How To Do the Right Thing

Patti Maes
Overview

- Introduction
- Algorithm
- Example
- Results
Action Selection

- An autonomous agent with a variety of goals in a dynamic environment
- Complex world, limited resources
- How can we choose a ‘good enough’ next action?
Defining ‘good enough’

- Favors actions that are goal-oriented
- Favors actions relevant to the current situation
- Favors actions that contribute to the ongoing goal
- Looks ahead
- Is robust
- Is reactive
Society of the Mind theory

- Related to Subsumption Architecture
- Building an intelligent system as a society of interacting, mindless agents
- Local cooperation of agents to achieve goals
Controlling action in a distributed system

- Hard-wire the control flow among competence modules
- Use hierarchical structure
- ‘Bureaucratic’ competence modules
Or...

- Let the modules decide amongst themselves!
The hypothesis

- ‘Good enough’ action selection can be obtained by letting the competence modules activate and inhibit each other in the right way.

- Global parameters make it possible to smoothly mediate between various action selection criteria.
Overview

- Introduction
- Algorithm
- Example
- Results
Competence modules

- A competence module can be represented by a tuple, $(c_i, a_i, d_i, \alpha_i)$

- $c_i$ – preconditions
- $a_i$ – add list
- $d_i$ – delete list
- $\alpha_i$ – level of activation
Module example

(defmodule PICK-UP-SPRAYER
  :condition-list '(sprayer-somewhere hand-is-empty)
  :add-list '(sprayer-in-hand)
  :delete-list '(sprayer-somewhere hand-is-empty))
Competence modules (contd.)

- A competence module is *executable* at time $t$ if all of its preconditions are observed to be true at time $t$.

- An executable competence module whose activation-level surpasses a certain threshold may be *selected*. 
Competence module connections

- Successor links
- Predecessor links
- Conflicter links
Module activation by the environment and global goals

- Activation by the state
- Activation by the goals
  - Once-only goals
  - Permanent goals
- Inhibition by the protected goals
Module activation by other modules

- Activation of successors
- Activation of predecessors
- Inhibition of conflicters
Algorithm loop

- Impact of state, goals, and protected goals on activation level computed
- Activation and inhibition through successor links, predecessor links, and conflicter links computed
- Decay function ensures that overall activation level remains constant
Algorithm loop (contd.)

Competence module activated if:

- Executable
- Level of activation surpasses threshold
- Higher activation than other competence modules fulfilling these conditions

Otherwise, threshold is lowered
Four global parameters

- $\theta$ – threshold for becoming active
- $\phi$ – the amount of activation energy a proposition that is observed to be true injects
- $\gamma$ – the amount of activation energy a goal injects into the network
- $\delta$ – the amount of activation energy a protected goal takes away from the network
Additional details

- All input or removal of activation weighted by $1/n$

- Modules that achieve the same goal or modules that use the same precondition compete with one another to become active
Overview

- Introduction
- Algorithm
- Example
- Results
Example scenario

- Two handed-robot which must spray paint itself and sand a board

- Competence modules:
  - PICK-UP-BOARD
  - PICK-UP-SANDER
  - PICK-UP-SPRAYER
  - PUT-DOWN-BOARD
  - PUT-DOWN-SANDER
  - PUT-DOWN-SPRAYER
  - PUT-BOARD-IN-VISE
  - SAND-BOARD-IN-HAND
  - SAND-BOARD-IN-VISE
  - SPRAY-PAINT-SELF
Competence module interactions
Initial state and goals

- Initial state
  - $S(o) = (\text{hand-is-empty, hand-is-empty, sander-somewhere, sprayer-somewhere, board-somewhere, operational})$

- Initial goals
  - $G(o) = (\text{board-sanded, self-painted})$
Time 1

state gives PICK-UP-SANDER an extra activation of 3.3
state gives PICK-UP-SPRAYER an extra activation of 3.3
state gives PICK-UP-BOARD an extra activation of 3.3
...
goals give SAND-BOARD-IN-HAND an extra activation of 35.0
goals give SAND-BOARD-IN-VISE an extra activation of 35.0
goals give SPRAY-PAINT-SELF an extra activation of 70.0
...
Time 1 (contd.)

activation-levels of modules after decay:

activation-level PLACE-BOARD-IN-VISE: 0.0
activation-level SPRAY-PAINT-SELF: 73.3
activation-level SAND-BOARD-IN-HAND: 37.2
activation-level SAND-BOARD-IN-VISE: 37.2

... 

NO MODULE becoming active
threshold is lowered to 40.5
Time 2

... Sand-board-in-hand spreads 37.2 backward to Pick-up-board for board-in-hand
Sand-board-in-hand spread 37.2 backward to Pick-up-sander for sander-in-hand
Sand-board-in-hand decreases spray-paint-self with 26.6 for operational

... Pick-up-sander spreads 0.4 forward to Sand-board-in-hand for sander-in-hand
Pick-up-sander spreads 0.4 forward to Sand-board-in-vise for sander-in-hand

...
Activation levels over time
Overview

- Introduction
- Algorithm
- Example
- Results
What were we hoping to achieve?

- Goal-oriented actions
- Actions that contribute to the ongoing goal
- Actions adapt to the current situation
- Looks ahead
Remember the global parameters

- \( \theta \) – threshold for becoming active
- \( \phi \) – the amount of activation energy a proposition that is observed to be true injects
- \( \gamma \) – the amount of activation energy a goal injects into the network
- \( \delta \) – the amount of activation energy a protected goal takes away from the network
Goal-Orientedness

- Given that $g$ is a global goal of the system, then $\gamma$ of new activation energy is put into the modules that achieve this goal.

- Behavior can be made more or less goal-oriented in its selection by varying the ratio of $\gamma$ to $\phi$.
Bias towards ongoing plans

- Activation levels are not reinitialized every time a module is activated
- The history of past activation plays a role
Horizontal bias

- Horizontal bias favoring actions contributing to the current goal
Vertical bias

- Vertical bias favoring actions that contribute to a ‘brother’ goal (a subgoal of the same overall goal)
Adaptivity

- Because of continuous ‘reevaluation’, action selection behaviors adapts easily to unforeseen or changing situations

- System does not ‘replan’ at every timestamp

- Fault tolerant
Planning capabilities

- ‘Consider’ to some extent the effects of a sequence of actions
  - Goal input and backward spreading
  - State input and forward spreading
  - Local maxima avoided
  - Biases toward ongoing plans
Differences from classic AI planning

- No explicit representation of a single plan
- No centralized preprogrammed search process

Consequences
- Faster
- More robust
- But less ‘rational’
Thoughtfulness

- Behavior can be made more or less *thoughtful* by increasing the threshold $\theta$

- Allows the network to look ahead further

- Setting threshold too high problematic
Avoiding goal conflicts

- A bad ordering of actions can dramatically increase the number of actions necessary to achieve a goal.

- Modules in a network that undo a protected goal are weakened by a factor of $\delta$. 
Limits

- Oversimplified input-output relationship of a competence module
- Network does not maintain a record of its past ‘search’
- Unclear how to select values for the global parameters
Conclusions

- Demonstrates “the feasibility of using an activation/inhibition dynamic among competence modules to solve the problem of action selection for an autonomous agent operating a dynamic world”

- Global parameters allow us to tune the action selection behavior