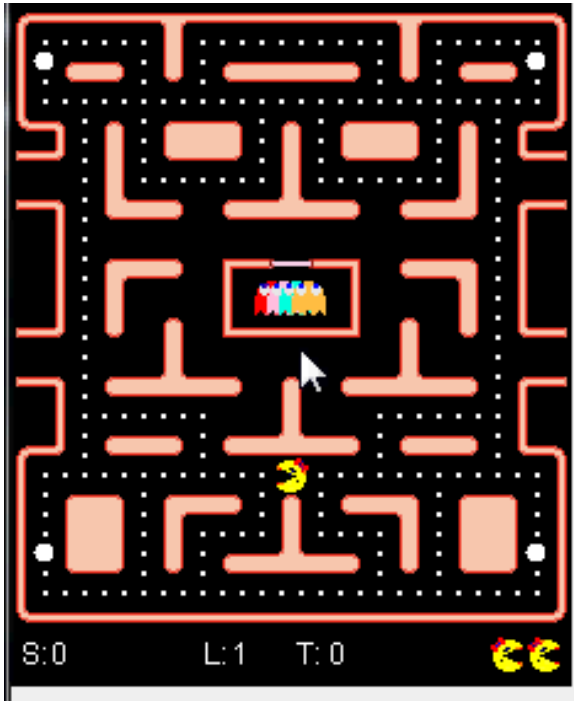
Conflict Sensor Most Used Module
100
80
60
40
20
0
6000
8000
10000
12000
14000
16000
18000
20000
22000
% Usage
Game Score
Low One Module
Edible/Threat Division
Medium-scoring networks use their primary module 80% of the time
One Module
Two Modules
Three Modules
MM(R)
MM(D)
Two-Module Multitask
Three-Module Multitask



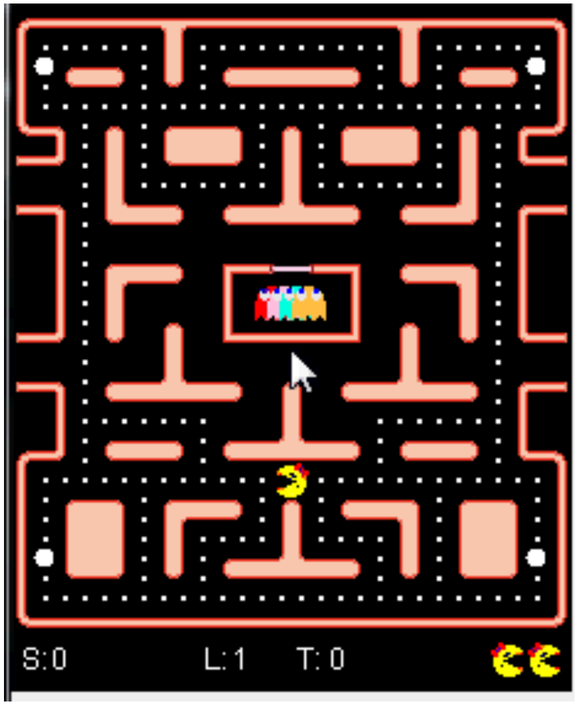
Conflict Sensor Most Used Module
% Usage
Game Score
100
80
60
40
20
0
6000
8000
10000
12000
14000
16000
18000
20000
22000
Low One Module
Edible/Threat Division
Luring/Surrounded Module
The best networks use one module 95% of the time
One Module
Two Modules
Three Modules
MM(R)
MM(D)
Two-Module Multitask
Three-Module Multitask

Full Game One Life Behavior

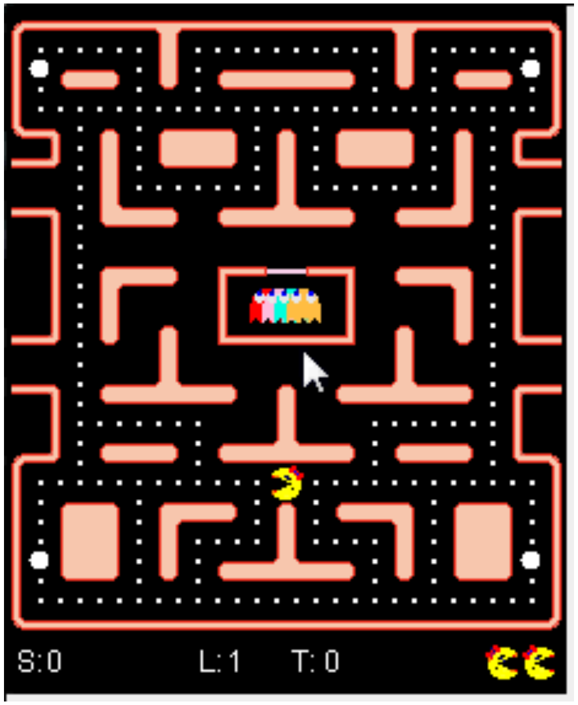
- Different colors are for different modules



Three-Module Multitask



Learned Edible/Threat Division



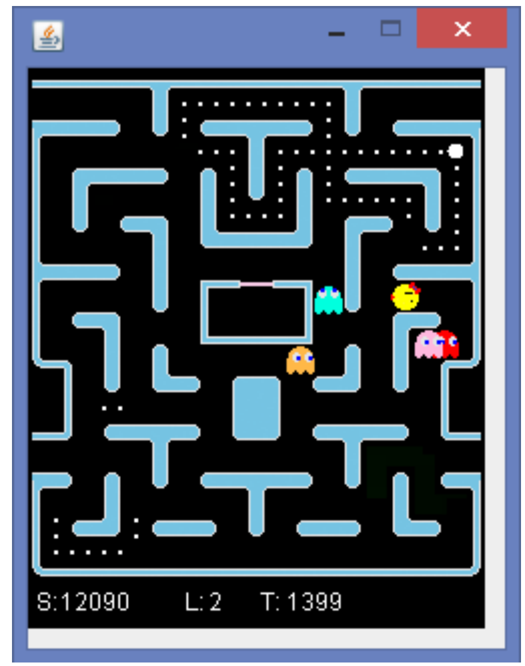
Learned Luring/Surrounded Module

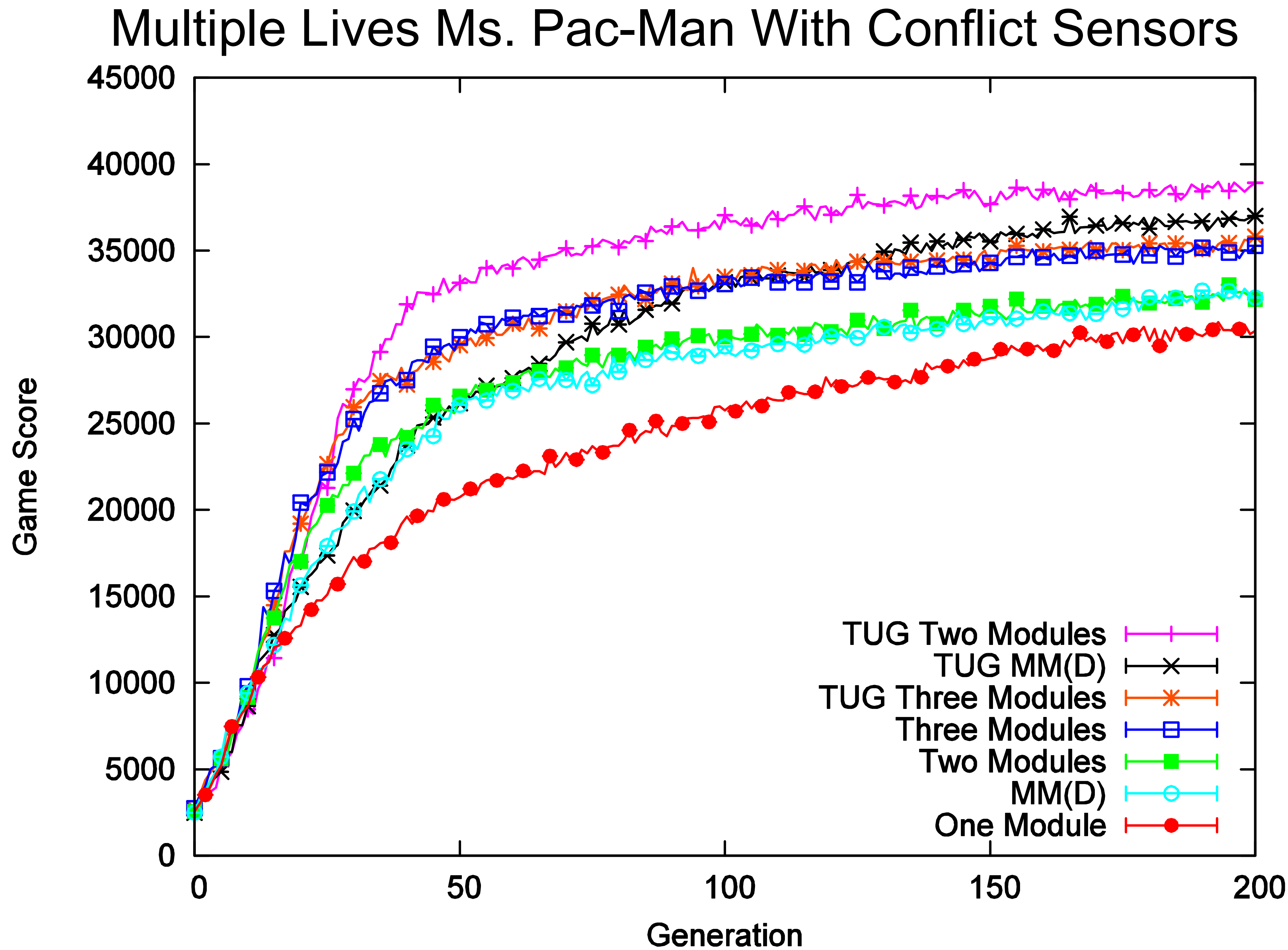
Full Game One Life Conclusion

- Obvious division is between edible and threat
  - But these tasks are blended
    - Strict Multitask divisions do not perform well
    - Preference neurons can learn when best to switch
- Better division: one module when surrounded
  - Very asymmetrical: surprising
  - Highest scoring runs use one module rarely
  - Module activates when Pac-Man almost surrounded
    - Often leads to eating power pill: luring
    - Helps Pac-Man escape in other risky situations

Full Game One Life Conclusion

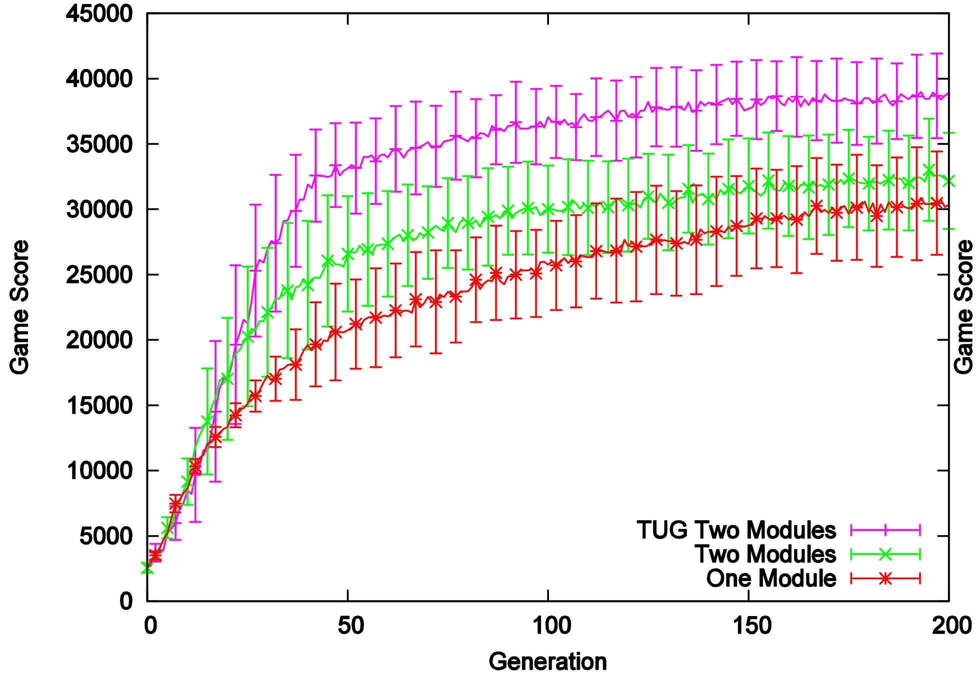
- Good divisions are harder to discover
  - Some modular champions use only one module
    - Particularly MM(R): new modules too random
- Are evaluations too harsh/noisy?
  - Easy to lose one life
  - Hard to eat all pills to progress
  - Discourages exploration
    - Hard to discover useful modules
  - Make search more forgiving
  - TUG to enhance performance

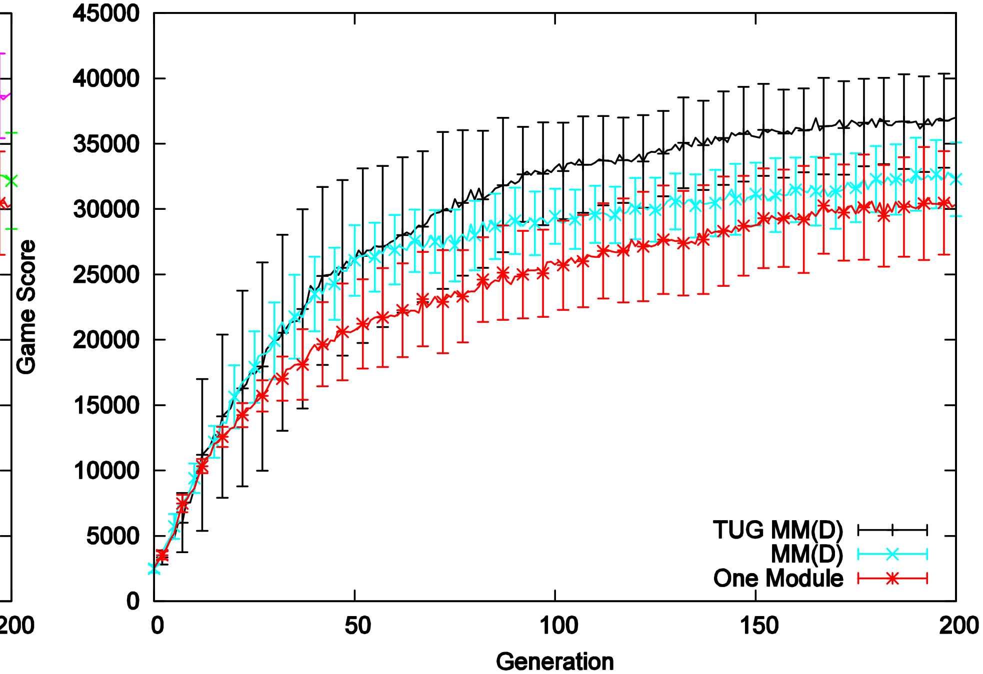




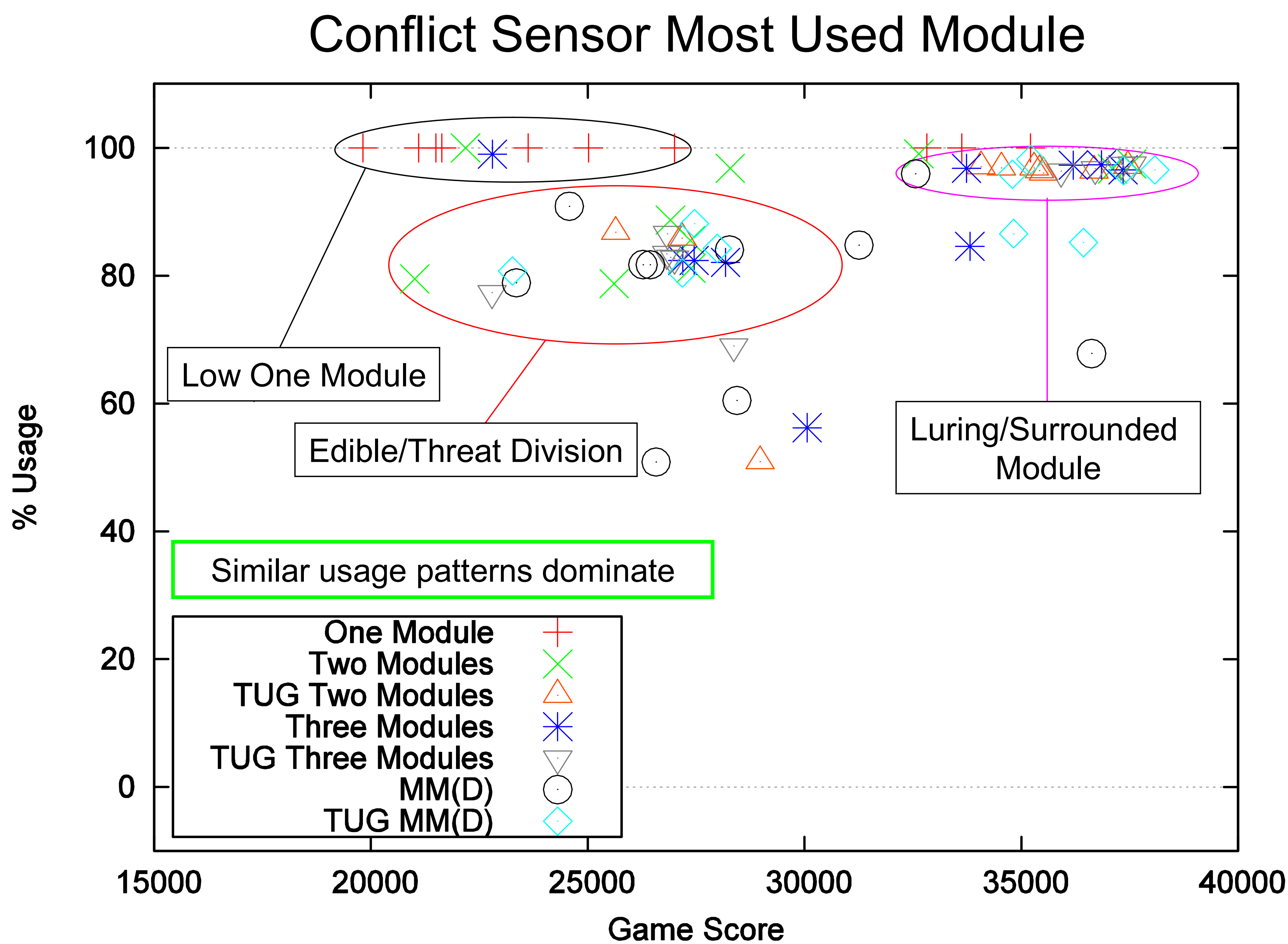
Multiple Lives Ms. Pac-Man With Conflict Sensors
Game Score
Generation
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
0
50
100
150
200
TUG Two Modules
TUG MM(D)
TUG Three Modules
Three Modules
Two Modules
MM(D)
One Module

Modular Networks With TUG

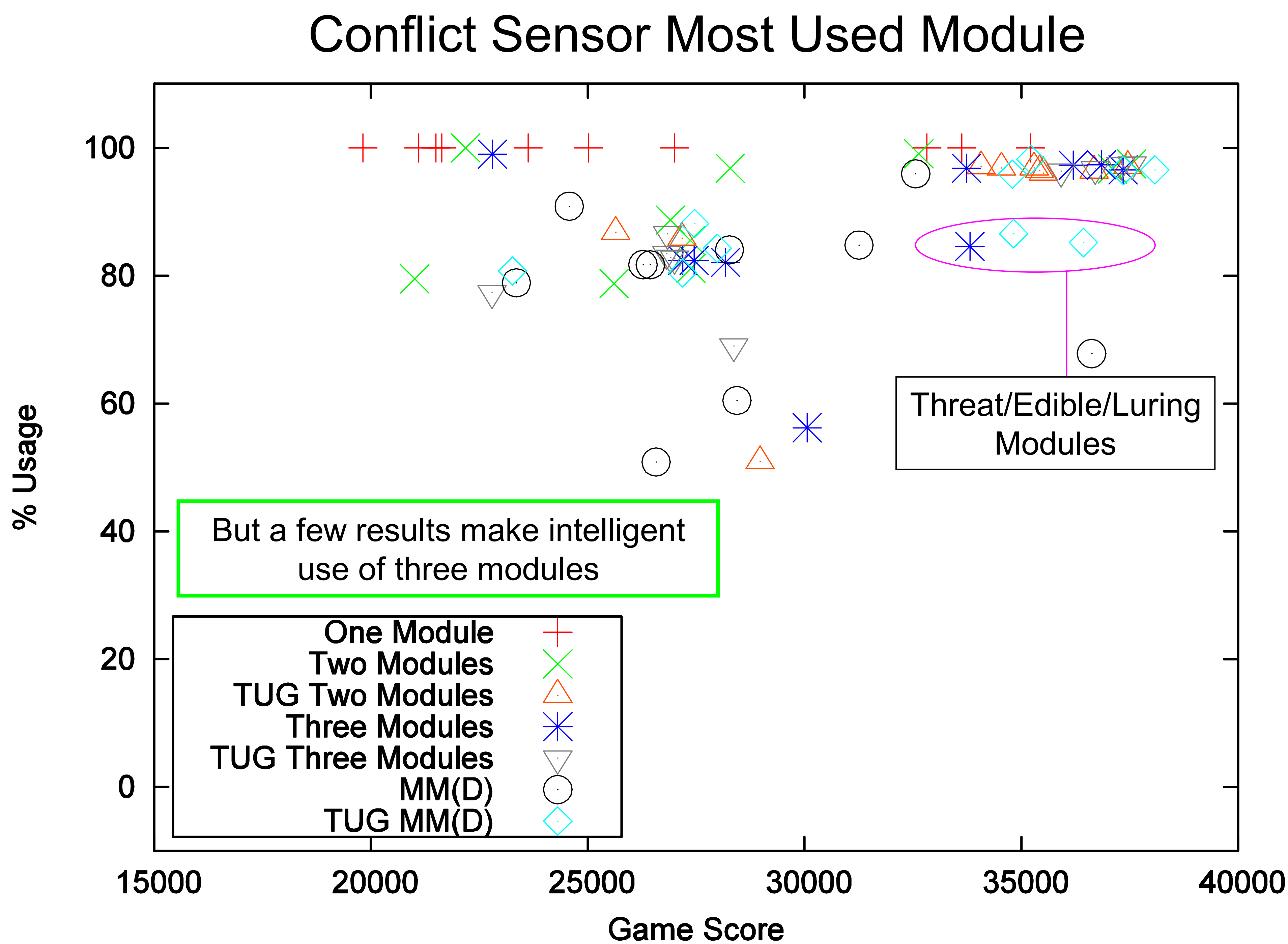




- Extra lives make evaluations easier for all methods
- TUG pushes modular performance significantly higher

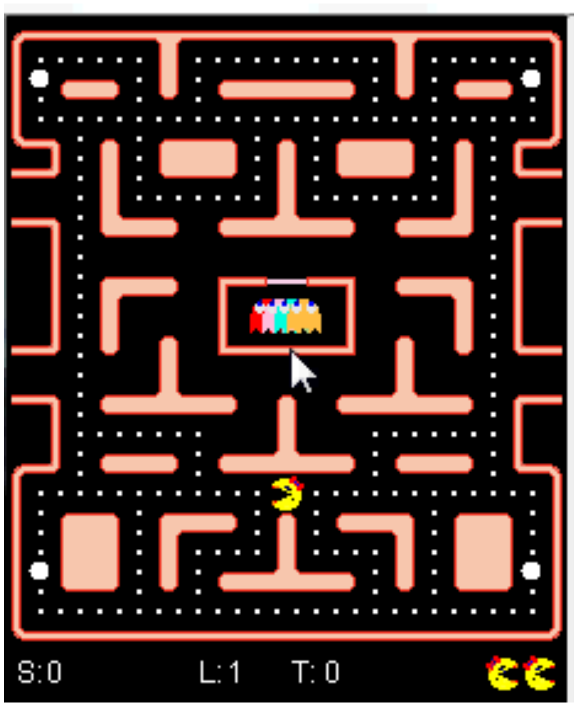


Conflict Sensor Most Used Module
100
80
60
40
20
0
15000
20000
25000
30000
35000
40000
% Usage
Game Score
Low One Module
Edible/Threat Division
Luring/Surrounded Module
Similar usage patterns dominate
One Module
Two Modules
TUG Two Modules
Three Modules
TUG Three Modules
MM(D)
TUG MM(D)


Conflict Sensor Most Used Module
% Usage
Game Score
100
80
60
40
20
0
15000
20000
25000
30000
35000
40000
Threat/Edible/Luring Modules
But a few results make intelligent use of three modules
One Module
Two Modules
TUG Two Modules
Three Modules
TUG Three Modules
MM(D)
TUG MM(D)

Full Game Multiple Lives Behavior

- Different colors are for different modules



One Module
Stalling

Three Modules:
Threat/Edible/Luring

Comparison with Other Work

Authors	Method	Eval Type	AVG	MAX
Alhejali and Lucas 2010	GP	Four Maze	16,014	44,560
Alhejali and Lucas 2011	GP+Camps	Four Maze	11,413	31,850
Best Dissertation Result	**Con, TUG, 3 Modules**	Four Maze	**37,549**	**48,130**

Recio et al. 2012	ACO	MPMvsG	36,031	43,467
Brandstetter and Ahmadi 2012	GP Direction	MPMvsG	19,198	33,420
Alhejali and Lucas 2013	MCTS	MPMvsG	28,117	62,630
	MCTS+GP	MPMvsG	32,641	69,010
Best Dissertation Result	**Split, 3 Modules**	MPMvsG	**68,524**	**90,890**

*The MPMvsG evaluation procedure makes the game easier, because Pac-Man gets to skip to the next level after 3000 time steps, allowing hard-to-reach pills to be ignored. This eval scheme also cycles the mazes for multiple visits, allowing for higher scores.

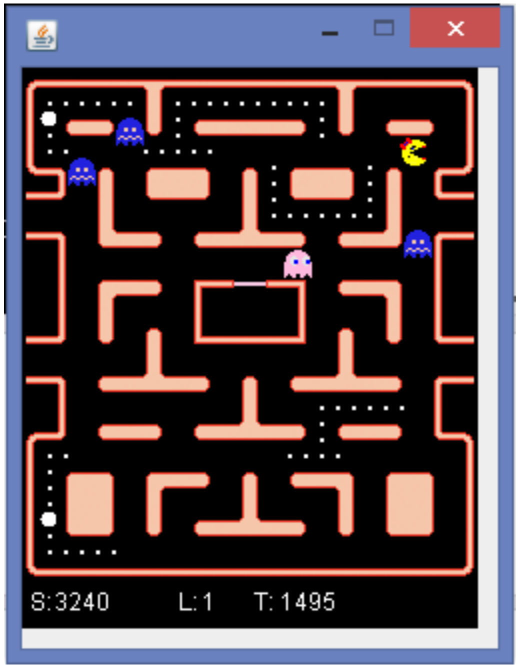
Outline

- Motivation
- Multimodal Behavior
- Methods
- Domains/Experiments
- Discussion/Conclusion

Discussion

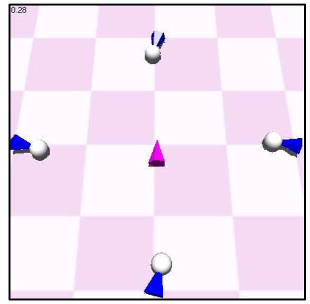
- Intelligent module divisions result in best results
  - Modular networks make learning separate modes easier
  - TUG helps take advantage of multiple modules
- Results are better than previous work
- Module division unexpected
  - Half of neural resources for seldom-used module (< 5%)
  - Rare situations can be very important
  - Some modules handle multiple modes
    - Pills, threats, edible ghosts

Future Work



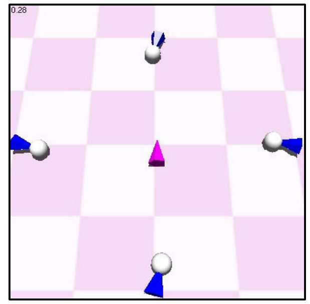
- Go beyond two modules
  - Issue with domain or evolution?
- More consistent success
  - How are objectives used? TUG a starting point
  - Behavioral diversity/novelty an option
- Multimodal behavior of teams
  - Ghost team in Pac-Man
- Physical simulation
  - Unreal Tournament, robotics

Conclusion



- Domains with clear task division
  - Variety of modular approaches are successful
- Domains with unclear task divisions
  - Surprising task divisions perform best
    - Multitask stops working well
  - Best divisions become much harder to learn
  - TUG makes learning more reliable
- Results in Ms. Pac-Man surpass previous evolved controllers, and other methods

Conclusion



- Contributions
  - Identified types of task divisions
    - Isolated, Interleaved, Blended
  - Split sensors impose a task division
    - Elaborated on in dissertation
  - Modular networks learn multiple behavioral modes
    - Learned task division better than human in blended tasks
  - TUG reaches higher scores more consistently
    - Extends benefits of multiobjective approach

Questions?