Автономная навигация мобильных роботов

A. Matveev

almat1712@yahoo.com

Department of Mathematics and Mechanics,
Saint Petersburg state University,

Scientific and Technological
University “Sirius”

Тема 3: Алгоритмы навигации на основе обширных данных о сцене

Общие определения и примеры

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

Decomposition of the environment

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
  - rectangular worlds
  - spherical worlds
  - polygonal worlds
  - semi-algebraic worlds

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

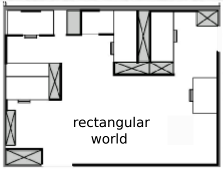
Decomposition of the environment

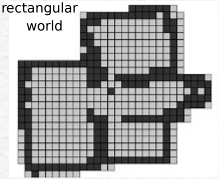
Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
  - rectangular worlds
  - spherical worlds
  - polygonal worlds
  - semi-algebraic worlds





Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

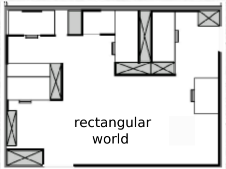
Decomposition of the environment

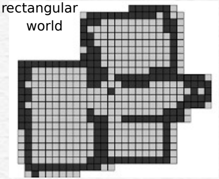
Localization

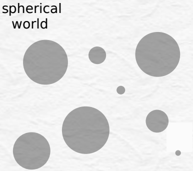
- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
  - rectangular worlds
  - spherical worlds
  - polygonal worlds
  - semi-algebraic worlds







Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

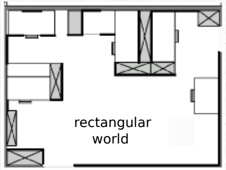
Decomposition of the environment

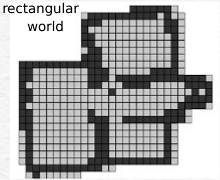
Localization

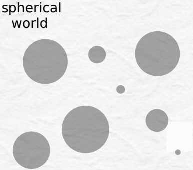
- Determine the targeted object in the map
- Find its own position in the map

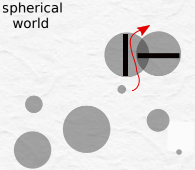
Some popular types of maps

- Continuous geometric representations
  - rectangular worlds
  - spherical worlds
  - polygonal worlds
  - semi-algebraic worlds









Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

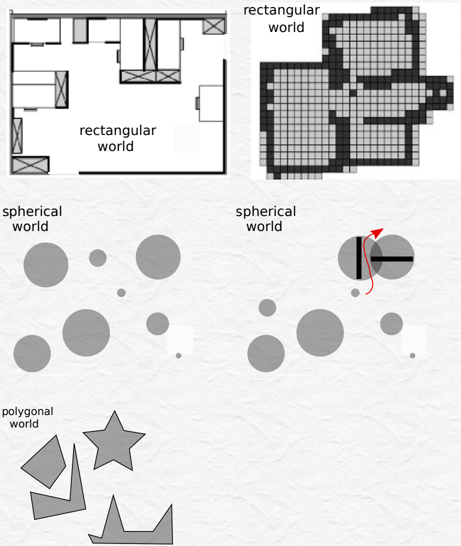
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
  - rectangular worlds
  - spherical worlds
  - polygonal worlds
  - semi-algebraic worlds



Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

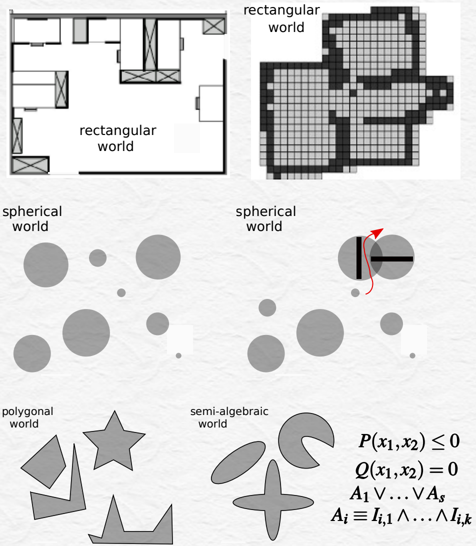
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
  - rectangular worlds
  - spherical worlds
  - polygonal worlds
  - semi-algebraic worlds



Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

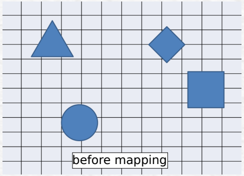
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid



Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

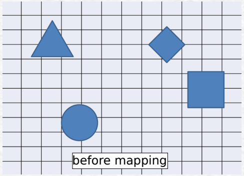
Decomposition of the environment

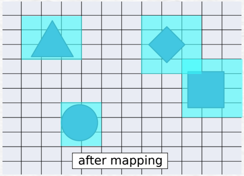
Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid





Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

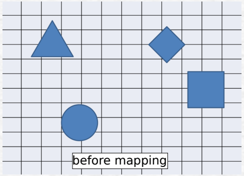
Decomposition of the environment

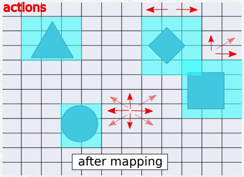
Localization

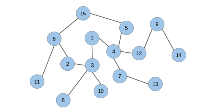
- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid







occupancy grid representation

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

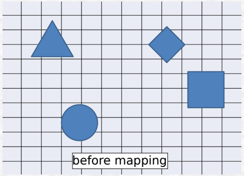
Decomposition of the environment

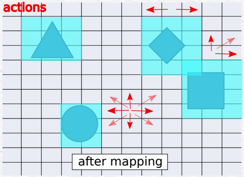
Localization

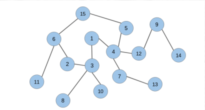
- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid







occupancy grid representation

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

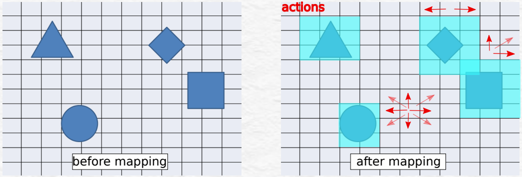
Decomposition of the environment

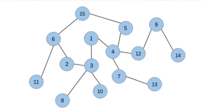
Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
    - Exact occupancy grid





occupancy grid representation

Nodes ∼ free cells;
two nodes are linked ∼ the robot can immediately move from any of them to the other, and “knows” how
Target point ∼ target node
Path planning ∼ finding a path on the graph
Sequence of way-points (“way-cells”)

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

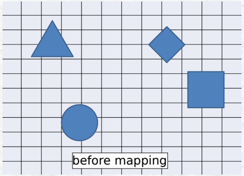
Decomposition of the environment

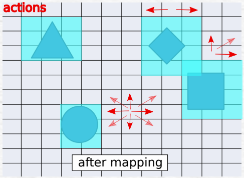
Localization

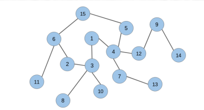
- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition







occupancy grid representation

Nodes ∼ free cells;
two nodes are linked ∼ the robot can immediately move from any of them to the other, and “knows” how
Target point ∼ target node
Path planning ∼ finding a path on the graph
Sequence of way-points (“way-cells”)

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

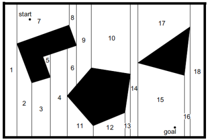
Decomposition of the environment

Localization

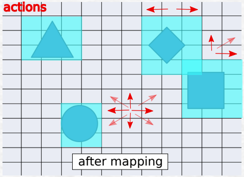
- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

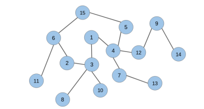
- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition



cell decomposition



after mapping



occupancy grid representation

Nodes $\sim$ free cells;
two nodes are linked $\sim$ the robot can immediately move from any of them to the other, and “knows” how
Target point $\sim$ target node
Path planning $\sim$ finding a path on the graph
Sequence of way-points (“way-cells”)

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

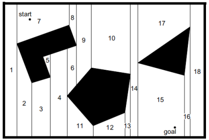
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

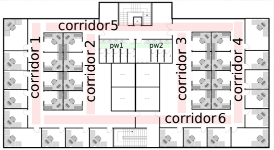
Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition

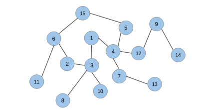


cell decomposition

34 workplaces
8 toilet cabins



cell decomposition



occupancy grid representation

Nodes $\sim$ free cells;
two nodes are linked $\sim$ the robot can immediately move from any of them to the other, and “knows” how
Target point $\sim$ target node
Path planning $\sim$ finding a path on the graph
Sequence of way-points (“way-cells”)

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

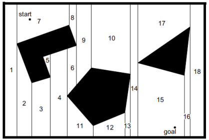
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

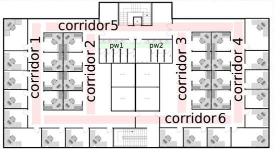
Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition
  - Sampling-based representation

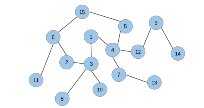


cell decomposition

34 workplaces
8 toilet cabins



cell decomposition



occupancy grid representation

Nodes ∼ free cells;
two nodes are linked ∼ the robot can immediately move from any of them to the other, and "knows" how
Target point ∼ target node
Path planning ∼ finding a path on the graph
Sequence of way-points ("way-cells")

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

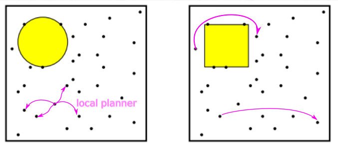
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition
  - Sampling-based representation



Nodes $\sim$ free cells;
two nodes are linked $\sim$ the robot can immediately move from any of them to the other, and "knows" how
Target point $\sim$ target node
Path planning $\sim$ finding a path on the graph
Sequence of way-points ("way-cells")

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

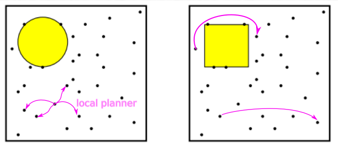
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition
  - Sampling-based representation



Nodes $\sim$ sample points;
two nodes are linked $\sim$ the robot can immediately move from any of them to the other, and "knows" how
Target point $\sim$ target node
Path planning $\sim$ finding a path on the graph
Sequence of way-points ("way-cells")

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

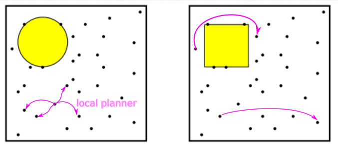
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition
  - Sampling-based representation



Nodes $\sim$ sample points;
an edge goes from point p_1 to p_2 $\sim$ the robot can immediately move from any of them to the other, and “knows” how
Target point $\sim$ target node
Path planning $\sim$ finding a path on the graph
Sequence of way-points (“way-cells”)

Общие определения и примеры

Map

An abstracted representation of the salient features of the environment in which the robot moves. Should identify

- the zones where the robot can move and
- the zones (obstacles) where the robot cannot do so

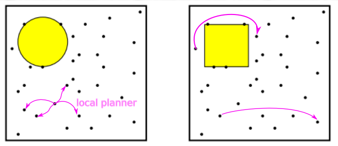
Decomposition of the environment

Localization

- Determine the targeted object in the map
- Find its own position in the map

Some popular types of maps

- Continuous geometric representations
- Topological (logical) representations
  - Exact occupancy grid
  - Exact cell decomposition
  - Sampling-based representation



Nodes $\sim$ sample points;
an edge goes from point p_1 to p_2 $\sim$ the robot can safely move from from p_1 to p_2, and “knows” how
Target point $\sim$ target node
Path planning $\sim$ finding a path on the graph
Sequence of way-points

Разбиение (мозаика) Вороного

Разбиение (мозаика) Вороного

Definition

Let X be a metric space with the distance function $d(\cdot,\cdot)$ and let $\{x_1,\dots,x_n\} \subset X$ be a finite set of points. The Voronoi cell with the center x_i is defined to be $C_i := \{x \in X : d(x,x_i) < d(x,x_j)\ \forall j \neq i\}$.

Разбиение (мозаика) Вороного

Definition

Let X be a metric space with the distance function $d(\cdot,\cdot)$ and let $\{x_1,\dots,x_n\} \subset X$ be a finite set of points. The Voronoi cell with the center x_i is defined to be $C_i := \{x \in X : d(x,x_i) < d(x,x_j)\ \forall j \neq i\}$.

Basic properties

- $C_i \cap C_j = \emptyset\ \forall i \neq j$
- $x \notin \bigcup_{i=1}^{n} C_i \Rightarrow x \in H_\cup := \bigcup_{i \neq j} H_{i,j}$,
  where
  $H_{i,j} := \{x : d(x,x_i) = d(x,x_j)\}$
- $\{C_i\}_{i=1}^{n}$ partition X “up to” $H_\cup$

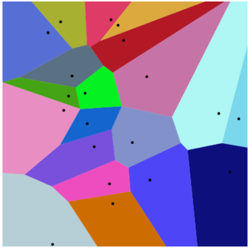
Разбиение (мозаика) Вороного

Definition

Let X be a metric space with the distance function $d(\cdot,\cdot)$ and let $\{x_1,\ldots,x_n\} \subset X$ be a finite set of points. The Voronoi cell with the center x_i is defined to be $C_i := \{x \in X : d(x,x_i) < d(x,x_j)\ \forall j \neq i\}$.

Basic properties

- $C_i \cap C_j = \emptyset\ \forall i \neq j$
- $x \notin \bigcup_{i=1}^{n} C_i \Rightarrow x \in H_\cup := \bigcup_{i \neq j} H_{i,j}$, where $H_{i,j} := \{x : d(x,x_i) = d(x,x_j)\}$
- $\{C_i\}_{i=1}^{n}$ partition X "up to" $H_\cup$



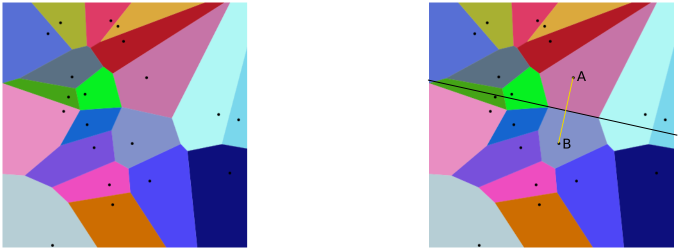
Разбиение (мозаика) Вороного

Definition

Let X be a metric space with the distance function $d(\cdot,\cdot)$ and let $\{x_1,\dots,x_n\} \subset X$ be a finite set of points. The Voronoi cell with the center x_i is defined to be $C_i := \{x \in X : d(x,x_i) < d(x,x_j)\ \forall j \neq i\}$.

Basic properties

- $C_i \cap C_j = \emptyset\ \forall i \neq j$
- $x \notin \bigcup_{i=1}^{n} C_i \Rightarrow x \in H_\cup := \bigcup_{i \neq j} H_{i,j}$, where $H_{i,j} := \{x : d(x,x_i) = d(x,x_j)\}$
- $\{C_i\}_{i=1}^{n}$ partition X "up to" $H_\cup$



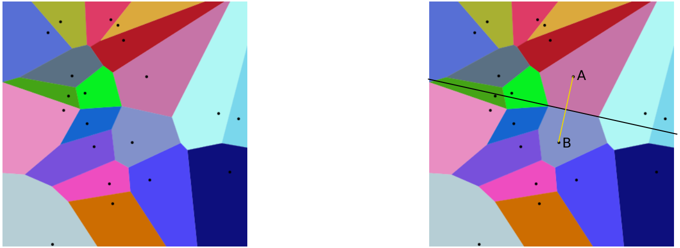
Разбиение (мозаика) Вороного

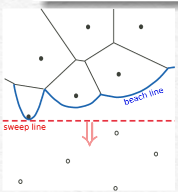
Definition

Let X be a metric space with the distance function $d(\cdot,\cdot)$ and let $\{x_1,\ldots,x_n\} \subset X$ be a finite set of points. The Voronoi cell with the center x_i is defined to be $C_i := \{x \in X : d(x,x_i) < d(x,x_j)\ \forall j \neq i\}$.

Basic properties

- $C_i \cap C_j = \emptyset\ \forall i \neq j$
- $x \notin \bigcup_{i=1}^{n} C_i \Rightarrow x \in H_{\cup} := \bigcup_{i \neq j} H_{i,j}$, where $H_{i,j} := \{x : d(x,x_i) = d(x,x_j)\}$
- $\{C_i\}_{i=1}^{n}$ partition X "up to" $H_{\cup}$





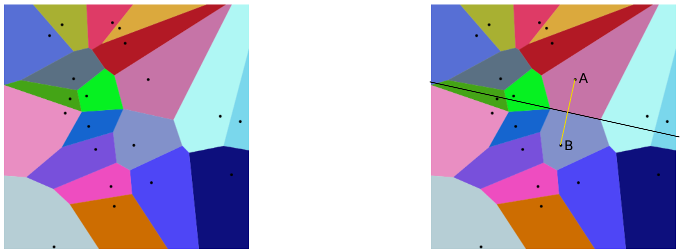
Разбиение (мозаика) Вороного

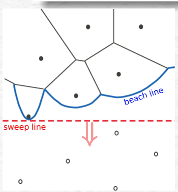
Definition

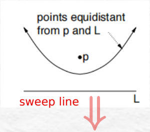
Let X be a metric space with the distance function $d(\cdot,\cdot)$ and let $\{x_1,\dots,x_n\} \subset X$ be a finite set of points. The Voronoi cell with the center x_i is defined to be $C_i := \{x \in X : d(x,x_i) < d(x,x_j)\ \forall j \neq i\}$.

Basic properties

- $C_i \cap C_j = \emptyset\ \forall i \neq j$
- $x \notin \bigcup_{i=1}^{n} C_i \Rightarrow x \in H_\cup := \bigcup_{i \neq j} H_{i,j}$, where $H_{i,j} := \{x : d(x,x_i) = d(x,x_j)\}$
- $\{C_i\}_{i=1}^{n}$ partition X “up to” $H_\cup$







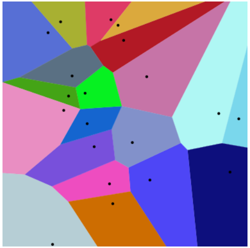
Разбиение (мозаика) Вороного

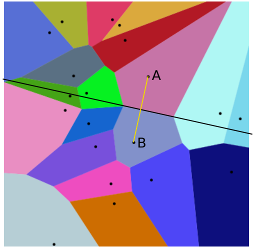
Definition

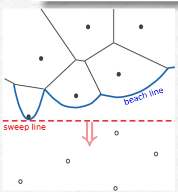
Let X be a metric space with the distance function $d(\cdot,\cdot)$ and let $\{x_1,\dots,x_n\} \subset X$ be a finite set of points. The Voronoi cell with the center x_i is defined to be $C_i := \{x \in X : d(x,x_i) < d(x,x_j)\ \forall j \neq i\}$.

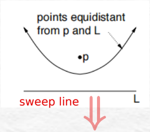
Basic properties

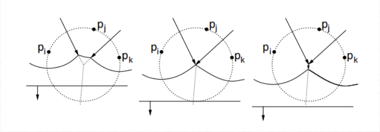
- $C_i \cap C_j = \emptyset\ \forall i \neq j$
- $x \notin \bigcup_{i=1}^{n} C_i \Rightarrow x \in H_\cup := \bigcup_{i \neq j} H_{i,j}$, where $H_{i,j} := \{x : d(x,x_i) = d(x,x_j)\}$
- $\{C_i\}_{i=1}^{n}$ partition X "up to" $H_\cup$



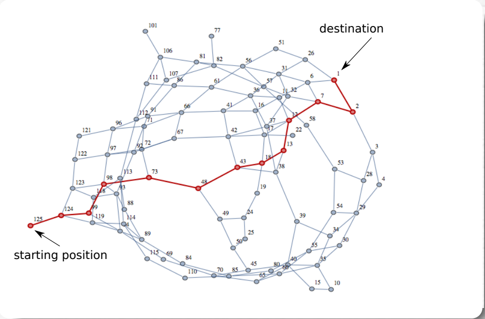






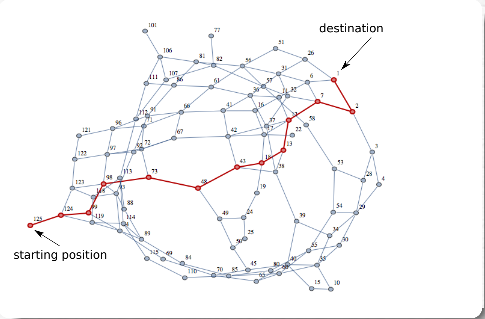


Планирование: подходы, основанные на логическом представлении



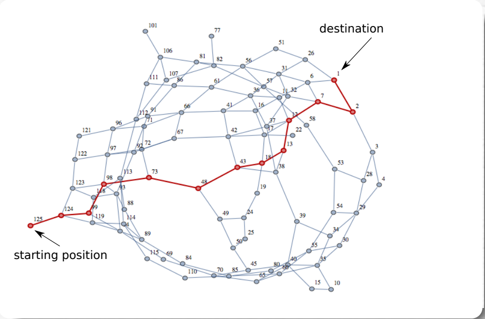
Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology



Walk over nodes: terminology

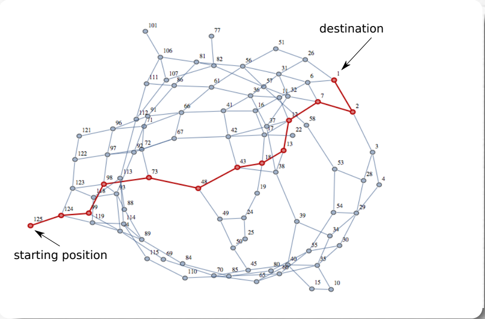
- Visited/unvisited node



Планирование: подходы, основанные на логическом представлении

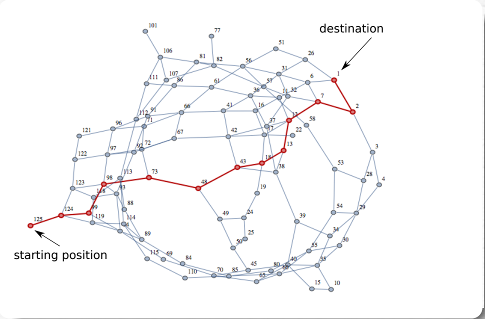
Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node



Walk over nodes: terminology

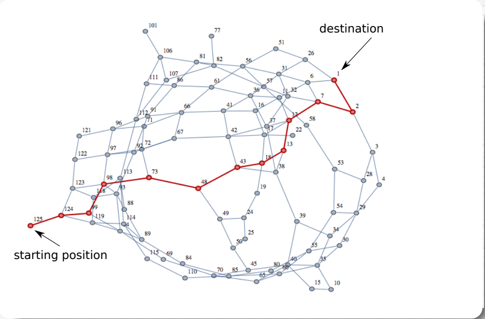
- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded



Планирование: подходы, основанные на логическом представлении

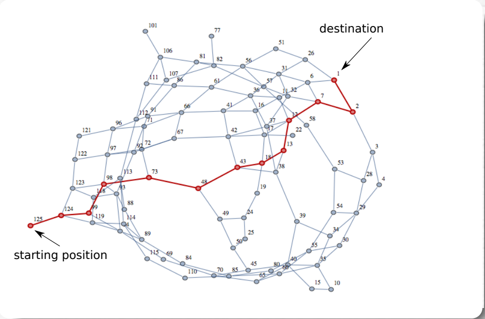
Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q



Walk over nodes: terminology

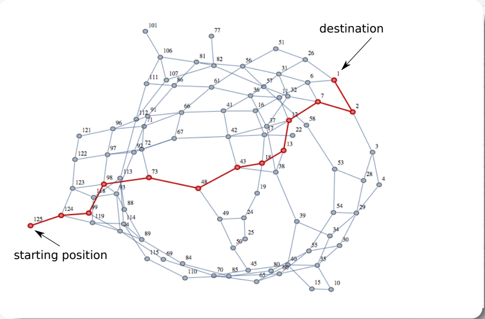
- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination



Планирование: подходы, основанные на логическом представлении

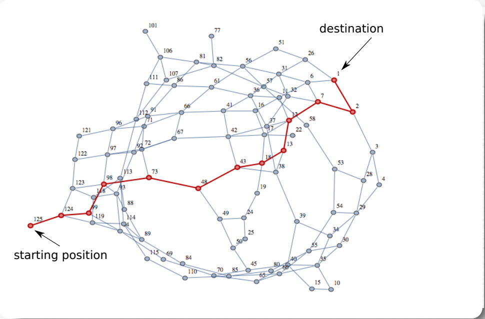
Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent



Walk over nodes: terminology

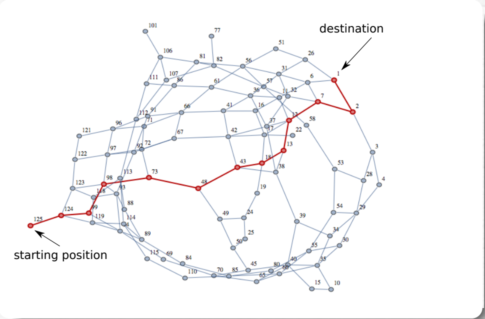
- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method ∼ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference ∼ indication to a parent
- Mark “dead”



Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

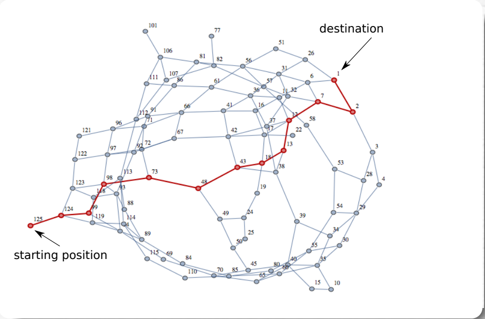
- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)



Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

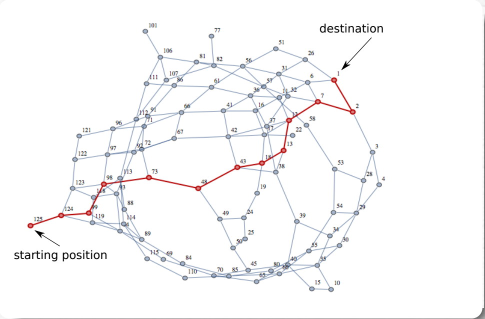
- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)



Breadth first (grass-fire front) method

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method ∼ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference ∼ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)

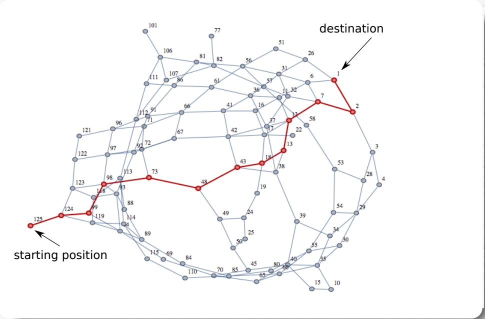


Breadth first (grass-fire front) method

- Start with the source, put in in Q

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method ∼ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference ∼ indication to a parent
- Mark "dead"
- Use the references to find the path (finalizing backward walk)



Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy

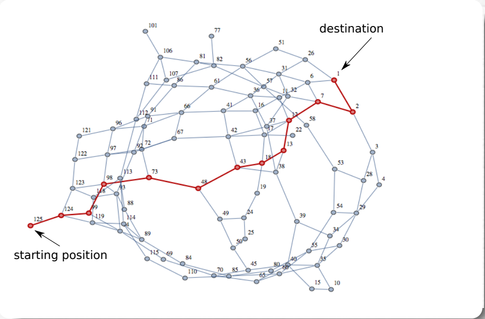
Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method ∼ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference ∼ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)

Breadth first (grass-fire front) method

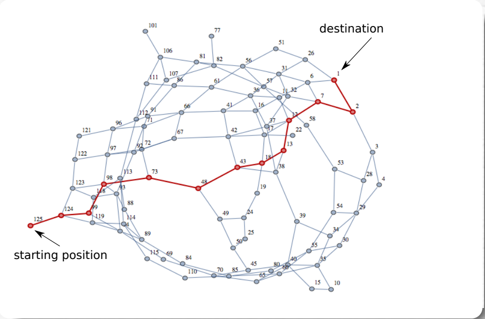
- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors



Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)



Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes

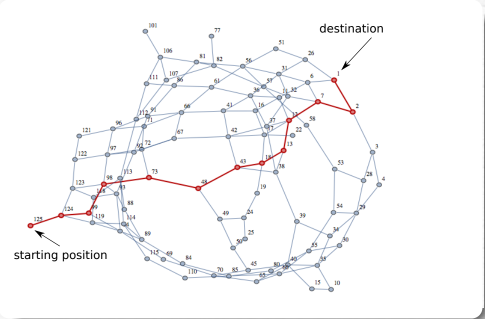
Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method ∼ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference ∼ indication to a parent
- Mark "dead"
- Use the references to find the path (finalizing backward walk)

Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path



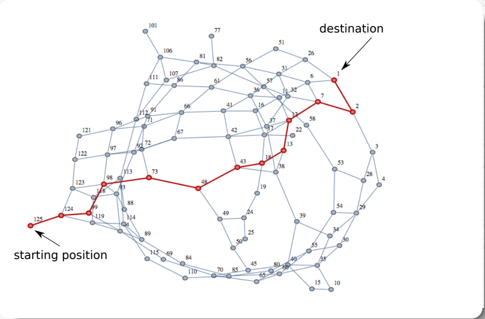
Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method ∼ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference ∼ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)

Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

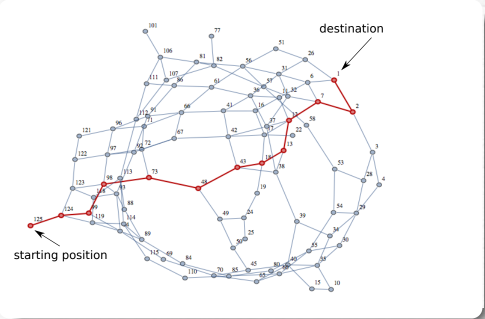


Properties

Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)



Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

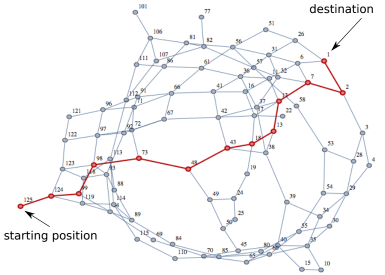
Properties

- Degree of the node := the length of the shortest path from the source

Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)



Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

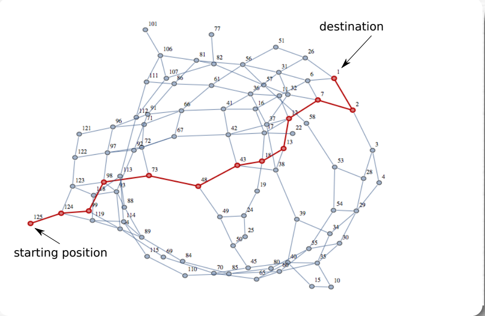
Properties

- Degree of the node := the length of the shortest path from the source
- for any step of the search stage, there exists $k \geq 0$ such that Q contains only nodes of degree k or $k+1$ and all nodes of degree $< k$ are dead

Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)



Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

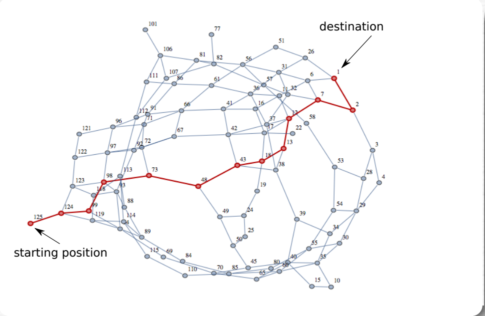
Properties

- Degree of the node := the length of the shortest path from the source
- for any step of the search stage, there exists $k \geq 0$ such that Q contains only nodes of degree k or $k+1$ and all nodes of degree $< k$ are dead
- References highlight a shortest path

Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark "dead"
- Use the references to find the path (finalizing backward walk)



Breadth first (grass-fire front) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

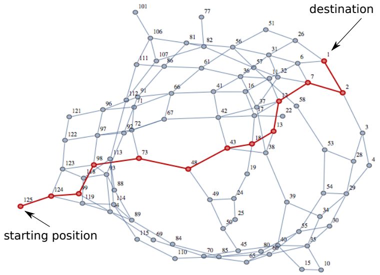
Properties

- Degree of the node := the length of the shortest path from the source
- for any step of the search stage, there exists $k \geq 0$ such that Q contains only nodes of degree k or $k+1$ and all nodes of degree $< k$ are dead
- References highlight a shortest path
- $O(|V| + |E|)$

Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark "dead"
- Use the references to find the path (finalizing backward walk)



Depth first (aggressive deepening) method

- Start with the source, put in in Q
- Sort Q according to FIFO (first-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

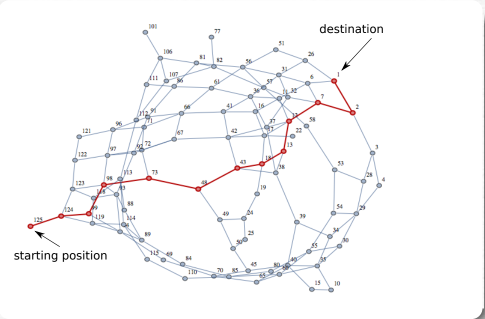
Properties

- Degree of the node := the length of the shortest path from the source
- for any step of the search stage, there exists $k \geq 0$ such that Q contains only nodes of degree k or $k+1$ and all nodes of degree $< k$ are dead
- References highlight a shortest path
- $O(|V| + |E|)$

Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark "dead"
- Use the references to find the path (finalizing backward walk)



Depth first (aggressive deepening) method

- Start with the source, put in in Q
- Sort Q according to LIFO (last-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

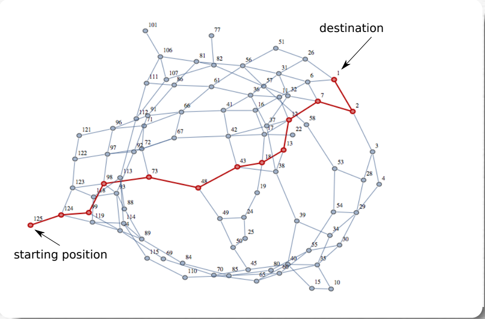
Properties

- Degree of the node := the length of the shortest path from the source
- for any step of the search stage, there exists $k \geq 0$ such that Q contains only nodes of degree k or $k+1$ and all nodes of degree $< k$ are dead
- References highlight a shortest path
- $O(|V| + |E|)$

Планирование: подходы, основанные на логическом представлении

Walk over nodes: terminology

- Visited/unvisited node
- Visited dead/alive node
- Priority queue Q of the alive nodes, the dead ones are excluded
- Prioritization method $\sim$ to sort Q
- Direct walk (search) starts from the source and aims at the destination
- Reference $\sim$ indication to a parent
- Mark “dead”
- Use the references to find the path (finalizing backward walk)



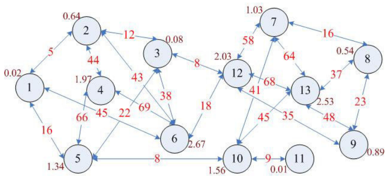
Depth first (aggressive deepening) method

- Start with the source, put in in Q
- Sort Q according to LIFO (last-in-first-out) policy
- Test only unvisited neighbors
- When arriving at the destination, terminate processing nodes
- Use the references to build a path

Properties

- Degree of the node := the length of the shortest path from the source
- Finds the destination
- References highlight a path
- $O(|V| + |E|)$

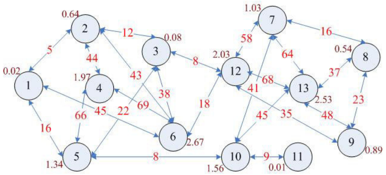
Планирование: подходы, основанные на логическом представлении



$W(n \to n') > 0$ weight of the edge

Планирование: подходы, основанные на логическом представлении

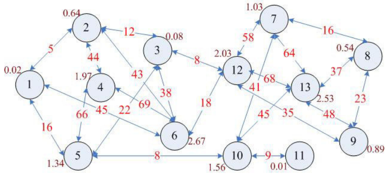
Dijkstra's algorithm



$W(n \to n') > 0$ weight of the edge

Dijkstra's algorithm

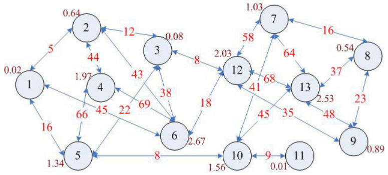
- At any step, forms a partition of the nodes into V (visited) and U (unvisited)



$W(n \to n') > 0$ weight of the edge

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n

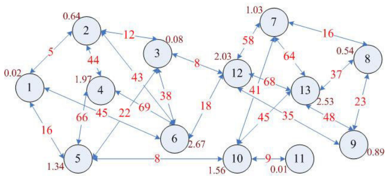


$W(n \to n') > 0$ weight of the edge

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

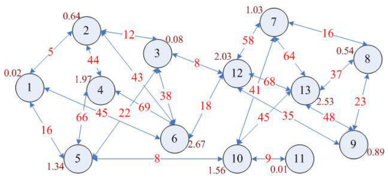
- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset$, U contains all nodes, $L(s) := 0$, $L(n) := \infty \ \forall n \neq s$



$W(n \to n') > 0$ weight of the edge

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
- At any step, do the following

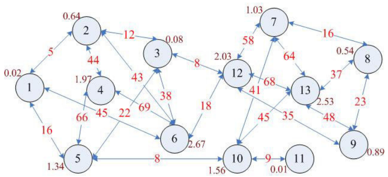


$W(n \to n') > 0$ weight of the edge

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$

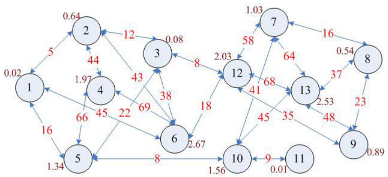


$W(n \to n') > 0$ weight of the edge

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

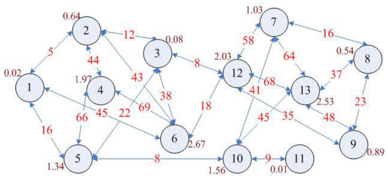
- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)



$W(n \to n') > 0$ weight of the edge

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$

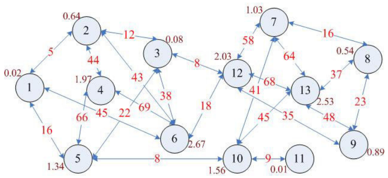


$W(n \to n') > 0$ weight of the edge

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

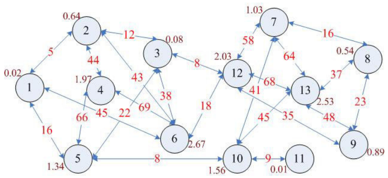
- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V



$W(n \to n') > 0$ weight of the edge

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min\{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$

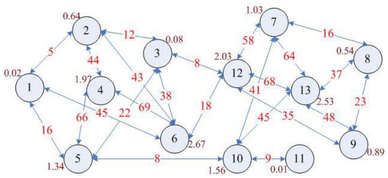


$W(n \to n') > 0$ weight of the edge

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



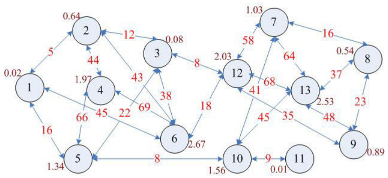
$W(n \to n') > 0$ weight of the edge

Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

Proof:

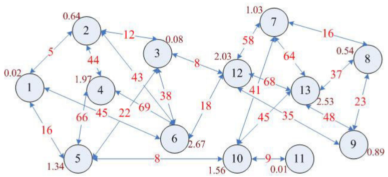
Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
- At any step, do the following
    - find a minimizer of $L(n)$ over $n \in U$
    - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
    - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
    - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \; \forall n \in U$



$W(n \to n') > 0$ weight of the edge

Proof: Induction on the number N of visited nodes.

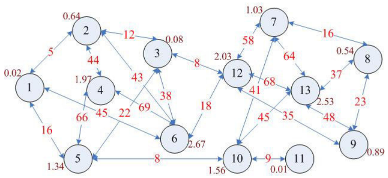
Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min\{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

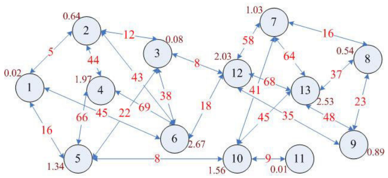
Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial.

Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \; \forall n \in U$



$W(n \to n') > 0$ weight of the edge

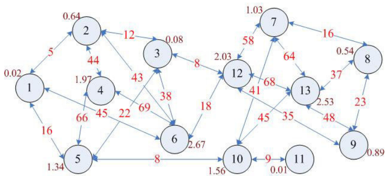
Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited.

Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

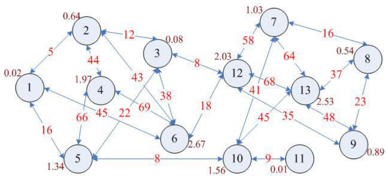
Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited. Then $L(n_*) \geq d(n_*)$.

Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty\ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty\ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

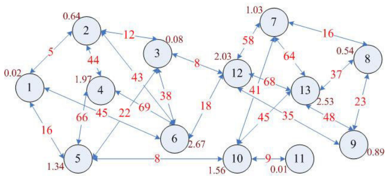
Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited. Then $L(n_*) \geq d(n_*)$. Consider the shortest path from the source s to this node n_*.

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \; \forall n \in U$



$W(n \to n') > 0$ weight of the edge

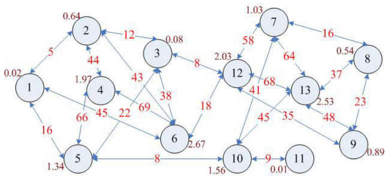
Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited. Then $L(n_*) \geq d(n_*)$. Consider the shortest path from the source s to this node n_*. Let n_u be the first unvisited node on this path.

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

Lemma

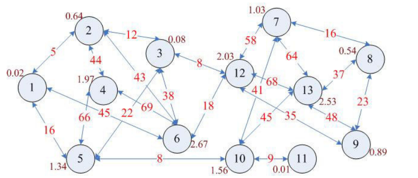
At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited. Then $L(n_*) \geq d(n_*)$. Consider the shortest path from the source s to this node n_*. Let n_u be the first unvisited node on this path. It is preceded by a visited node n_v.

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \; \forall n \in U$



$W(n \to n') > 0$ weight of the edge

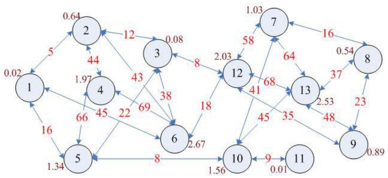
Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited. Then $L(n_*) \geq d(n_*)$. Consider the shortest path from the source s to this node n_*. Let n_u be the first unvisited node on this path. It is preceded by a visited node n_v.
$d(n_u) = d(n_v) + W(n_v \to n_u)$.

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

Lemma

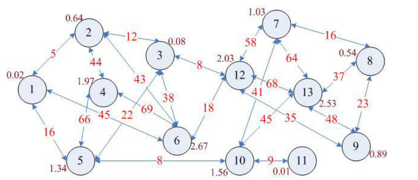
At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited. Then $L(n_*) \geq d(n_*)$. Consider the shortest path from the source s to this node n_*. Let n_u be the first unvisited node on this path. It is preceded by a visited node n_v.
$d(n_u) = d(n_v) + W(n_v \to n_u)$. Meanwhile, $L(n_v) = d(n_v)$ and so
$L(n_u) \leq L(n_v) + W(n_v \to n_u) = d(n_v) + W(n_v \to n_u) = d(n_u)$.

Планирование: подходы, основанные на логическом представлении

Dijkstra's algorithm

- At any step, forms a partition of the nodes into V (visited) and U (unvisited)
- Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
- Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
- At any step, do the following
  - find a minimizer of $L(n)$ over $n \in U$
  - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
  - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
  - when the run is completed, remove n from U to V
- terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \; \forall n \in U$



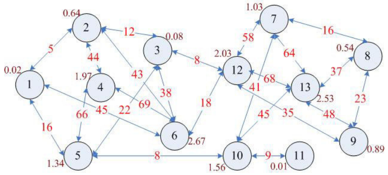
$W(n \to n') > 0$ weight of the edge

Lemma

At any step and for the node visited at this step, its label $L(n)$ is the shortest length $d(n)$ of the path from the source s to n. The final set U is the set of the nodes unreachable from s.

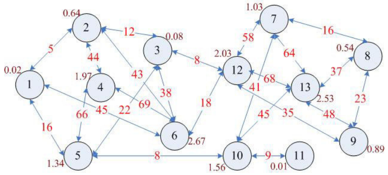
Proof: Induction on the number N of visited nodes. For $N = 0$ is trivial. Let the claim be true for some N and one more node n_* is visited. Then $L(n_*) \geq d(n_*)$. Consider the shortest path from the source s to this node n_*. Let n_u be the first unvisited node on this path. It is preceded by a visited node n_v.
$d(n_u) = d(n_v) + W(n_v \to n_u)$. Meanwhile, $L(n_v) = d(n_v)$ and so
$L(n_u) \leq L(n_v) + W(n_v \to n_u) = d(n_v) + W(n_v \to n_u) = d(n_u)$. On the other hand, $L(n_*) \leq L(n_u) \leq d(n_u) \leq d(n_*)$

Планирование: подходы, основанные на логическом представлении



$W(n \to n') > 0$ weight of the edge

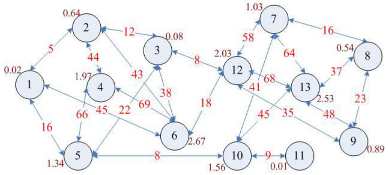
$A^{\star}$-algorithm



$W(n \to n') > 0$ weight of the edge

$A^{\star}$-algorithm

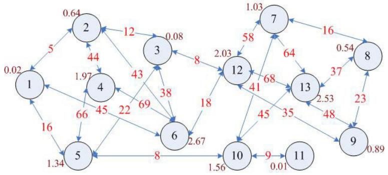
- $G(n)$ – the minimum weight of the path from n to the destination (cost-to-go)



$W(n \to n') > 0$ weight of the edge

$A^{\star}$-algorithm

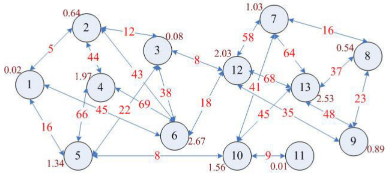
- $G(n)$ – the minimum weight of the path from n to the destination (cost-to-go)
- $G(n) \geq G_{\downarrow}(n)$ - its known lower estimate



$W(n \to n') > 0$ weight of the edge

$A^\star$-algorithm

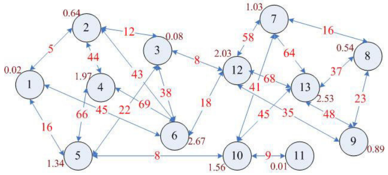
- $G(n)$ – the minimum weight of the path from n to the destination (cost-to-go)
- $G(n) \geq G_\downarrow(n)$ - its known lower estimate
  - At any step, forms a partition of the nodes into V and U
  - Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
  - Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
  - At any step, do the following
    - find a minimizer of $L(n)$ over $n \in U$
    - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
    - for any such n', put $L(n') := \min \{L(n'); L(n) + W(n \to n')\}$
    - when the run is completed, remove n from U to V
  - terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

$A^\star$-algorithm

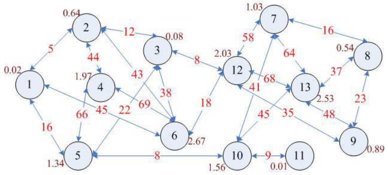
- $G(n)$ – the minimum weight of the path from n to the destination (cost-to-go)
- $G(n) \geq G_\downarrow(n)$ - its known lower estimate
  - At any step, forms a partition of the nodes into V and U
  - Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
  - Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \ \forall n \neq s$
  - At any step, do the following
    - find a minimizer of $L(n)+G_\downarrow(n)$ over $n \in U$
    - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
    - for any such n', put $L(n') := \min\{L(n'); L(n) + W(n \to n')\}$
    - when the run is completed, remove n from U to V
  - terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \ \forall n \in U$



$W(n \to n') > 0$ weight of the edge

$A^\star$-algorithm

- $G(n)$ – the minimum weight of the path from n to the destination (cost-to-go)
- $G(n) \geq G_\downarrow(n)$ - its known lower estimate
  - At any step, forms a partition of the nodes into V and U
  - Iteratively re-calculates labels $L(n) \geq 0$ of nodes n
  - Initially, $V := \emptyset, U$ contains all nodes, $L(s) := 0, L(n) := \infty \; \forall n \neq s$
  - At any step, do the following
    - find a minimizer of $L(n)+G_\downarrow(n)$ over $n \in U$
    - run through the set of all unvisited nodes n' such that $n \to n'$ is an edge (the set of nearest unvisited descendants)
    - for any such n', put $L(n') := \min\{L(n'); L(n) + W(n \to n')\}$
    - when the run is completed, remove n from U to V
  - terminate the algorithm whenever either $U = \emptyset$ or $L(n) = \infty \; \forall n \in U$

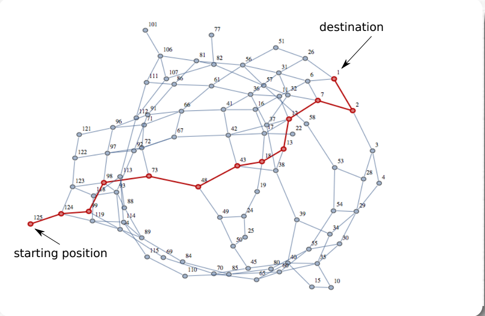


$W(n \to n') > 0$ weight of the edge

Lemma

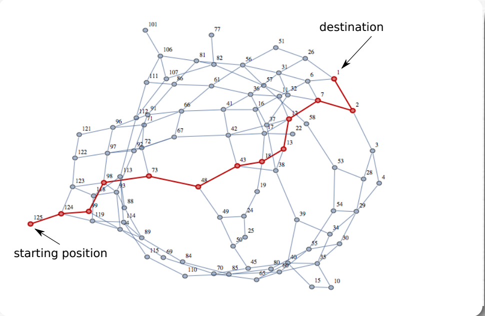
At any step and for any node, its label $L(n)$ is the shortest length $d(n)$ of the path among those that go from the source s to n through V (except for the last node). (The minimum over the empty set is defined to be ∞.) The final set U is the set of the nodes unreachable from s.

Планирование: подходы, основанные на логическом представлении



Best first

- A numerical evaluation $G_*(n)$ of "successfulness" of node n (e.g., cost-to-go)
  - Start with the source, put it in Q
  - Sort Q in the decreasing value of $G_*(n)$
  - Test only unvisited neighbors
  - When arriving at the destination, terminate processing nodes
  - Use the references to build a path



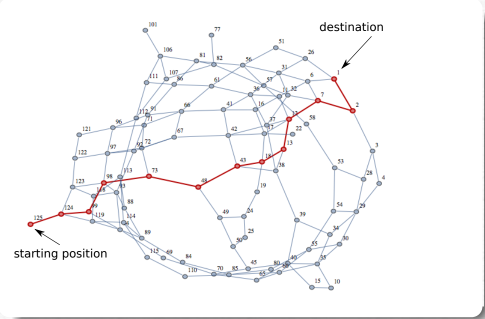
Планирование: подходы, основанные на логическом представлении

Best first

- A numerical evaluation $G_*(n)$ of “successfulness” of node n (e.g., cost-to-go)
  - Start with the source, put it in Q
  - Sort Q in the decreasing value of $G_*(n)$
  - Test only unvisited neighbors
  - When arriving at the destination, terminate processing nodes
  - Use the references to build a path

Iterative deepening

1. Put $d := 1$
2. Perform the depth-first search, with limiting it to nodes of degree $\leq d$
3. If the search is not terminated (the destination is not found)
   - Erase the results of all previous computations
   - Put $d := d + 1$
   - Go to step 2, where the search starts with the source once more



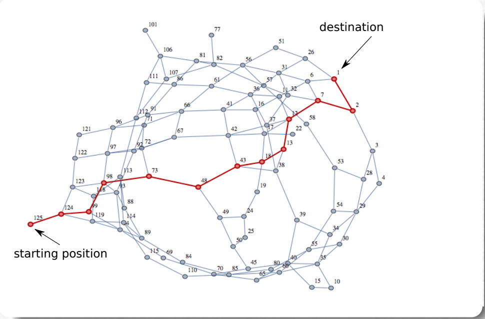
Планирование: подходы, основанные на логическом представлении

Best first

- A numerical evaluation $G_*(n)$ of “successfulness” of node n (e.g., cost-to-go)
  - Start with the source, put it in Q
  - Sort Q in the decreasing value of $G_*(n)$
  - Test only unvisited neighbors
  - When arriving at the destination, terminate processing nodes
  - Use the references to build a path

Iterative deepening

1. Put $d := 1$
2. Perform the depth-first search, with limiting it to nodes of degree $\leq d$
3. If the search is not terminated (the destination is not found)
   - Erase the results of all previous computations
   - Put $d := d + 1$
   - Go to step 2, where the search starts with the source once more



Backward search

Any of the forward search methods, where the roles of the source and destination are interchanged.

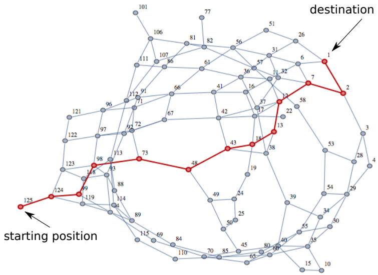
Планирование: подходы, основанные на логическом представлении

Best first

- A numerical evaluation $G_*(n)$ of “successfulness” of node n (e.g., cost-to-go)
    - Start with the source, put it in Q
    - Sort Q in the decreasing value of $G_*(n)$
    - Test only unvisited neighbors
    - When arriving at the destination, terminate processing nodes
    - Use the references to build a path

Iterative deepening

1. Put $d := 1$
2. Perform the depth-first search, with limiting it to nodes of degree $\leq d$
3. If the search is not terminated (the destination is not found)
    - Erase the results of all previous computations
    - Put $d := d + 1$
    - Go to step 2, where the search starts with the source once more



Backward search

Any of the forward search methods, where the roles of the source and destination are interchanged.

Bidirectional search

A forward and backward search methods are run simultaneously until meeting.

Search as evolution of a dynamical system

Search as evolution of a dynamical system

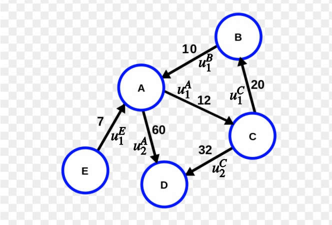
- States x, nodes of the graph

Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes

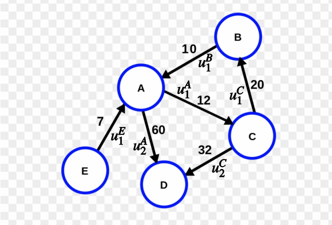
Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x



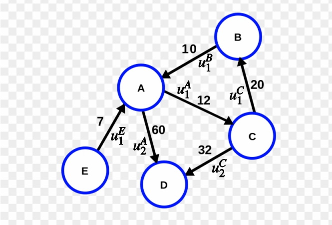
Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x



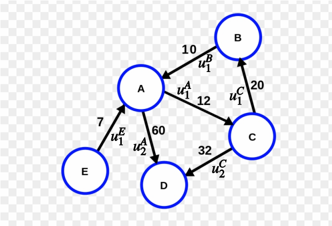
Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u



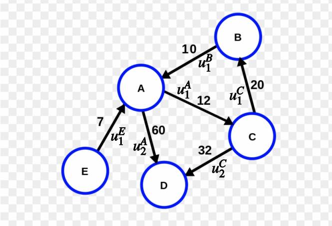
Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u
- $\varphi(x, u) \geq 0$, cost of applying u at the state x



Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u
- $\varphi(x, u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x



Search as evolution of a dynamical system

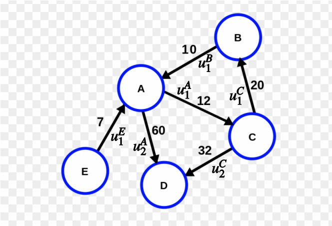
- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x,u) \in X$, head of the edge u
- $\varphi(x,u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x

Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✐)$$



Search as evolution of a dynamical system

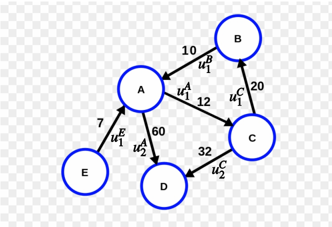
- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u
- $\varphi(x, u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x

Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad x_k = b \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) \to \min \qquad (⚮)$$



Search as evolution of a dynamical system

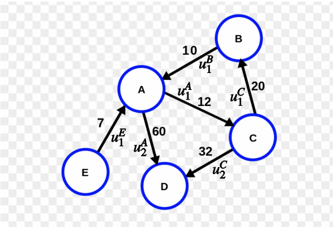
- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x,u) \in X$, head of the edge u
- $\varphi(x,u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x

Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$



Search as evolution of a dynamical system

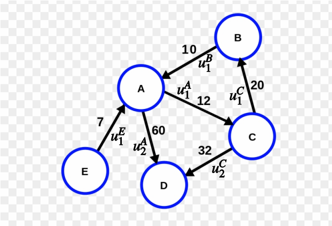
- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x,u) \in X$, head of the edge u
- $\varphi(x,u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x

Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (\maltese)$$

$$x_{t_0} = a, \qquad (\maltese)$$

$$I := \sum_{t \in [t_0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\infty)$$



Setup hints

It is needed to

Search as evolution of a dynamical system

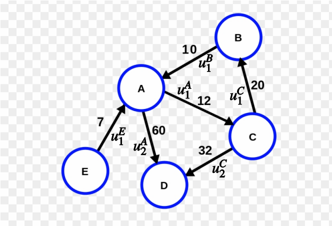
- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u
- $\varphi(x, u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x

Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✾)$$



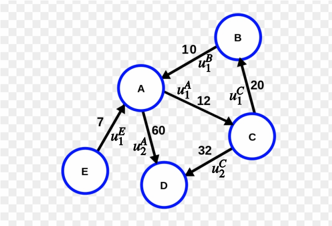
Setup hints

It is needed to

- arrive at b: ☞ $\eta(x) := \lambda\|x - b\|, \lambda \approx \infty$

Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x,u) \in X$, head of the edge u
- $\varphi(x,u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x



Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✇)$$

Setup hints

It is needed to

- arrive at b: ☞ $\eta(x) := \lambda \|x - b\|, \lambda \approx \infty$
- arrive at the set D: ☞ $\eta(x) := \lambda \mathrm{dist}(x, C), \lambda \approx \infty$

Search as evolution of a dynamical system

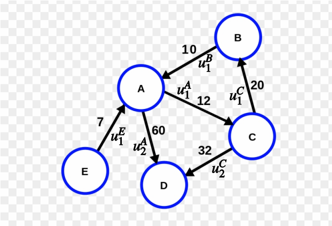
- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u
- $\varphi(x, u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x

Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \tag{✣}$$

$$x_{t_0} = a, \tag{✚}$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \tag{⚭}$$



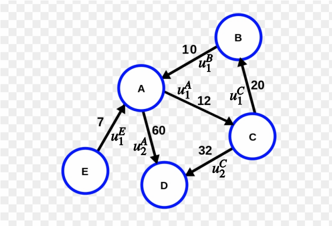
Setup hints

It is needed to

- arrive at b: ☞ $\eta(x) := \lambda\|x - b\|, \lambda \approx \infty$
- arrive at the set D: ☞ $\eta(x) := \lambda \mathrm{dist}(x, C), \lambda \approx \infty$
- trespass a given landmark $x_1 \geq x_1^0$ ☞ $\eta(x) := \lambda(x_1^0 - x_1), \lambda \approx \infty$

Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u
- $\varphi(x, u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x



Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

$$I := \sum_{t \in [t_0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (⚮)$$

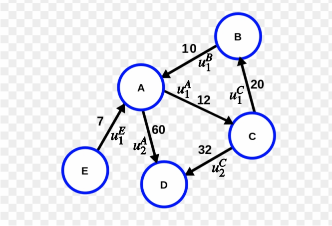
Setup hints

It is needed to

- arrive at b: ☞ $\eta(x) := \lambda \|x - b\|, \lambda \approx \infty$
- arrive at the set D: ☞ $\eta(x) := \lambda \mathrm{dist}(x, C), \lambda \approx \infty$
- trespass a given landmark $x_1 \geq x_1^0$ ☞ $\eta(x) := \lambda(x_1^0 - x_1), \lambda \approx \infty$
- arrive at some of $b_1, \ldots, b_s$: ☞ $\eta(x) := \lambda \min_{i=1,\ldots,s} \|x - b_i\|, \lambda \approx \infty$

Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x,u) \in X$, head of the edge u
- $\varphi(x,u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x



Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

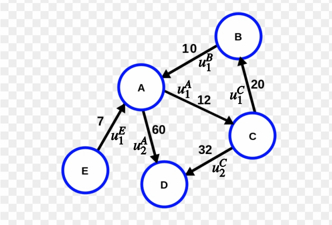
$$I := \sum_{t \in [t_0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Rigorous problem statement

$$\mathfrak{M}(t_0, a) := \left\{ p = \left[\{x_t\}_{t=t_0}^{k}, \{u_t\}_{t=t_0}^{k-1} \right] : (✣) \text{ and } (✚) \text{ hold} \right\}$$

Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x, u) \in X$, head of the edge u
- $\varphi(x, u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x



Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

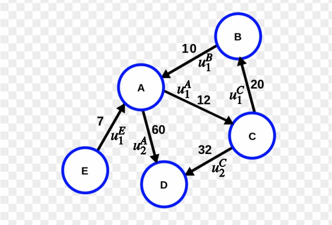
Rigorous problem statement

$$\mathfrak{M}(t_0, a) := \left\{ p = \left[\{x_t\}_{t=t_0}^{k}, \{u_t\}_{t=t_0}^{k-1} \right] : (✣) \text{ and } (✚) \text{ hold} \right\}$$

$$\min_{p \in \mathfrak{M}(t_0, a)} I =: V_{t_0}(a)$$

Search as evolution of a dynamical system

- States x, nodes of the graph
- Phase space X, finite set of the nodes
- Controls u encode the edges outgoing from node x
- $U(x) \neq \emptyset$, finite set of the edges outgoing from x
- $f(x,u) \in X$, head of the edge u
- $\varphi(x,u) \geq 0$, cost of applying u at the state x
- $\eta(x) \geq 0$, penalty for terminating at state x



Optimal search with given number k of steps

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (⚯)$$

Rigorous problem statement

$$\mathfrak{M}(t_0, a) := \left\{ p = \left[\{x_t\}_{t=t_0}^{k}, \{u_t\}_{t=t_0}^{k-1} \right] : (✣) \text{ and } (✚) \text{ hold} \right\}$$

$$\min_{p \in \mathfrak{M}(t_0, a)} I =: V_{t_0}(a) \qquad V_k(\cdot) := \eta(\cdot)$$

Belman equation

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Belman equation

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k)$$ (✣)

$$x_{t_0} = a,$$ (✚)

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min$$ (⚯)

Belman equation

$$V_t(a) = \min_{u \in U(a)} \left\{ V_{t+1}[f(a,u)] + \varphi(a,u) \right\} \ \forall a \quad t = t_0, \dots, k-1$$

Belman equation

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \qquad (✣)$$

$$x_{t_0} = a, \qquad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✇)$$

Belman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a \quad t = t_0, \dots, k-1$

Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Belman equation

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \quad (✣)$$

$$x_{t_0} = a, \quad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \quad (✑)$$

Belman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a \quad t = t_0, \ldots, k-1$

Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Permits us to compute

$$V_k(\cdot) \curvearrowright V_{k-1}(\cdot) \curvearrowright \ldots \curvearrowright V_0(\cdot), \ V_t(a) \in [0, \infty);$$

$$\mathcal{U}_t(a) := \mathrm{Arg} \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \quad \forall a, t = t_0, \ldots, k-1$$

Belman equation

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [t_0 : k) \quad (✣)$$

$$x_{t_0} = a, \quad (✚)$$

$$I := \sum_{t \in [t_0 : k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \quad (✐)$$

Belman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a \quad t = t_0, \ldots, k-1$

Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Permits us to compute

$$V_k(\cdot) \curvearrowright V_{k-1}(\cdot) \curvearrowright \ldots \curvearrowright V_0(\cdot), \ V_t(a) \in [0, \infty);$$

$$\mathcal{U}_t(a) := \operatorname{Arg} \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \quad \forall a, t = t_0, \ldots, k-1$$

Theorem

The following statements hold:

- The Bellman function is the unique solution of the Bellman equation
- $u_t \in \mathcal{U}_t(x_t)$ is the optimal control law (search rule)

Proof of the theorem

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

$$(t_0, a) \xrightarrow{u_{t_0} := u \in U(a)} \Big[\underbrace{t_0 + 1}_{\tau}, x_{t_0+1} = \underbrace{f(a, u)}_{x_+}\Big] \xrightarrow{\text{continue optimally from the state}(\tau, x_+)}$$

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

$$(t_0, a) \xrightarrow{u_{t_0} := u \in U(a)} \left[\underbrace{t_0 + 1}_{\tau}, x_{t_0+1} = \underbrace{f(a, u)}_{x_+}\right] \xrightarrow{\text{continue optimally from the state}(\tau, x_+)}$$

$$V_{t_0}(a) \leq I_{t_0}(\text{the just constructed process}) = \underbrace{\varphi(a, u)}_{\text{the first addend}} + I_{t_0+1}(\text{the continuation}) = \varphi(a, u) + V_{t_0+1}[f(a, u)]$$

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

$$(t_0, a) \xrightarrow{u_{t_0} := u \in U(a)} \left[\underbrace{t_0 + 1}_{\tau}, x_{t_0+1} = \underbrace{f(a, u)}_{x_+}\right] \xrightarrow{\text{continue optimally from the state}(\tau, x_+)}$$

$$V_{t_0}(a) \leq I_{t_0}(\text{the just constructed process}) = \underbrace{\varphi(a, u)}_{\text{the first addend}} + I_{t_0+1}(\text{the continuation}) = \varphi(a, u) + V_{t_0+1}[f(a, u)]$$

$$V_{t_0}(a) \leq \varphi(a, u) + V_{t_0+1}[f(a, u)] \qquad \forall u \in U(a)$$

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

$$(t_0, a) \xrightarrow{u_{t_0} := u \in U(a)} \Big[\underbrace{t_0 + 1}_{\tau}, x_{t_0+1} = \underbrace{f(a, u)}_{x_+}\Big] \xrightarrow{\text{continue optimally from the state}(\tau, x_+)}$$

$$V_{t_0}(a) \leq I_{t_0}(\text{the just constructed process}) = \underbrace{\varphi(a, u)}_{\text{the first addend}} + I_{t_0+1}(\text{the continuation}) = \varphi(a, u) + V_{t_0+1}[f(a, u)]$$

$$V_{t_0}(a) \leq \varphi(a, u) + V_{t_0+1}[f(a, u)] \qquad \forall u \in U(a)$$

Process optimal for$(t_0, a) \xrightarrow{\text{disintegrate}}$ the first term$(t_0, a, u^0_{t_0})$ and the remainder that corresponds to $t \geq t_0 + 1$

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

$$(t_0, a) \xrightarrow{u_{t_0} := u \in U(a)} \Big[\underbrace{t_0 + 1}_{\tau}, x_{t_0+1} = \underbrace{f(a, u)}_{x_+}\Big] \xrightarrow{\text{continue optimally from the state}(\tau, x_+)}$$

$$V_{t_0}(a) \le I_{t_0}(\text{the just constructed process}) = \underbrace{\varphi(a, u)}_{\text{the first addend}} + I_{t_0+1}(\text{the continuation}) = \varphi(a, u) + V_{t_0+1}[f(a, u)]$$

$$V_{t_0}(a) \le \varphi(a, u) + V_{t_0+1}[f(a, u)] \qquad \forall u \in U(a)$$

Process optimal for $(t_0, a) \xrightarrow{\text{disintegrate}}$ the first term $(t_0, a, u^0_{t_0})$ and the remainder that corresponds to $t \ge t_0 + 1$

the remainder starts at $t = t_0 + 1$ from the state $x_+ := f(a, u^0_{t_0})$ and should be optimal for these initial conditions

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

$$(t_0, a) \xrightarrow{u_{t_0} := u \in U(a)} \Big[\underbrace{t_0 + 1}_{\tau}, x_{t_0+1} = \underbrace{f(a, u)}_{x_+}\Big] \xrightarrow{\text{continue optimally from the state}(\tau, x_+)}$$

$$V_{t_0}(a) \leq I_{t_0}(\text{the just constructed process}) = \underbrace{\varphi(a, u)}_{\text{the first addend}} + I_{t_0+1}(\text{the continuation}) = \varphi(a, u) + V_{t_0+1}[f(a, u)]$$

$$V_{t_0}(a) \leq \varphi(a, u) + V_{t_0+1}[f(a, u)] \qquad \forall u \in U(a)$$

Process optimal for$(t_0, a) \xrightarrow{\text{disintegrate}}$ the first term$(t_0, a, u^0_{t_0})$ and the remainder that corresponds to $t \geq t_0 + 1$

the remainder starts at $t = t_0 + 1$ from the state $x_+ := f(a, u^0_{t_0})$ and should be optimal for these initial conditions

$$V_{t_0}(a) = I_{t_0}(\text{the optimal process}) = \varphi_{t_0}(a, u^0_{t_0}) + \underbrace{I_{t_0+1}(\text{the remainder})}_{\text{cost at the optimal process}} = \varphi_{t_0}(a, u^0_{t_0}) + V_{t_0+1}[f(a, u^0_{t_0})]$$

Proof of the theorem

Given t_0 and $a \in \mathbb{R}^n$, we build a process from $\mathfrak{M}(t_0, a)$ as follows:

$$(t_0, a) \xrightarrow{u_{t_0} := u \in U(a)} \left[\underbrace{t_0 + 1}_{\tau}, x_{t_0+1} = \underbrace{f(a,u)}_{x_+}\right] \xrightarrow{\text{continue optimally from the state}(\tau, x_+)}$$

$$V_{t_0}(a) \leq I_{t_0}(\text{the just constructed process}) = \underbrace{\varphi(a,u)}_{\text{the first addend}} + I_{t_0+1}(\text{the continuation}) = \varphi(a,u) + V_{t_0+1}[f(a,u)]$$

$$V_{t_0}(a) \leq \varphi(a,u) + V_{t_0+1}[f(a,u)] \qquad \forall u \in U(a)$$

Process optimal for$(t_0, a) \xrightarrow{\text{disintegrate}}$ the first term$(t_0, a, u^0_{t_0})$ and the remainder that corresponds to $t \geq t_0 + 1$

the remainder starts at $t = t_0 + 1$ from the state $x_+ := f(a, u^0_{t_0})$ and should be optimal for these initial conditions

$$V_{t_0}(a) = I_{t_0}(\text{the optimal process}) = \varphi_{t_0}(a, u^0_{t_0}) + \underbrace{I_{t_0+1}(\text{the remainder})}_{\text{cost at the optimal process}} = \varphi_{t_0}(a, u^0_{t_0}) + V_{t_0+1}[f(a, u^0_{t_0})]$$

Overall: $V_{t_0}(a) = \min\limits_{u \in U(a)} \Big\{\varphi(a,u) + V_{t_0+1}[f(a,u)]\Big\}, \qquad u^0_{t_0} = \operatorname{argmin}_{u \in U(a)}\Big\{\ldots\Big\} \Rightarrow u^0_{t_0} \in \mathfrak{U}_{t_0}[x^0_{t_0}] \Rightarrow u^0_t \in \mathfrak{U}_{t_0}[x^0_t]\ \forall t \geq t_0.$

Proof of the theorem

Proof of the theorem

Let the process $p = \left[\{x_t\}_{t=t_0}^{t_1}, \{u_t\}_{t=t_0}^{t_1-1}\right] \in \mathfrak{M}(t_0, a)$ be generated by the dynamic programming regulator $u_t \in \mathcal{U}_t(x_t) = \text{Arg}\,\min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\}$

Proof of the theorem

Let the process $p = \left[\{x_t\}_{t=t_0}^{t_1}, \{u_t\}_{t=t_0}^{t_1-1}\right] \in \mathfrak{M}(t_0, a)$ be generated by the dynamic programming regulator $u_t \in \mathcal{U}_t(x_t) = \text{Arg}\ \min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\}$

$$V_{t+1}[\underbrace{f(x_t, u_t)}_{x_{t+1}}] + \varphi(x_t, u_t) = \min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\} = V_t(x_t)$$

Proof of the theorem

Let the process $p = [\{x_t\}_{t=t_0}^{t_1}, \{u_t\}_{t=t_0}^{t_1-1}] \in \mathfrak{M}(t_0, a)$ be generated by the dynamic programming regulator $u_t \in \mathfrak{U}_t(x_t) = \text{Arg}\ \min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\}$

$$V_{t+1}[\underbrace{f(x_t, u_t)}_{x_{t+1}}] + \varphi(x_t, u_t) = \min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\} = V_t(x_t)$$

$$I_{t_0}(p) = \sum_{t \in [t_0:k)} \varphi[x_t, u_t] = \sum_{t=t_0}^{k-1} \varphi[x_t, u_t] + V_k[x_k] = \sum_{t=t_0}^{k-2} \varphi[x_t, u_t] + \left\{\varphi_{k-1}[x_{k-1}, u_{k-1}] + V_k[x_k]\right\}$$

Proof of the theorem

Let the process $p = \left[\{x_t\}_{t=t_0}^{t_1}, \{u_t\}_{t=t_0}^{t_1-1}\right] \in \mathfrak{M}(t_0, a)$ be generated by the dynamic programming regulator $u_t \in \mathfrak{U}_t(x_t) = \operatorname{Arg\,min}_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\}$

$$V_{t+1}[\underbrace{f(x_t, u_t)}_{x_{t+1}}] + \varphi(x_t, u_t) = \min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\} = V_t(x_t)$$

$$I_{t_0}(p) = \sum_{t \in [t_0:k)} \varphi[x_t, u_t] = \sum_{t=t_0}^{k-1} \varphi[x_t, u_t] + V_k[x_k] = \sum_{t=t_0}^{k-2} \varphi[x_t, u_t] + \left\{\varphi_{k-1}[x_{k-1}, u_{k-1}] + V_k[x_k]\right\}$$

$$\overset{\tau := k-1}{=\!=\!=} \sum_{t=t_0}^{k-2} \varphi[x_t, u_t] + \left\{\varphi[\underbrace{x_\tau}_{=:a_*}, \underbrace{u_\tau}_{=:v \in U(a_*)}] + V_{\tau+1}[f(a_*, v)]\right\}$$

Proof of the theorem

Let the process $p = [\{x_t\}_{t=t_0}^{t_1}, \{u_t\}_{t=t_0}^{t_1-1}] \in \mathfrak{M}(t_0, a)$ be generated by the dynamic programming regulator $u_t \in \mathfrak{U}_t(x_t) = \text{Arg}\ \min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\}$

$$V_{t+1}[\underbrace{f(x_t, u_t)}_{x_{t+1}}] + \varphi(x_t, u_t) = \min_{u \in U(x_t)} \{V_{t+1}[f(x_t, u)] + \varphi(x_t, u)\} = V_t(x_t)$$

$$I_{t_0}(p) = \sum_{t \in [t_0 : k)} \varphi[x_t, u_t] = \sum_{t=t_0}^{k-1} \varphi[x_t, u_t] + V_k[x_k] = \sum_{t=t_0}^{k-2} \varphi[x_t, u_t] + \left\{\varphi_{k-1}[x_{k-1}, u_{k-1}] + V_k[x_k]\right\}$$

$$\overset{\tau := k-1}{=\!=} \sum_{t=t_0}^{k-2} \varphi[x_t, u_t] + \left\{\varphi[\underbrace{x_\tau}_{=: a_*}, \underbrace{u_\tau}_{=: v \in U(a_*)}] + V_{\tau+1}[f(a_*, v)]\right\}$$

$$\overset{\text{Belmann equation}}{=} \sum_{t=t_0}^{k-2} \varphi[x_t, u_t] + V_\tau(a_*) = \sum_{t=t_0}^{k-2} \varphi[x_t, u_t] + V_{k-1}(x_{k-1}) = \cdots = V_{t_0}[x_{t_0}] = V_{t_0}(a)$$

Proof of the theorem

Let the process $p = [\{x_t\}_{t=t_0}^{t_1}, \{u_t\}_{t=t_0}^{t_1-1}] \in \mathfrak{M}(t_0, a)$ be generated by the dynamic programming regulator $u_t \in \mathfrak{U}_t(x_t) = \text{Arg}\,\min_{u \in U(x_t)} \{V_{t+1}[f(x_t,u)] + \varphi(x_t,u)\}$

$$V_{t+1}[\underbrace{f(x_t,u_t)}_{x_{t+1}}] + \varphi(x_t,u_t) = \min_{u \in U(x_t)} \{V_{t+1}[f(x_t,u)] + \varphi(x_t,u)\} = V_t(x_t)$$

$$I_{t_0}(p) = \sum_{t \in [t_0:k)} \varphi[x_t,u_t] = \sum_{t=t_0}^{k-1} \varphi[x_t,u_t] + V_k[x_k] = \sum_{t=t_0}^{k-2} \varphi[x_t,u_t] + \left\{\varphi_{k-1}[x_{k-1},u_{k-1}] + V_k[x_k]\right\}$$

$$\overset{\tau:=k-1}{=\!=\!=} \sum_{t=t_0}^{k-2} \varphi[x_t,u_t] + \left\{\varphi[\underbrace{x_\tau}_{=:a_*}, \underbrace{u_\tau}_{=:v \in U(a_*)}] + V_{\tau+1}[f(a_*,v)]\right\}$$

$$\overset{\text{Belmann equation}}{=} \sum_{t=t_0}^{k-2} \varphi[x_t,u_t] + V_\tau(a_*) = \sum_{t=t_0}^{k-2} \varphi[x_t,u_t] + V_{k-1}(x_{k-1}) = \cdots = V_{t_0}[x_{t_0}] = V_{t_0}(a)$$

$$V_{t_0}(a) = I_{t_0}(p)$$

Оптимальное планирование с нефиксированной длиной пути

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

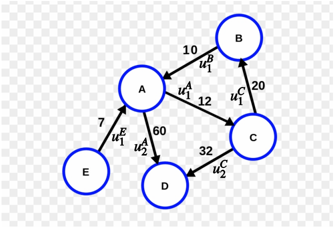
Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (\text{✣})$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (\text{✚})$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\text{✾})$$



Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

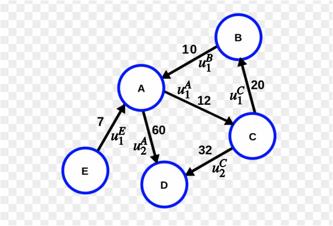
$$I := \sum_{t\in[0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✇)$$

Bellman equation

$V_t(a) = \min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}\ \forall a$

Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathcal{U}_t(a) :=$ Arg $\min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$



Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Bellman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

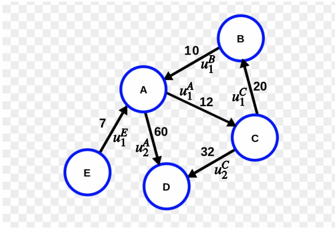
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathfrak{U}_t(a) := \operatorname{Arg}\min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u \in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)]$$



Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Bellman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

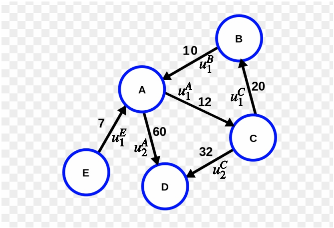
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathfrak{U}_t(a) :=$ Arg $\min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u \in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = k-1, k-2, \ldots, 0$$



Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Bellman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

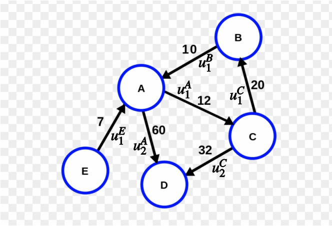
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathfrak{U}_t(a) := \text{Arg} \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u \in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \dots, -k$$



Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t\in[0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✐)$$

Bellman equation

$V_t(a) = \min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

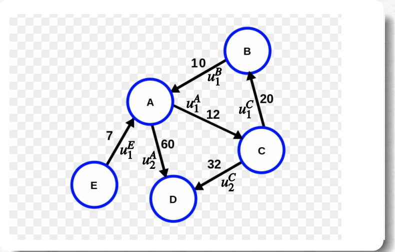
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathfrak{U}_t(a) := \operatorname{Arg}\min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u\in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$

$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \dots, -\infty$



$V_k^{\downarrow}(a) := \min_{\theta\in[-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Bellman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

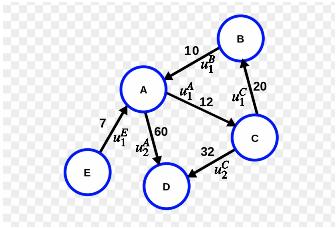
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathcal{U}_t(a) :=$ Arg $\min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min\limits_{u \in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$

$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \dots, -\infty$



$V_k^{\downarrow}(a) := \min_{\theta \in [-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$

$$V_{-k}^{\downarrow}(a) = \min_{\theta \in [-k:0]} V_\theta(a)$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (\ast)$$

$x_0 = a,$ $\quad$ k is freely manipulable $\qquad (+)$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\infty)$$

Bellman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

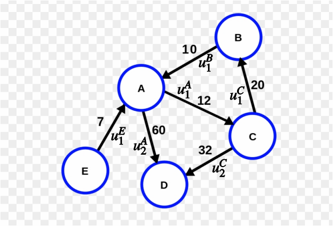
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathcal{U}_t(a) := \text{Arg}\min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u \in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \ldots, -\infty$$



$V_k^{\downarrow}(a) := \min_{\theta \in [-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$

$$V_{-k}^{\downarrow}(a) = \min_{\theta \in [-k:0]} V_\theta(a) = \min \left\{ \min_{\theta \in [-k:-1]} V_\theta(a); \eta(a) \right\}$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✇)$$

Bellman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

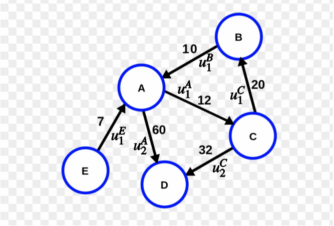
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathfrak{U}_t(a) := \operatorname{Arg} \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u \in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \dots, -\infty$$



$$V_k^{\downarrow}(a) := \min_{\theta \in [-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$$

$$V^{\downarrow}_{-k}(a) = \min_{\theta \in [-k:0]} V_\theta(a) = \min \left\{ \min_{\theta \in [-k:-1]} V_\theta(a); \eta(a) \right\}$$

$$= \min \left\{ \min_{\theta \in [-k:-1]} \min_{u \in U(a)} \Big(V_{\theta+1}[f(a,u)] + \varphi(a,u) \Big); \eta(a) \right\}$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad \text{k is freely manipulable} \qquad (✚)$$

$$I := \sum_{t\in[0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (⚮)$$

Bellman equation

$V_t(a) = \min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \;\; \forall a$

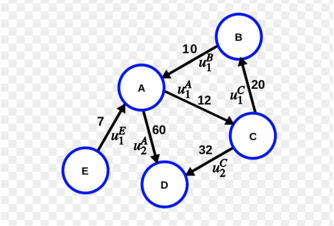
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathfrak{U}_t(a) := \operatorname{Arg}\,\min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \;\; V_-(a) := \min_{u\in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \ldots, -\infty$$



$$V_k^{\downarrow}(a) := \min_{\theta\in[-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$$

$$V^{\downarrow}_{-k}(a) = \min_{\theta\in[-k:0]} V_\theta(a) = \min\left\{\min_{\theta\in[-k:-1]} V_\theta(a); \eta(a)\right\}$$

$$= \min\left\{\min_{u\in U(a)} \min_{\theta\in[-k:-1]} \Big(V_{\theta+1}[f(a,u)] + \varphi(a,u)\Big); \eta(a)\right\}$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t\in[0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✾)$$

Bellman equation

$V_t(a) = \min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

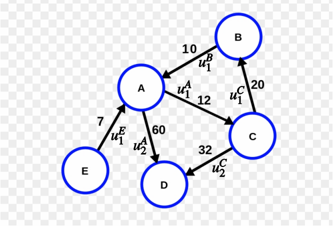
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathcal{U}_t(a) := \operatorname{Arg}\min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u\in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \dots, -\infty$$



$$V_k^{\downarrow}(a) := \min_{\theta\in[-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$$

$$V^{\downarrow}_{-k}(a) = \min_{\theta\in[-k:0]} V_\theta(a) = \min\left\{\min_{\theta\in[-k:-1]} V_\theta(a); \eta(a)\right\}$$

$$= \min\left\{\min_{u\in U(a)} \min_{\theta\in[-k:-1]} \Big(V_{\theta+1}[f(a,u)] + \varphi(a,u)\Big); \eta(a)\right\}$$

$$= \min\left\{\min_{u\in U(a)} \Big[\varphi(a,u) + \min V_{\theta+1}[f(a,u)]\Big]; \eta(a)\right\}$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad \color{blue}{k \text{ is freely manipulable}} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Bellman equation

$V_t(a) = \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \;\; \forall a$

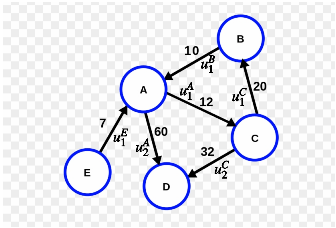
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathcal{U}_t(a) := \text{Arg}\ \min_{u \in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \;\; V_-(a) := \min_{u \in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \ldots, -\infty$$



$$V_k^{\downarrow}(a) := \min_{\theta \in [-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$$

$$V^{\downarrow}_{-k}(a) = \min_{\theta \in [-k:0]} V_\theta(a) = \min\left\{\min_{\theta \in [-k:-1]} V_\theta(a); \eta(a)\right\}$$

$$= \min\left\{\min_{u \in U(a)} \min_{\theta \in [-k:-1]} \Big(V_{\theta+1}[f(a,u)] + \varphi(a,u)\Big); \eta(a)\right\}$$

$$= \min\left\{\min_{u \in U(a)} \left[\varphi(a,u) + \min_{\theta \in [-k:-1]} V[f(a,u)]\right]; \eta(a)\right\}$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad \color{blue}{k \text{ is freely manipulable}} \qquad (✚)$$

$$I := \sum_{t\in[0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (✑)$$

Bellman equation

$V_t(a) = \min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\} \ \forall a$

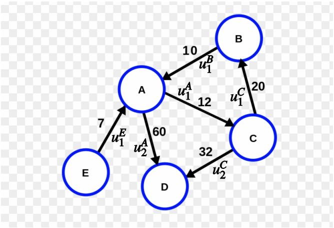
Boundary condition: $V_k(\cdot) := \eta(\cdot)$

Regulator $\mathcal{U}_t(a) := \text{Arg} \min_{u\in U(a)} \{V_{t+1}[f(a,u)] + \varphi(a,u)\}$

The same as an iteration

$$V(\cdot) \xrightarrow{\mathfrak{F}} V_-(\cdot), \ V_-(a) := \min_{u\in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$$

$$V_t(\cdot) = \mathfrak{F}[V_{t+1}(\cdot)] \qquad t = -1, \dots, -\infty$$



$$V_k^{\downarrow}(a) := \min_{\theta\in[-k:0]} V_\theta(a), \qquad V_0^{\downarrow}(\cdot) = \eta(\cdot)$$

$$V^{\downarrow}_{-k}(a) = \min_{\theta\in[-k:0]} V_\theta(a) = \min\left\{\min_{\theta\in[-k:-1]} V_\theta(a); \eta(a)\right\}$$

$$= \min\left\{\min_{u\in U(a)} \min_{\theta\in[-k:-1]} \Big(V_{\theta+1}[f(a,u)] + \varphi(a,u)\Big); \eta(a)\right\}$$

$$= \min\left\{\min_{u\in U(a)} \left[\varphi(a,u) + \min_{\tau\in[-k+1:0]} V_\tau[f(a,u)]\right]; \eta(a)\right\}$$

$$= \min\left\{\min_{u\in U(a)} \left[\varphi(a,u) + V^{\downarrow}_{-k+1}[f(a,u)]\right]; \eta(a)\right\}$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\varphi(\cdot) > 0) \qquad (✇)$$

Bellman equation

$$V^{\downarrow}_{-k}(a) = \min \left\{ \min_{u \in U(a)} \left[\varphi(a, u) + V^{\downarrow}_{-k+1}[f(a, u)] \right]; \eta(a) \right\}, \; V^{\downarrow}_0(\cdot) := \eta(\cdot)$$

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t\in[0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\varphi(\cdot) > 0) \qquad (✾)$$

Bellman equation

$$V^{\downarrow}_{-k}(a) = \min\left\{ \min_{u\in U(a)} \left[\varphi(a,u) + V^{\downarrow}_{-k+1}[f(a,u)]\right]; \eta(a) \right\}, \; V^{\downarrow}_{0}(\cdot) := \eta(\cdot)$$

Theorem

The sequence $V^{\downarrow}_{-k}(\cdot)$ is monotonically decreasing $V^{\downarrow}_{-k}(\cdot) \le V^{\downarrow}_{-k+1}(\cdot)$ and there exists $k \le -1$ such that $V^{\downarrow}_{-k}(\cdot) \equiv V^{\downarrow}_{-k+1}(\cdot)$. For such k, we have $V^{\downarrow}(\cdot) := V^{\downarrow}_{-k}(\cdot) = V^{\downarrow}_{-\theta}(\cdot)\ \forall \theta \ge k$ and $V^{\downarrow}(\cdot)$ is the unique solution of the equation

$V(a) = \min\left\{ \min_{u\in U(a)} \left[\varphi(a,u) + V[f(a,u)]\right]; \eta(a) \right\}$ and simultaneously the Bellman function of the optimization problem.

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k] \qquad (✣)$$

$$x_0 = a, \qquad \textcolor{blue}{k \text{ is freely manipulable}} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\varphi(\cdot) > 0) \qquad (⚮)$$

Bellman equation

$$V^{\downarrow}_{-k}(a) = \min \left\{ \min_{u \in U(a)} \left[\varphi(a,u) + V^{\downarrow}_{-k+1}[f(a,u)] \right]; \eta(a) \right\}, \; V^{\downarrow}_0(\cdot) := \eta(\cdot)$$

V-closing state $\sim V(a) = \eta(a)$

Theorem

The sequence $V^{\downarrow}_{-k}(\cdot)$ is monotonically decreasing $V^{\downarrow}_{-k}(\cdot) \leq V^{\downarrow}_{-k+1}(\cdot)$ and there exists $k \leq -1$ such that $V^{\downarrow}_{-k}(\cdot) \equiv V^{\downarrow}_{-k+1}(\cdot)$. For such k, we have $V^{\downarrow}(\cdot) := V^{\downarrow}_{-k}(\cdot) = V^{\downarrow}_{-\theta}(\cdot) \; \forall \theta \geq k$ and $V^{\downarrow}(\cdot)$ is the unique solution of the equation $V(a) = \min \left\{ \min_{u \in U(a)} \left[\varphi(a,u) + V[f(a,u)] \right]; \eta(a) \right\}$ and simultaneously the Bellman function of the optimization problem.

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t \in [0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\varphi(\cdot) > 0) \qquad (✑)$$

Bellman equation

$$V^{\downarrow}_{-k}(a) = \min \left\{ \min_{u \in U(a)} \Big[\varphi(a,u) + V^{\downarrow}_{-k+1}[f(a,u)] \Big]; \eta(a) \right\}, \; V^{\downarrow}_0(\cdot) := \eta(\cdot)$$

V-closing state $\sim V(a) = \eta(a)$

$\mathcal{U}_V(a) := \operatorname{Arg} \min_{u \in U(a)} \{ V[f(a,u)] + \varphi(a,u) \}$

Theorem

The sequence $V^{\downarrow}_{-k}(\cdot)$ is monotonically decreasing $V^{\downarrow}_{-k}(\cdot) \le V^{\downarrow}_{-k+1}(\cdot)$ and there exists $k \le -1$ such that $V^{\downarrow}_{-k}(\cdot) \equiv V^{\downarrow}_{-k+1}(\cdot)$. For such k, we have $V^{\downarrow}(\cdot) := V^{\downarrow}_{-k}(\cdot) = V^{\downarrow}_{-\theta}(\cdot)\ \forall \theta \ge k$ and $V^{\downarrow}(\cdot)$ is the unique solution of the equation

$$V(a) = \min \left\{ \min_{u \in U(a)} \Big[\varphi(a,u) + V[f(a,u)] \Big]; \eta(a) \right\}$$ and simultaneously the Bellman function of the optimization problem.

Оптимальное планирование с нефиксированной длиной пути

Problem statement

$$x_{t+1} = f(x_t, u_t), \quad u_t \in U[x_t] \quad \forall t \in [0:k) \qquad (✣)$$

$$x_0 = a, \qquad k \text{ is freely manipulable} \qquad (✚)$$

$$I := \sum_{t\in[0:k)} \varphi(x_t, u_t) + \eta(x_k) \to \min \qquad (\varphi(\cdot) > 0) \qquad (✍)$$

Bellman equation

$$V^{\downarrow}_{-k}(a) = \min \left\{ \min_{u\in U(a)} \left[\varphi(a,u) + V^{\downarrow}_{-k+1}[f(a,u)] \right]; \eta(a) \right\}, \; V^{\downarrow}_0(\cdot) := \eta(\cdot)$$

V-closing state $\sim V(a) = \eta(a)$
$\mathfrak{U}_V(a) := \operatorname{Arg\,min}_{u\in U(a)} \{V[f(a,u)] + \varphi(a,u)\}$

Theorem

The set of $V^{\downarrow}$-closing states is not empty. Optimal process $\Leftrightarrow u_t \in \mathfrak{U}_V(x_t)$ and the process terminates terminates upon arrival at a closing state.

Theorem

The sequence $V^{\downarrow}_{-k}(\cdot)$ is monotonically decreasing $V^{\downarrow}_{-k}(\cdot) \leq V^{\downarrow}_{-k+1}(\cdot)$ and there exists $k \leq -1$ such that $V^{\downarrow}_{-k}(\cdot) \equiv V^{\downarrow}_{-k+1}(\cdot)$. For such k, we have $V^{\downarrow}(\cdot) := V^{\downarrow}_{-k}(\cdot) = V^{\downarrow}_{-\theta}(\cdot) \; \forall \theta \geq k$ and $V^{\downarrow}(\cdot)$ is the unique solution of the equation

$$V(a) = \min \left\{ \min_{u\in U(a)} \left[\varphi(a,u) + V[f(a,u)] \right]; \eta(a) \right\}$$ and simultaneously the Bellman function of the optimization problem.

Доказательства.

Доказательства.

Lemma

Let $[\{x_t\}_{t=0}^{k}, \{u_t\}_{t=0}^{k-1}]$ be an optimal process. Then the process $[\{x_t\}_{t=\theta}^{k}, \{u_t\}_{t=\theta}^{k-1}]$ is also optimal for any $\theta \in [0:k]$, the state x_k is stopping, $u_t \in \mathcal{U}_{V^\downarrow}(x_t)$, and $V^\downarrow(x_0) = \sum_{t=0}^{\theta-1} \varphi(x_t, u_t) + V^\downarrow(x_\theta)$.

Доказательства.

Lemma

Let $[\{x_t\}_{t=0}^k, \{u_t\}_{t=0}^{k-1}]$ be an optimal process. Then the process $[\{x_t\}_{t=\theta}^k, \{u_t\}_{t=\theta}^{k-1}]$ is also optimal for any $\theta \in [0:k]$, the state x_k is stopping, $u_t \in \mathcal{U}_{V^\downarrow}(x_t)$, and $V^\downarrow(x_0) = \sum_{t=0}^{\theta-1} \varphi(x_t, u_t) + V^\downarrow(x_\theta)$.

$$V^\downarrow(x_0) = \varphi(x_0, u_0) + V^\downarrow(x_1) = \varphi(a, u_0) + V^\downarrow[f(a, u_0)], \quad a := x_0$$

$$V^\downarrow(a) = \min\left\{ \min_{u \in U(a)} \Big[\varphi(a, u) + V^\downarrow[f(a, u)]\Big]; \eta(a) \right\}$$

Доказательства.

Lemma

Let $[\{x_t\}_{t=0}^{k}, \{u_t\}_{t=0}^{k-1}]$ be an optimal process. Then the process $[\{x_t\}_{t=\theta}^{k}, \{u_t\}_{t=\theta}^{k-1}]$ is also optimal for any $\theta \in [0:k]$, the state x_k is stopping, $u_t \in \mathcal{U}_{V\downarrow}(x_t)$, and $V^{\downarrow}(x_0) = \sum_{t=0}^{\theta-1} \varphi(x_t, u_t) + V^{\downarrow}(x_\theta)$.

Lemma

Any solution of the equation $V(a) = \min\left\{\min_{u \in U(a)}\left[\varphi(a,u) + V[f(a,u)]\right]; \eta(a)\right\}$ is a lower estimate of the Bellman function.

Доказательства.

Lemma

Let $[\{x_t\}_{t=0}^{k}, \{u_t\}_{t=0}^{k-1}]$ be an optimal process. Then the process $[\{x_t\}_{t=\theta}^{k}, \{u_t\}_{t=\theta}^{k-1}]$ is also optimal for any $\theta \in [0:k]$, the state x_k is stopping, $u_t \in \mathcal{U}_{V\downarrow}(x_t)$, and $V^{\downarrow}(x_0) = \sum_{t=0}^{\theta-1} \varphi(x_t, u_t) + V^{\downarrow}(x_\theta)$.

Lemma

Any solution of the equation $V(a) = \min\left\{ \min\limits_{u \in U(a)} \left[\varphi(a,u) + V[f(a,u)]\right]; \eta(a) \right\}$ is a lower estimate of the Bellman function.

$$V(a) = \min\left\{ \min_{u \in U(a)} \left[\varphi(a,u) + V[f(a,u)]\right]; \eta(a) \right\}$$

$$I = \sum_{t=0}^{k-1} \varphi(x_t, u_t) + \eta[x_k] \geq \sum_{t=0}^{k-1} \varphi(x_t, u_t) + V[x_k]$$

Доказательства.

Lemma

Let $\left[\{x_t\}_{t=0}^{k}, \{u_t\}_{t=0}^{k-1}\right]$ be an optimal process. Then the process $\left[\{x_t\}_{t=\theta}^{k}, \{u_t\}_{t=\theta}^{k-1}\right]$ is also optimal for any $\theta \in [0:k]$, the state x_k is stopping, $u_t \in \mathcal{U}_{V^\downarrow}(x_t)$, and $V^\downarrow(x_0) = \sum_{t=0}^{\theta-1} \varphi(x_t, u_t) + V^\downarrow(x_\theta)$.

Lemma

Any solution of the equation $V(a) = \min\left\{\min_{u \in U(a)} \left[\varphi(a,u) + V[f(a,u)]\right]; \eta(a)\right\}$ is a lower estimate of the Bellman function.

$$V(a) = \min\left\{\min_{u \in U(a)} \left[\varphi(a,u) + V[f(a,u)]\right]; \eta(a)\right\}$$

$$I = \sum_{t=0}^{k-1} \varphi(x_t, u_t) + \eta[x_k] \geq \sum_{t=0}^{k-1} \varphi(x_t, u_t) + V[x_k]$$

$$= \sum_{t=0}^{k-2} \varphi(x_t, u_t) + \varphi(x_{k-1}, u_{k-1}) + V[f(x_{k-1}, u_{k-1})]$$

Доказательства.

Lemma

Let $[\{x_t\}_{t=0}^{k}, \{u_t\}_{t=0}^{k-1}]$ be an optimal process. Then the process $[\{x_t\}_{t=\theta}^{k}, \{u_t\}_{t=\theta}^{k-1}]$ is also optimal for any $\theta \in [0:k]$, the state x_k is stopping, $u_t \in \mathcal{U}_{V\downarrow}(x_t)$, and $V^{\downarrow}(x_0) = \sum_{t=0}^{\theta-1} \varphi(x_t, u_t) + V^{\downarrow}(x_\theta)$.

Lemma

Any solution of the equation $V(a) = \min\left\{\min\limits_{u\in U(a)}\Big[\varphi(a,u) + V[f(a,u)]\Big]; \eta(a)\right\}$ is a lower estimate of the Bellman function.

$$V(a) = \min\left\{\min_{u\in U(a)}\Big[\varphi(a,u) + V[f(a,u)]\Big]; \eta(a)\right\}$$

$$I = \sum_{t=0}^{k-1} \varphi(x_t, u_t) + \eta[x_k] \geq \sum_{t=0}^{k-1} \varphi(x_t, u_t) + V[x_k]$$

$$= \sum_{t=0}^{k-2} \varphi(x_t, u_t) + \varphi(x_{k-1}, u_{k-1}) + V[f(x_{k-1}, u_{k-1})]$$

$$\geq \sum_{t=0}^{k-2} \varphi(x_t, u_t) + V[x_{k-1}] \geq \ldots \geq V[x_0]$$

Доказательства.

Lemma

Let $[\{x_t\}_{t=0}^{k}, \{u_t\}_{t=0}^{k-1}]$ be an optimal process. Then the process $[\{x_t\}_{t=\theta}^{k}, \{u_t\}_{t=\theta}^{k-1}]$ is also optimal for any $\theta \in [0:k]$, the state x_k is stopping, $u_t \in \mathcal{U}_{V^\downarrow}(x_t)$, and $V^\downarrow(x_0) = \sum_{t=0}^{\theta-1} \varphi(x_t, u_t) + V^\downarrow(x_\theta)$.

Lemma

Any solution of the equation $V(a) = \min\left\{\min\limits_{u \in U(a)}\Big[\varphi(a,u) + V[f(a,u)]\Big]; \eta(a)\right\}$ is a lower estimate of the Bellman function.

Lemma

Let $u_t \in \mathcal{U}_V(x_t)$ and the process terminates upon arrival at the first stopping state. Then this process is optimal.

$$V(a) = \min\left\{\min_{u \in U(a)}\Big[\varphi(a,u) + V[f(a,u)]\Big]; \eta(a)\right\}$$

$$I = \sum_{t=0}^{k-1} \varphi(x_t, u_t) + \eta[x_k] \geq \sum_{t=0}^{k-1} \varphi(x_t, u_t) + V[x_k]$$

$$= \sum_{t=0}^{k-2} \varphi(x_t, u_t) + \varphi(x_{k-1}, u_{k-1}) + V[f(x_{k-1}, u_{k-1})]$$

$$\geq \sum_{t=0}^{k-2} \varphi(x_t, u_t) + V[x_{k-1}] \geq \ldots \geq V[x_0]$$