Seminar: Search and Optimization

2. Mathematical Background

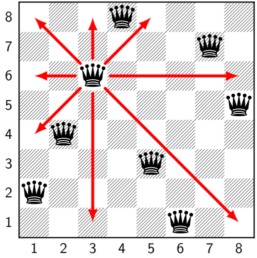
Gabi Röger

Universität Basel

September 25, 2014

Sets, tuples and relations

Example: Eight Queens

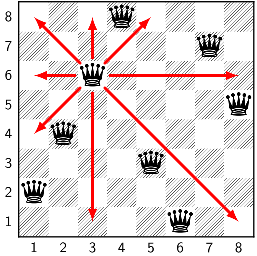


Variables:
One variable for each row

Domain of each variable:
possible positions in the row

Constraints:
No two queens may threaten each other.

Example: Eight Queens



Variables:
One variable for each row

Domain of each variable:
possible positions in the row

Constraints:
No two queens may threaten each other.

How can we formally specify such a constraint satisfaction problem?

Are there mathematical operations our algorithms can use?

Sets

- Set: unordered collection of distinguishable objects, where each object is contained at most once.
- Specification:
  - Explicitly by listing all members, e. g. $A = \{1, 2, 3\}$
  - Implicitly by giving a property that characterizes all members, e. g. $A = \{x \mid x \in \mathbb{N} \text{ and } 1 \leq x \leq 3\}$

Sets

- Set: unordered collection of distinguishable objects, where each object is contained at most once.
- Specification:
    - Explicitly by listing all members, e. g. $A = \{1, 2, 3\}$
    - Implicitly by giving a property that characterizes all members, e. g. $A = \{x \mid x \in \mathbb{N} \text{ and } 1 \leq x \leq 3\}$
- $e \in S$: e is in the set S (an element or member of the set)
- $e \notin S$: e is not in the set S

Sets

- Set: unordered collection of distinguishable objects, where each object is contained at most once.
- Specification:
  - Explicitly by listing all members, e. g. $A = \{1, 2, 3\}$
  - Implicitly by giving a property that characterizes all members, e. g. $A = \{x \mid x \in \mathbb{N} \text{ and } 1 \leq x \leq 3\}$
- $e \in S$: e is in the set S (an element or member of the set)
- $e \notin S$: e is not in the set S
- $A \subseteq B$: A is a subset of B, i. e., every element of A is an element of B.
- $A \subset B$: A is a proper subset of B, i. e., $A \subseteq B$ and $A \neq B$.

Sets

- Intersection $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
- Union $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- Difference $A - B = \{x \mid x \in A \text{ and } x \notin B\}$

Sets

- Intersection $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
- Union $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- Difference $A - B = \{x \mid x \in A \text{ and } x \notin B\}$
- Empty set $\emptyset = \{\}$
- Sets A and B are disjoint if $A \cap B = \emptyset$.
- Size or cardinality $|S|$ of set S: number of elements in S

Sets

- Intersection $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
- Union $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- Difference $A - B = \{x \mid x \in A \text{ and } x \notin B\}$
- Empty set $\emptyset = \{\}$
- Sets A and B are disjoint if $A \cap B = \emptyset$.
- Size or cardinality $|S|$ of set S: number of elements in S
- If a set specifies the possible values for a CSP variable then we call it the domain associated with the variable.

Tuples and Cartesian Product

- *k-tuple*: sequence of k objects denoted by $(o_1, \ldots, o_k)$
- Objects in a tuple do not need to be distinct, i. e., an object can occur more than once.
- An object in the tuple is called a component.

Tuples and Cartesian Product

- k-tuple: sequence of k objects denoted by $(o_1, \ldots, o_k)$
- Objects in a tuple do not need to be distinct, i. e., an object can occur more than once.
- An object in the tuple is called a component.
- (Cartesian) product $D_1 \times D_2 \times \cdots \times D_n$ of sets $D_1, \ldots, D_n$: set of all n-tuples $(o_1, \ldots, o_n)$ such that $o_1 \in D_1, \ldots, o_n \in D_n$.
- Example:
  $\{a, b\} \times \{1, 2, 3\} = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$

Relations

- Let $\mathcal{S} = \{x_1, \ldots, x_k\}$ be a set of variables and $D_1, \ldots, D_k$ the domains associated with these variables.
- A relation R on $\mathcal{S}$ is a subset of $D_1 \times \cdots \times D_k$.
- $\mathcal{S}$ is called the scope of R (written $scope(R)$).
- k is the arity of R.
- For $k = 1, 2, 3$, R is called a unary, binary, or ternary relation, respectively.
- $R = D_1 \times \cdots \times D_k$ is the universal relation.
- If we want to make the scope $\mathcal{S}$ of a relation explicit, we often write $R_{\mathcal{S}}$.

Representing Relations

For the examples, we use variables x_1 and x_2 with associated domains $D_1 = \{1, 2\}$ and $D_2 = \{2, 4, 6\}$.

- Explicitly: $R = \{(1, 2), (1, 6), (2, 4), (2, 6)\}$
- Implicitly: $R = \{(a, b) \mid a \in D_1, b \in D_2, b = 2a \text{ or } b = 6\}$
- Table representation:

x_1	x_2
1	2
1	6
2	4
2	6

Additional Representation for Binary Relations

Matrix representation:
Matrix with entry 1 if tuple is in the relation and entry 0 otherwise.

$$\begin{array}{cc} & \begin{array}{ccc} 2 & 4 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

Operations on Relations

For two relations R and R' on the same scope $\mathcal{S}$,

- the union $R \cup R'$,
- intersection $R \cap R'$, and
- difference $R - R'$

are defined by the respective set operations.

The scope of the result is $\mathcal{S}$.

Union, Intersection and Difference of Relations – Examples

Variables x_1, x_2, x_3

Domains

$$D_1 = \{1, 2, 3\}$$

$$D_2 = \{a, b\}$$

$$D_3 = \{2, 4, 6\}$$

R

x_1	x_2	x_3
1	a	4
2	b	2
2	b	6
3	a	4

R'

x_1	x_2	x_3
1	a	4
1	a	6
2	b	2

$R \cup R'$

x_1	x_2	x_3
1	a	4
1	a	6
2	b	2
2	b	6
3	a	4

$R \cap R'$

x_1	x_2	x_3
1	a	4
2	b	2

$R - R'$

x_1	x_2	x_3
2	b	6
3	a	4

Operations on Relations – Selection

For a relation R, let x be a variable from the scope of R and let e be a value from the associated domain.

The selection $\sigma_{x=e}(R)$ is the subset of R that contains all elements where the component for x is e.

R

x_1	x_2	x_3
1	a	4
2	a	2
2	b	6
3	a	4

$\sigma_{x_1=2}(R)$

x_1	x_2	x_3
2	a	2
2	b	6

The scope of the resulting relation is the scope of R.

Operations on Relations – Selection

We use the abbreviations $\sigma_{x_1=e_1,\ldots,x_n=e_n}(R)$ and $\sigma_{(x_1,\ldots,x_n)=(e_1,\ldots,e_n)}(R)$ for $\sigma_{x_1=e_1}(\ldots(\sigma_{x_n=e_n}(R))\ldots)$.

R

x_1	x_2	x_3
1	a	4
2	a	2
2	a	6
2	b	6
3	a	4

$\sigma_{x_1=2,x_3=6}(R)$

x_1	x_2	x_3
2	a	6
2	b	6

Operations on Relations – Projection

For a relation R and $Y \subseteq scope(R)$, the projection $\pi_Y(R)$ with scope Y consists of all tuples that can be constructed from a tuple in R by removing all components for variables that are not in Y.

R

x_1	x_2	x_3
1	a	2
1	a	4
2	b	6
3	a	4

$\pi_{\{x_1,x_2\}}(R)$

x_1	x_2
1	a
2	b
3	a

Put simply: In the table representation, remove all columns for variables that are not in Y. Afterwards, remove duplicate rows.

Operations on Relations – Join

For two relations R and R' the join operator $R \bowtie R'$ combines each tuple from R with all tuples from R' that agree on the common variables.

R

x_1	x_2	x_3
1	a	2
1	a	6
3	a	4
3	b	4

R'

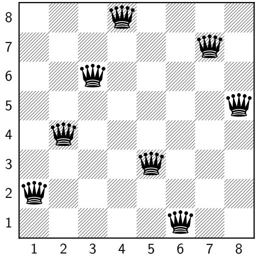
x_1	x_3	x_4
1	2	$*$
1	2	$-$
2	4	$+$
3	4	$+$

$R \bowtie R'$

x_1	x_2	x_3	x_4
1	a	2	$*$
1	a	2	$-$
3	a	4	$+$
3	b	4	$+$

The scope of $R \bowtie R'$ is $scope(R) \cup scope(R')$.

Example: Eight Queens



Variables:
One variable for each row

Domain of each variable:
possible positions in the row

Constraints:
No two queens may threaten each other.

Formally, the CSP is given by

- A set of variables $V = \{x_1, x_2, \dots, x_8\}$
- Associated domains $D_i = \{1, \dots, 8\}$ for $1 \leq i \leq 8$
- For $1 \leq i < j \leq 8$ a constraint
  $R_{ij} = \{(p, q) \mid p \neq q \text{ and } |j - i| \neq |p - q|\}$

Questions on Sets, Tuples, Relations, ...

Questions?

Graphs

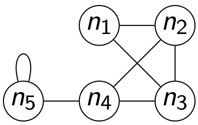
Undirected Graphs I

An undirected graph $G = (V, E)$ is given by

- a finite set V of vertices (or nodes), and
- a finite set $E \subseteq \{\{u, v\} \mid u, v \in V\}$ of edges (or arcs).

For $e = \{u, v\} \in E$, we say that e connects v and v' and that u and v are adjacent or neighbors. The degree $d(v)$ of node v is the number of adjacent neighbors.

$$V = (\{n_1, n_2, n_3, n_4, n_5\},\\ \{\{n_1, n_2\}, \{n_1, n_3\}, \{n_2, n_3\}, \{n_2, n_4\}, \{n_3, n_4\}, \{n_4, n_5\}, \{n_5\}\})$$



Undirected Graphs II

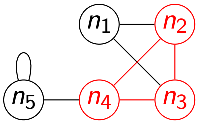
- Path: Sequence $e_1, e_2, \ldots, e_k$ of edges such that e_i and e_{i+1} share an end point (for $1 \leq i < k$)
- Alternatively: Path as sequence $v_0, \ldots, v_k$ of vertices, where $\{v_i, v_{i+1}\} \in E$ for $0 \leq i < k$.
- In path $v_0, \ldots, v_k$, v_0 is the start vertex, v_k the end vertex and the path has length k.

Undirected Graphs II

- Path: Sequence $e_1, e_2, \ldots, e_k$ of edges such that e_i and e_{i+1} share an end point (for $1 \leq i < k$)
- Alternatively: Path as sequence $v_0, \ldots, v_k$ of vertices, where $\{v_i, v_{i+1}\} \in E$ for $0 \leq i < k$.
- In path $v_0, \ldots, v_k$, v_0 is the start vertex, v_k the end vertex and the path has length k.
- A path is simple if no vertex occurs more than once.
- A cycle is a path whose start and end vertices are the same.
- A cycle is simple if without the end vertex it is a simple path.

Undirected Graphs III

- An undirected graph without any cycles is a tree.
- If there is a path between any two edges, the graph is connected.
- A graph is complete if any two nodes are adjacent.
- For $S \subseteq V$, the subgraph relative to S is $G_S = \{S, \{\{u, v\} \mid \{u, v\} \in E \text{ and } \{u, v\} \subseteq S\}\}$.
- A clique in a graph is a complete subgraph.



The subgraph relative to $\{n_2, n_3, n_4\}$ is a clique.

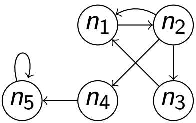
Directed Graphs I

A directed graph (or digraph) $G = (V, E)$ is given by

- a finite set V of vertices (or nodes), and
- a finite set $E \subseteq \{(u, v) \mid u, v \in V\}$ of edges (or arcs).

Edge $e = (u, v) \in E$ (also written $u \to v$) is directed from start vertex u to end vertex v.

$$V = (\{n_1, n_2, n_3, n_4, n_5\},$$
$$\{(n_1, n_2), (n_2, n_1), (n_2, n_3), (n_2, n_4), (n_1, n_3), (n_4, n_5), (n_5, n_5)\})$$



Directed Graphs II

- Outdegree of a node u: number of edges with start vertex u
- Indegree of a node u: number of edges with end vertex u
- Node u is a parent of node v if there is an edge (u, v).
  The set of all parents of v is denoted $pa(v)$.
- Node u is a child of node v if there is an edge (v, u).
  The set of all children of v is denoted $ch(v)$.

Directed Graphs II

- Outdegree of a node u: number of edges with start vertex u
- Indegree of a node u: number of edges with end vertex u
- Node u is a parent of node v if there is an edge (u, v).
  The set of all parents of v is denoted $pa(v)$.
- Node u is a child of node v if there is an edge (v, u).
  The set of all children of v is denoted $ch(v)$.
- Directed path: Sequence $e_1, e_2, \ldots, e_k$ of edges such that end vertex of e_i is start vertex of e_{i+1} (for $1 \leq i < k$)
- A directed cycle is a directed path whose start and end vertices are the same.

Directed Graphs II

- Outdegree of a node u: number of edges with start vertex u
- Indegree of a node u: number of edges with end vertex u
- Node u is a parent of node v if there is an edge (u, v). The set of all parents of v is denoted $pa(v)$.
- Node u is a child of node v if there is an edge (v, u). The set of all children of v is denoted $ch(v)$.
- Directed path: Sequence $e_1, e_2, \ldots, e_k$ of edges such that end vertex of e_i is start vertex of e_{i+1} (for $1 \leq i < k$)
- A directed cycle is a directed path whose start and end vertices are the same.
- A digraph is acyclic if it has no directed cycles.
- A digraph is strongly connected if for each two difference vertices u and v there is a directed path from u to v.

Questions on Graphs

Questions?

Complexity

Asymptotic Runtime Analysis

How fast is a given algorithm?

- For an algorithm, consider worst-case running time $T(n)$ over all inputs of fixed size n.
- Evaluate algorithms by the growth rate of $T(n)$ with increasing n.

Asymptotic Runtime Analysis

How fast is a given algorithm?

- For an algorithm, consider worst-case running time $T(n)$ over all inputs of fixed size n.
- Evaluate algorithms by the growth rate of $T(n)$ with increasing n.
- $O(g(n)) = \{f(n) \mid$ there exist positive constants c and n_0 s.t. $0 \leq f(n) \leq cg(n)$ for all $n > n_0\}$

  Examples: $n^2 + 3n \in O(n^2)$ and $O(n^3) \subset O(2^n)$

Asymptotic Runtime Analysis

How fast is a given algorithm?

- For an algorithm, consider worst-case running time $T(n)$ over all inputs of fixed size n.
- Evaluate algorithms by the growth rate of $T(n)$ with increasing n.
- $O(g(n)) = \{f(n) \mid$ there exist positive constants c and n_0 s.t. $0 \leq f(n) \leq cg(n)$ for all $n > n_0\}$

  Examples: $n^2 + 3n \in O(n^2)$ and $O(n^3) \subset O(2^n)$
- $o(g(n)) = \{f(n) \mid$ for any positive constant c exists an n_0 s.t. $0 \leq f(n) \leq cg(n)$ for all $n > n_0\}$

Asymptotic Runtime Analysis – More Roughly

How fast is a given algorithm?
(Please do not tell me too many details. . .)

- Polynomial of degree d:
  function $f(n) = \sum_{i=1}^{d} a_i n^i$, where a_i are constants.
- An algorithm is tractable if $T(n)$ is bound by a polynomial, i. e., if $T(n) \in O(n^k)$ for some k.
  Otherwise it is intractable.

Asymptotic Runtime Analysis – More Roughly

How fast is a given algorithm?
(Please do not tell me too many details...)

- Polynomial of degree d:
  function $f(n) = \sum_{i=1}^{d} a_i n^i$, where a_i are constants.
- An algorithm is tractable if $T(n)$ is bound by a polynomial, i. e., if $T(n) \in O(n^k)$ for some k.
  Otherwise it is intractable.
- For a constant a, $f(n) = a^n$ is an exponential function.
- For $a > 0$ it holds for all b that $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$.

Complexity of Problems

How fast can the best algorithm for a problem class be?

- If there is a tractable algorithm, the problem class is tractable.
- NP-complete problem classes are believed to require exponential runtime in the worst case.
- For NP-complete problem classes, a potential solution can be verified in polynomial time.

Complexity of Problems

How fast can the best algorithm for a problem class be?

- If there is a tractable algorithm, the problem class is tractable.
- NP-complete problem classes are believed to require exponential runtime in the worst case.
- For NP-complete problem classes, a potential solution can be verified in polynomial time.
- Bad news: Constraint Satisfaction Problems are NP-complete.
  - There is probably no tractable algorithm.
  - Idea 1: Find algorithms that are sufficiently fast in most of the cases
  - Idea 2: Identify tractable subclasses

Questions on Complexity

Questions?