Agents

Artificial Intelligence @ Allegheny College

Janyl Jumadinova

4-6 September, 2018
Week’s Material Coverage

4 September class

6 September class
What is AI?

<table>
<thead>
<tr>
<th>Systems that think like humans</th>
<th>Systems that think rationally</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems that act like humans</td>
<td>Systems that act rationally</td>
</tr>
</tbody>
</table>
Agents and environments

Agent

An agent is something that acts in an environment.

An agent acts intelligently if:

- its actions are appropriate for its goals and circumstances,
- it is flexible to changing environments and goals,
- it learns from experience,
- it makes appropriate choices given perceptual and computational limitations.
Agents and environments

Agent

An agent is something that acts in an environment.

An agent acts *intelligently* if:

- its actions are appropriate for its goals and circumstances,
- it is flexible to changing environments and goals,
- it learns from experience,
- it makes appropriate choices given perceptual and computational limitations.
Agents and environments

Agents include humans, robots, softbots, thermostats, etc. The agent function maps from percept histories to actions:

\[ f : P^* \rightarrow A \]

The agent program runs on the physical architecture to produce \( f \).
Agents and environments

Agents include humans, robots, softbots, thermostats, etc.
Agents include humans, robots, softbots, thermostats, etc. The agent function maps from percept histories to actions:

\[ f : \mathcal{P}^* \rightarrow A \]

The agent program runs on the physical architecture to produce \( f \).
A vacuum cleaner agent

Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left, Right, Suck, NoOp$
## A vacuum cleaner agent

What is the **right** function?
What makes an agent good or bad, intelligent or stupid?

<table>
<thead>
<tr>
<th>Percept sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B, Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[B, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

Janyl Jumadinova  
Agents  
4-6 September, 2018  
7 / 52
Agents and environments

For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance.
Agents and environments

For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance.

Caveat: computational limitations make perfect rationality unachievable.
Agents and environments

For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance.

**Caveat:** computational limitations make perfect rationality unachievable. → design best program for given machine resources.
Autonomy: The ability to operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
Restricting the definition of an **agent**: an ideal agent

- **Autonomy**: The ability to operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.

- **(Structural) Reactivity**: The ability to perceive the environment, and respond regularly to changes that occur in it.
Restricting the definition of an **agent**: an ideal agent

- **Autonomy**: The ability to operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.

- (Structural) **reactivity**: The ability to perceive the environment, and respond regularly to changes that occur in it.

- **Social Ability**: The ability to interact with other agents (and possibly humans).
Restricting the definition of an **agent**: an ideal agent

- **Autonomy**: The ability to operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.

- (Structural) **Reactivity**: The ability to perceive the environment, and respond regularly to changes that occur in it.

- **Social Ability**: The ability to interact with other agents (and possibly humans).

- **Pro-Activity**: The ability to exhibit goal-directed behavior by taking the initiative instead of just acting in response.
Restricting the definition of an \textbf{agent}: an ideal agent

Other attributes:

- **Mobility**: The ability to move around an electronic network.
Other attributes:

- **Mobility**: The ability to move around an electronic network.
- **Veracity**: The assumption of not communicating false information knowingly.
Restricting the definition of an **agent**: an ideal agent

Other attributes:

- **Mobility**: The ability to move around an electronic network.
- **Veracity**: The assumption of not communicating false information knowingly.
- **Benevolence**: The assumption of not having conflicting goals.
Restricting the definition of an agent: an ideal agent

Other attributes:

- **Mobility**: The ability to move around an electronic network.
- **Veracity**: The assumption of not communicating false information knowingly.
- **Benevolence**: The assumption of not having conflicting goals.
- **Rationality**: The assumption of acting with a view to achieve its goals, instead of preventing them.
Agents vs. Objects

<table>
<thead>
<tr>
<th></th>
<th>OOP</th>
<th>AOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic unit</td>
<td>object</td>
<td>agent</td>
</tr>
<tr>
<td>Parameters defining state of basic unit</td>
<td>unconstrained</td>
<td>beliefs, commitments, capabilities, choices, …</td>
</tr>
<tr>
<td>Process of computation</td>
<td>message passing and response methods</td>
<td>message passing and response methods</td>
</tr>
<tr>
<td>Types of message</td>
<td>unconstrained</td>
<td>inform, request, offer, promise, decline, …</td>
</tr>
<tr>
<td>Constraints on methods</td>
<td>none</td>
<td>honesty, consistency, …</td>
</tr>
</tbody>
</table>

“Agent-Oriented Programming”, Y. Shoham
Agents have the quality of **volition**.
- using AI techniques, intelligent agents are able to judge their results, and then modify their behavior (and thus their own internal structure) to improve their perceived fitness.
Agents vs. Objects

- Agents have the quality of **volition**.
  - using AI techniques, intelligent agents are able to judge their results, and then modify their behavior (and thus their own internal structure) to improve their perceived fitness.

- Objects are abstractions of things like invoices. Agents are abstractions of intelligent beings – they are essentially **anthropomorphic**.
  Note that this does not mean that agents are intelligent in the human sense, only that they are modeled after an anthropomorphic architecture, with beliefs, desires, etc.
- Design an object-oriented solution and an agent-oriented solution for a car wash task.
- Identify why it is an object-oriented or an agent-oriented solution.
- List agents and objects for both solutions.
A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date.

A system is rational if it does the “right thing”, given what it knows.
Rationality

Fixed performance measure evaluates the environment sequence

- one point per square cleaned up in time $T$?
- one point per clean square per time step, minus one per move?
- penalize for $>k$ dirty squares?
Rationality

Fixed **performance measure** evaluates the *environment sequence*

- one point per square cleaned up in time $T$?
- one point per clean square per time step, minus one per move?
- penalize for $> k$ dirty squares?

A **rational agent** chooses whichever action maximizes the **expected** value of the performance measure **given the percept sequence to date**
Rational $\neq$ omniscient
- percepts may not supply all relevant information
Rational \neq omniscient
- percepts may not supply all relevant information
Rational \neq clairvoyant
- action outcomes may not be as expected
Rationality

Rational ≠ omniscient
– percepts may not supply all relevant information
Rational ≠ clairvoyant
– action outcomes may not be as expected
Hence, rational ≠ successful
Rational $\neq$ omniscient
– percepts may not supply all relevant information
Rational $\neq$ clairvoyant
– action outcomes may not be as expected
Hence, rational $\neq$ successful
Rational $\implies$ exploration, learning, autonomy
To design a rational agent, we must specify the task environment:

- **Performance measure**
- **Environment**
- **Actuators**
- **Sensors**
To design a rational agent, we must specify the task environment.

Consider, e.g., the task of designing an automated taxi:

Performance measure: safety, destination, profits, legality, comfort, ...
To design a rational agent, we must specify the task environment.

Consider, e.g., the task of designing an automated taxi:

- **Performance measure**: safety, destination, profits, legality, comfort, . . .
- **Environment**: US streets/freeways, traffic, pedestrians, weather, . . .
- **Actuators**: steering, accelerator, brake, horn, speaker/display, . . .
- **Sensors**: video, accelerometers, gauges, engine sensors, keyboard, GPS, . . .
To design a rational agent, we must specify the **task environment**.

Consider, e.g., the task of designing an **automated taxi**:
- **Performance measure** safety, destination, profits, legality, comfort, . . .
- **Environment** US streets/freeways, traffic, pedestrians, weather, . . .
To design a rational agent, we must specify the task environment.

Consider, e.g., the task of designing an automated taxi:
Performance measure safety, destination, profits, legality, comfort, . . .
Environment US streets/freeways, traffic, pedestrians, weather, . . .
Actuators
To design a rational agent, we must specify the task environment.

Consider, e.g., the task of designing an automated taxi:

**Performance measure** safety, destination, profits, legality, comfort, . . .

**Environment** US streets/freeways, traffic, pedestrians, weather, . . .

**Actuators** steering, accelerator, brake, horn, speaker/display, . . .
To design a rational agent, we must specify the task environment.

Consider, e.g., the task of designing an automated taxi:
- **Performance measure** safety, destination, profits, legality, comfort, . . .
- **Environment** US streets/freeways, traffic, pedestrians, weather, . . .
- **Actuators** steering, accelerator, brake, horn, speaker/display, . . .
- **Sensors**
To design a rational agent, we must specify the task environment.

Consider, e.g., the task of designing an automated taxi:
Performance measure safety, destination, profits, legality, comfort, ... 
Environment US streets/freeways, traffic, pedestrians, weather, ... 
Actuators steering, accelerator, brake, horn, speaker/display, ... 
Sensors video, accelerometers, gauges, engine sensors, keyboard, GPS, ...
Internet shopping agent?

Performance measure
price, quality, appropriateness, efficiency, . . .

Environment
current and future WWW sites, vendors, shippers, . . .

Actuators
display to user, follow URL, fill in form, . . .

Sensors
HTML pages (text, graphics, scripts), . . .
Internet shopping agent?

Performance measure
Internet shopping agent?

Performance measure price, quality, appropriateness, efficiency, . . .
Internet shopping agent?

Performance measure price, quality, appropriateness, efficiency, . . .

Environment
Internet shopping agent?

Performance measure price, quality, appropriateness, efficiency, . . .

Environment current and future WWW sites, vendors, shippers, . . .
Internet shopping agent?

Performance measure price, quality, appropriateness, efficiency, . . .

Environment current and future WWW sites, vendors, shippers, . . .

Actuators
Internet shopping agent?

Performance measure price, quality, appropriateness, efficiency, . . .
Environment current and future WWW sites, vendors, shippers, . . .
Actuators display to user, follow URL, fill in form, . . .
Internet shopping agent?

Performance measure price, quality, appropriateness, efficiency, . . .
Environment current and future WWW sites, vendors, shippers, . . .
Actuators display to user, follow URL, fill in form, . . .
Sensors
Internet shopping agent?

Performance measure price, quality, appropriateness, efficiency, ...  
Environment current and future WWW sites, vendors, shippers, ...  
Actuators display to user, follow URL, fill in form, ...  
Sensors HTML pages (text, graphics, scripts), ...
Environment Types

- **Fully Observable**: the agent can observe the state of the world, vs.
- **Partially Observable**: there can be a number states that are possible given the agent’s observations
Environment Types

- **Fully Observable**: the agent can observe the state of the world, vs.
- **Partially Observable**: there can be a number states that are possible given the agent’s observations
- **Deterministic**: the resulting state is determined from the action and the state, vs.
- **Stochastic**: there is uncertainty about the resulting state
Environment Types

- **Fully Observable**: the agent can observe the state of the world, *vs.*
- **Partially Observable**: there can be a number of states that are possible given the agent’s observations

- **Deterministic**: the resulting state is determined from the action and the state, *vs.*
- **Stochastic**: there is uncertainty about the resulting state

- **Episodic**: agent’s experience is divided into atomic episodes, *vs.*
- **Sequential**: the current decision could affect all future decisions
Environment Types

- **Static**: environment does not change, vs. **Dynamic**: the environment can change while an agent is deliberating, vs.
- **Semi**: the environment itself does not change with the passage of time but the agent’s performance score does
Environment Types

- **Static**: environment does not change, vs. **Dynamic**: the environment can change while an agent is deliberating, vs.

- **Semi**: the environment itself does not change with the passage of time but the agent’s performance score does.

- **Discrete vs. Continuous**: applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
Environment Types

- **Static**: environment does not change, *vs.*
  - **Dynamic**: the environment can change while an agent is deliberating, *vs.*
- **Semi**: the environment itself does not change with the passage of time but the agent’s performance score does
- **Discrete vs. Continuous**: applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent
- **Single-agent vs. Multi-agent**
Environment Types

- **Static**: environment does not change, *vs.*
  - **Dynamic**: the environment can change while an agent is deliberating, *vs.*
- **Semi**: the environment itself does not change with the passage of time but the agent’s performance score does
- **Discrete vs. Continuous**: applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent
- **Single-agent vs. Multi-agent**

*The environment type largely determines the agent design*
## Environment Types

<table>
<thead>
<tr>
<th>Feature</th>
<th>Solitaire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
</tr>
<tr>
<td>Static</td>
<td>Yes</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
</tr>
<tr>
<td>Single-agent</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Environment Types

<table>
<thead>
<tr>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Single-agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solitaire</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Internet shopping</td>
<td>No</td>
<td>Partly</td>
<td>No</td>
<td>Semi</td>
<td>Yes (except auctions)</td>
</tr>
</tbody>
</table>
Environment Types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static</td>
<td>Yes</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single-agent</td>
<td>Yes</td>
<td>Yes (except auctions)</td>
<td>No</td>
</tr>
</tbody>
</table>
## Environment Types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static</td>
<td>Yes</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single-agent</td>
<td>Yes</td>
<td>Yes (except auctions)</td>
<td>No</td>
</tr>
</tbody>
</table>

The real world is partially observable, stochastic, sequential, dynamic, continuous, multi-agent
Getting to an ideal agent

Agent types in order of increasing generality:

- simple reflex agents
- reflex agents with state
- goal-based agents
- utility-based agents
- learning agents
Simple Reflex Agent

Agent

Sensors

What the world is like now

Condition–action rules

What action I should do now

Actuators

Environment
Simple Reflex Agent - An Example

function Reflex-Vacuum-Agent([location, status]) returns an action

if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
Reflex Agent with State

Agent

- State
- How the world evolves
- What my actions do
- Condition–action rules

Sensors

Environment

- What the world is like now
- What action I should do now

Actuators
Reflex Agent with State - An Example

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
static: last_A, last_B, numbers, initially ∞
    if status = Dirty then ...
```
Goal-based Agent

Agent

Environment

Sensors

State

How the world evolves

What the world is like now

What my actions do

What it will be like if I do action A

Goals

What action I should do now

Actuators
Utility-based Agent

- **Agent**
  - State
  - How the world evolves
  - What my actions do
  - Utility

- **Environment**
  - Sensors
  - What the world is like now
  - What it will be like if I do action A
  - How happy I will be in such a state
  - What action I should do now

- **Actuators**

Janyl Jumadinova

Agents

4-6 September, 2018 29 / 52
All the previous agents can be turned into learning agents.
Agents interact with environments through actuators and sensors.
Agents interact with environments through actuators and sensors.

The agent function describes what the agent does in all circumstances.
Agent Summary

- Agents interact with environments through actuators and sensors.
- The **agent function** describes what the agent does in all circumstances.
- The **performance measure** evaluates the environment sequence.

Environments are categorized along several dimensions: observable, deterministic, episodic, static, discrete, single-agent.

Several basic agent architectures exist: reflex, reflex with state, goal-based, utility-based, learning.
Agents interact with environments through actuators and sensors

The agent function describes what the agent does in all circumstances

The performance measure evaluates the environment sequence

A perfectly rational agent maximizes expected performance
Agents interact with environments through actuators and sensors

The agent function describes what the agent does in all circumstances

The performance measure evaluates the environment sequence

A perfectly rational agent maximizes expected performance

Agent programs implement (some) agent functions
Agent Summary

- Agents interact with environments through actuators and sensors
- The **agent function** describes what the agent does in all circumstances
- The **performance measure** evaluates the environment sequence
- A **perfectly rational** agent maximizes expected performance
- **Agent programs** implement (some) agent functions
- **PEAS** descriptions define task environments
Agents interact with environments through actuators and sensors.

The agent function describes what the agent does in all circumstances.

The performance measure evaluates the environment sequence.

A perfectly rational agent maximizes expected performance.

Agent programs implement (some) agent functions.

PEAS descriptions define task environments.

Environments are categorized along several dimensions: observable? deterministic? episodic? static? discrete? single-agent?
Agents interact with environments through actuators and sensors.

The agent function describes what the agent does in all circumstances.

The performance measure evaluates the environment sequence.

A perfectly rational agent maximizes expected performance.

Agent programs implement (some) agent functions.

PEAS descriptions define task environments.

Environments are categorized along several dimensions: observable? deterministic? episodic? static? discrete? single-agent?

Several basic agent architectures exist: reflex, reflex with state, goal-based, utility-based, learning.
NetLogo

the Agent Based Modeling (ABM) language
NetLogo

the Agent Based Modeling (ABM) language

A language built specifically for agent based modeling
- a modeling environment
- interactively adjust parameters
- feedback through plots and visualizations
What is Modeling?

- A simplified mathematical representation of a system
- Only include features essential to explaining phenomenon of interest
NetLogo

Model Types
- deterministic
- stochastic
- evolving
Deterministic: flocking
Stochastic: network growth
Stochastic: termites
Evolving: genetic algorithms
NetLogo is an IDE (integrated development environment) that can be used to create programs that simulate natural and social phenomena.
NetLogo

- NetLogo is an IDE (integrated development environment) that can be used to create programs that simulate natural and social phenomena.
- NetLogo is particularly well suited for modeling complex systems that develop over time.
NetLogo

- NetLogo is an IDE (integrated development environment) that can be used to create programs that simulate natural and social phenomena.
- NetLogo is particularly well suited for modeling complex systems that develop over time.
- Using NetLogo you can create programs containing thousands of agents (called “turtles”) all operating independently.
NetLogo

- NetLogo is an IDE (integrated development environment) that can be used to create programs that simulate natural and social phenomena.
- NetLogo is particularly well suited for modeling complex systems that develop over time.
- Using NetLogo you can create programs containing thousands of agents (called “turtles”) all operating independently.
- For us, NetLogo will serve as another programming environment in which to explore the Imperative, Procedural and Object-Oriented Paradigms.
Resources

- Documentation
  http://ccl.northwestern.edu/netlogo/docs/

- Quick Guide by Luis Izquierdo
  http://luis.izqui.org/resources/NetLogo-6-0-QuickGuide.pdf

- Dictionary (full list of commands and descriptions of them)
  http://ccl.northwestern.edu/netlogo/docs/dictionary.html
Example: Romania

On holiday in Romania; currently in Arad.
Flight leaves tomorrow from Bucharest

**Formulate goal:** be in Bucharest

**Formulate problem:**

**states:** various cities
**actions:** drive between cities

**Find solution:** sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest
Example: Romania

Can use Tree Search Algorithms (BFS, DFS)

Special cases: greedy search, $A^*$ search
Romania with step costs in km

<table>
<thead>
<tr>
<th>City</th>
<th>Bucharest</th>
<th>Giurgiu</th>
<th>Ursiceni</th>
<th>Hirsova</th>
<th>Bucharest</th>
<th>Neamt</th>
<th>Iasi</th>
<th>Bucharest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arad</td>
<td>366</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>366</td>
</tr>
<tr>
<td>Bucharest</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>92</td>
<td>98</td>
<td>142</td>
</tr>
<tr>
<td>Craiova</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>92</td>
<td>87</td>
<td>85</td>
</tr>
<tr>
<td>Dobrosha</td>
<td>242</td>
<td></td>
<td></td>
<td></td>
<td>151</td>
<td>87</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Eforie</td>
<td>161</td>
<td></td>
<td></td>
<td></td>
<td>241</td>
<td>87</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Fagaras</td>
<td>178</td>
<td></td>
<td></td>
<td></td>
<td>140</td>
<td>211</td>
<td>92</td>
<td>85</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
<td></td>
<td></td>
<td></td>
<td>77</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Hirsova</td>
<td>151</td>
<td></td>
<td></td>
<td></td>
<td>99</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Iasi</td>
<td>226</td>
<td></td>
<td></td>
<td></td>
<td>118</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Lugoj</td>
<td>244</td>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Mehadia</td>
<td>241</td>
<td></td>
<td></td>
<td></td>
<td>70</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Neamt</td>
<td>234</td>
<td></td>
<td></td>
<td></td>
<td>146</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
<td></td>
<td></td>
<td></td>
<td>120</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Pitesti</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td>138</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Rimnicu Vilcea</td>
<td>193</td>
<td></td>
<td></td>
<td></td>
<td>211</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
<td></td>
<td></td>
<td></td>
<td>75</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Timisoara</td>
<td>329</td>
<td></td>
<td></td>
<td></td>
<td>120</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Urziceni</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td>151</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Vaslui</td>
<td>199</td>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td>92</td>
<td>92</td>
<td>90</td>
</tr>
</tbody>
</table>
A* search

Idea:

avoid expanding paths that are already expensive
A* search

Idea:
avoid expanding paths that are already expensive

- Evaluation function $f(n) = g(n) + h(n)$
- $g(n) =$ cost so far to reach $n$
- $h(n) =$ estimated cost to goal from $n$
- $f(n) =$ estimated total cost of path through $n$ to goal

Romania with step costs
Example: Romania
A* Search

Arad
366 = 0 + 366
A* Search

Janyl Jumadinova
Agents
4-6 September, 2018
47 / 52
A* Search

Arad
Sibiu
Timisoara
Zerind

646 = 280 + 366
415 = 239 + 176
671 = 291 + 380
413 = 220 + 193

447 = 118 + 329
449 = 75 + 374
415 = 239 + 176
671 = 291 + 380

449 = 75 + 374
A* Search

Agents

4-6 September, 2018
A* Search

Agent 4-6 September, 2018
A* Search

Janyl Jumadinova

Agents

4-6 September, 2018
A* Search in Netlogo

A* demo in Netlogo:
http://ccl.northwestern.edu/netlogo/models/community/Astardemo1