Linear Programming

Abhiram Ranade

March 16, 2016

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

What if no polytime algorithm can be designed?

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

What if no polytime algorithm can be designed?

- Use exponential time algorithms.

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

What if no polytime algorithm can be designed?

- Use exponential time algorithms.
  - Use brute force search.

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

What if no polytime algorithm can be designed?

- Use exponential time algorithms.
  - Use brute force search.
  - Use integer linear programming.

Next

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

What if no polytime algorithm can be designed?

- Use exponential time algorithms.
  - Use brute force search.
  - Use integer linear programming. Next
- Do not worry about getting optimal solutions, but accept near optimal solutions if they can be found quickly.

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

What if no polytime algorithm can be designed?

- Use exponential time algorithms.
  - Use brute force search.
  - Use integer linear programming. Next
- Do not worry about getting optimal solutions, but accept near optimal solutions if they can be found quickly.
- Can we prove that no polytime algorithms exist?

Combinatorial optimization in practice

Suppose you are solving some combinatorial optimization problem, e.g. constructing the course timetable for IITB.

Hope: Design polytime algorithm: greedy, dynamic programming..

What if no polytime algorithm can be designed?

- Use exponential time algorithms.
  - Use brute force search.
  - Use integer linear programming. Next
- Do not worry about getting optimal solutions, but accept near optimal solutions if they can be found quickly.
- Can we prove that no polytime algorithms exist?

Theory of NP-completeness, soon.

Linear Programming (original, continuous version of ILP)

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Input: $m \times n$ matrix A, $m \times 1$ vector b, and an $n \times 1$ vector c.

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Input: $m \times n$ matrix A, $m \times 1$ vector b, and an $n \times 1$ vector c.

Output: $n \times 1$ vector $x \geq 0$ such that $Ax \geq b$, and $c^T x$ is minimum.

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Input: $m \times n$ matrix A, $m \times 1$ vector b, and an $n \times 1$ vector c.

Output: $n \times 1$ vector $x \geq 0$ such that $Ax \geq b$, and $c^T x$ is minimum.

Algorithm: Can be solved in time polynomial in m, n.

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Input: $m \times n$ matrix A, $m \times 1$ vector b, and an $n \times 1$ vector c.

Output: $n \times 1$ vector $x \geq 0$ such that $Ax \geq b$, and $c^T x$ is minimum.

Algorithm: Can be solved in time polynomial in m, n.

Other forms: Polytime possible even if

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Input: $m \times n$ matrix A, $m \times 1$ vector b, and an $n \times 1$ vector c.

Output: $n \times 1$ vector $x \geq 0$ such that $Ax \geq b$, and $c^T x$ is minimum.

Algorithm: Can be solved in time polynomial in m, n.

Other forms: Polytime possible even if

- Some rows of $Ax \geq b$ are equalities, or inequalities $\geq, \leq$, but not $\neq, <, >$.

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Input: $m \times n$ matrix A, $m \times 1$ vector b, and an $n \times 1$ vector c.

Output: $n \times 1$ vector $x \geq 0$ such that $Ax \geq b$, and $c^T x$ is minimum.

Algorithm: Can be solved in time polynomial in m, n.

Other forms: Polytime possible even if

- Some rows of $Ax \geq b$ are equalities, or inequalities $\geq, \leq$, but not $\neq, <, >$.
- We drop some non-negativity constraints $x \geq 0$

Linear Programming (original, continuous version of ILP)

"Canonical Form"

Input: $m \times n$ matrix A, $m \times 1$ vector b, and an $n \times 1$ vector c.

Output: $n \times 1$ vector $x \geq 0$ such that $Ax \geq b$, and $c^T x$ is minimum.

Algorithm: Can be solved in time polynomial in m, n.

Other forms: Polytime possible even if

- Some rows of $Ax \geq b$ are equalities, or inequalities $\geq, \leq$, but not $\neq, <, >$.
- We drop some non-negativity constraints $x \geq 0$
- We ask for maximization instead of minimization.

Example

“Wheat costs Rs 4/kg and rice Rs 3/kg. Each kg of wheat gives 2 units of proteins, and each kg of rice 1. Each kg of wheat or rice gives 1 unit of carbohydrates. I need 4 units of proteins and 3 units of carbohydrates. What is my most economical diet?”

Example

"Wheat costs Rs 4/kg and rice Rs 3/kg. Each kg of wheat gives 2 units of proteins, and each kg of rice 1. Each kg of wheat or rice gives 1 unit of carbohydrates. I need 4 units of proteins and 3 units of carbohydrates. What is my most economical diet?"

Let $x_1, x_2 =$ amount in kg of wheat in rice I should buy.

Example

"Wheat costs Rs 4/kg and rice Rs 3/kg. Each kg of wheat gives 2 units of proteins, and each kg of rice 1. Each kg of wheat or rice gives 1 unit of carbohydrates. I need 4 units of proteins and 3 units of carbohydrates. What is my most economical diet?"

Let $x_1, x_2 =$ amount in kg of wheat in rice I should buy.

Problem can be formulated as an LP:

Example

"Wheat costs Rs 4/kg and rice Rs 3/kg. Each kg of wheat gives 2 units of proteins, and each kg of rice 1. Each kg of wheat or rice gives 1 unit of carbohydrates. I need 4 units of proteins and 3 units of carbohydrates. What is my most economical diet?"

Let $x_1, x_2 =$ amount in kg of wheat in rice I should buy.

Problem can be formulated as an LP:

$\min 4x_1 + 3x_2$ such that

Example

"Wheat costs Rs 4/kg and rice Rs 3/kg. Each kg of wheat gives 2 units of proteins, and each kg of rice 1. Each kg of wheat or rice gives 1 unit of carbohydrates. I need 4 units of proteins and 3 units of carbohydrates. What is my most economical diet?"

Let $x_1, x_2 =$ amount in kg of wheat in rice I should buy.

Problem can be formulated as an LP:

$\min 4x_1 + 3x_2$ such that

$2x_1 + x_2 \geq 4$
$x_1 + x_2 \geq 3$
$x_1, x_2 \geq 0$

Example

“Wheat costs Rs 4/kg and rice Rs 3/kg. Each kg of wheat gives 2 units of proteins, and each kg of rice 1. Each kg of wheat or rice gives 1 unit of carbohydrates. I need 4 units of proteins and 3 units of carbohydrates. What is my most economical diet?”

Let $x_1, x_2 =$ amount in kg of wheat in rice I should buy.

Problem can be formulated as an LP:

$\min 4x_1 + 3x_2$ such that

$2x_1 + x_2 \geq 4$
$x_1 + x_2 \geq 3$
$x_1, x_2 \geq 0$

Geometric solution ...

Remarks

Remarks

Expressive: many real life problems can be expressed as an LP.

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

- Equalities = hyperplanes. Inequalities = half spaces.

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

- Equalities = hyperplanes. Inequalities = half spaces.
- Feasible region: polyhedral region in n dimensional space.

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

- Equalities = hyperplanes. Inequalities = half spaces.
- Feasible region: polyhedral region in n dimensional space.
- Goal: Find the extreme point on the polyhedron in the direction indicated by the objective function.

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

- Equalities = hyperplanes. Inequalities = half spaces.
- Feasible region: polyhedral region in n dimensional space.
- Goal: Find the extreme point on the polyhedron in the direction indicated by the objective function.

Algorithmic strategies: Very rough sketches

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

- Equalities = hyperplanes. Inequalities = half spaces.
- Feasible region: polyhedral region in n dimensional space.
- Goal: Find the extreme point on the polyhedron in the direction indicated by the objective function.

Algorithmic strategies: Very rough sketches

- "Greedy" : Start with some feasible solution. Then move in the direction indicated by the objective function. If you hit a boundary, move in the best direction possible.

  Simplex method, exptime but behaves well in practice

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

- Equalities = hyperplanes. Inequalities = half spaces.
- Feasible region: polyhedral region in n dimensional space.
- Goal: Find the extreme point on the polyhedron in the direction indicated by the objective function.

Algorithmic strategies: Very rough sketches

- "Greedy" : Start with some feasible solution. Then move in the direction indicated by the objective function. If you hit a boundary, move in the best direction possible.

  Simplex method, exptime but behaves well in practice
- "Divide and conquer" : Keep cutting off pieces of the polyhedron that are determined to be useless.

  Ellipsoid method, polytime, but slow in practice

Remarks

Expressive: many real life problems can be expressed as an LP.

Geometric view:

- Equalities = hyperplanes. Inequalities = half spaces.
- Feasible region: polyhedral region in n dimensional space.
- Goal: Find the extreme point on the polyhedron in the direction indicated by the objective function.

Algorithmic strategies: Very rough sketches

- "Greedy" : Start with some feasible solution. Then move in the direction indicated by the objective function. If you hit a boundary, move in the best direction possible.

  Simplex method, exptime but behaves well in practice
- "Divide and conquer" : Keep cutting off pieces of the polyhedron that are determined to be useless.

  Ellipsoid method, polytime, but slow in practice
- "Quasi greedy" : Start with a feasible solution inside the polyhedron, but dont go too close to the boundary too quickly.

  Interior point method, polytime, good in practice

Integer Linear Programming

Linear programming with the constraint that the solution vector x, must have only integer components.

$$\min c^T x \quad \text{s.t.} \quad x \geq 0,\ Ax \geq b,\ x_i \text{ integer}$$

Integer Linear Programming

Linear programming with the constraint that the solution vector x, must have only integer components.

$$\min c^T x \quad \text{s.t.} \quad x \geq 0,\ Ax \geq b,\ x_i \text{ integer}$$

Not expected to have a polynomial time algorithm.

Integer Linear Programming

Linear programming with the constraint that the solution vector x, must have only integer components.

$$\min c^T x \quad \text{s.t.} \quad x \geq 0,\ Ax \geq b,\ x_i \text{ integer}$$

Not expected to have a polynomial time algorithm.

But used very commonly because many combinatorial optimization problems can be expressed easily as ILPs.

Integer Linear Programming

Linear programming with the constraint that the solution vector x, must have only integer components.

$$\min c^T x \quad \text{s.t.} \quad x \geq 0,\ Ax \geq b,\ x_i \text{ integer}$$

Not expected to have a polynomial time algorithm.

But used very commonly because many combinatorial optimization problems can be expressed easily as ILPs.

- Have a variable to represent every decision you want to take. Variable takes values 0 or 1.

Integer Linear Programming

Linear programming with the constraint that the solution vector x, must have only integer components.

$$\min c^T x \quad \text{s.t.} \quad x \geq 0,\ Ax \geq b,\ x_i \text{ integer}$$

Not expected to have a polynomial time algorithm.

But used very commonly because many combinatorial optimization problems can be expressed easily as ILPs.

- Have a variable to represent every decision you want to take. Variable takes values 0 or 1.
- Express constraints in the problem using linear inequalities on the variables.

Integer Linear Programming

Linear programming with the constraint that the solution vector x, must have only integer components.

$$\min c^T x \quad \text{s.t.} \quad x \geq 0,\ Ax \geq b,\ x_i \text{ integer}$$

Not expected to have a polynomial time algorithm.

But used very commonly because many combinatorial optimization problems can be expressed easily as ILPs.

- Have a variable to represent every decision you want to take. Variable takes values 0 or 1.
- Express constraints in the problem using linear inequalities on the variables.

Commercial solvers available: lpsolve, CPLEX, ...

Integer Linear Programming

Linear programming with the constraint that the solution vector x, must have only integer components.

$$\min c^T x \quad \text{s.t.} \quad x \geq 0,\ Ax \geq b,\ x_i \text{ integer}$$

Not expected to have a polynomial time algorithm.

But used very commonly because many combinatorial optimization problems can be expressed easily as ILPs.

- Have a variable to represent every decision you want to take. Variable takes values 0 or 1.
- Express constraints in the problem using linear inequalities on the variables.

Commercial solvers available: lpsolve, CPLEX, ...

Medium sized problems can be solved fast.

Maximum Independent set as ILP

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

No polytime algorithm known, none expected.

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

No polytime algorithm known, none expected.

Can we express this as an ILP?

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

No polytime algorithm known, none expected.

Can we express this as an ILP?

x_v : 1 if in MIS, 0 if not.

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

No polytime algorithm known, none expected.

Can we express this as an ILP?

x_v : 1 if in MIS, 0 if not.
$\max \sum_u x_u$

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

No polytime algorithm known, none expected.

Can we express this as an ILP?

x_v : 1 if in MIS, 0 if not.
$\max \sum_u x_u$ s.t. $\forall (u, v) \in E: \ x_u + x_v \leq 1$

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

No polytime algorithm known, none expected.

Can we express this as an ILP?

x_v : 1 if in MIS, 0 if not.
$\max \sum_u x_u$ s.t. $\forall (u, v) \in E : \; x_u + x_v \leq 1 \;\; \forall v \in V : x_v \geq 0, x_v \leq 1$

Maximum Independent set as ILP

Definition: In a graph $G = (V, E)$, $V' \subseteq V$ is said to be independent if $u, v \in V' \Rightarrow (u, v) \notin E$.

An independent set of maximum possible size is said to be a maximum independent set.

No polytime algorithm known, none expected.

Can we express this as an ILP?

x_v : 1 if in MIS, 0 if not.
max $\sum_u x_u$ s.t. $\forall (u, v) \in E : \; x_u + x_v \leq 1 \;\; \forall v \in V : x_v \geq 0, x_v \leq 1$
Example..

Vertex cover

A subset V' of the edges of a graph are said to be a vertex cover if every edge is incident on some vertex in V'.

Vertex cover

A subset V' of the edges of a graph are said to be a vertex cover if every edge is incident on some vertex in V'.

No polytime algorithm known for finding a minimum vertex cover.

Vertex cover

A subset V' of the edges of a graph are said to be a vertex cover if every edge is incident on some vertex in V'.

No polytime algorithm known for finding a minimum vertex cover.

Can you express this as an ILP?

Vertex cover

A subset V' of the edges of a graph are said to be a vertex cover if every edge is incident on some vertex in V'.

No polytime algorithm known for finding a minimum vertex cover.

Can you express this as an ILP?
x_v : 1 if in MVC, 0 otherwise.

Vertex cover

A subset V' of the edges of a graph are said to be a vertex cover if every edge is incident on some vertex in V'.

No polytime algorithm known for finding a minimum vertex cover.

Can you express this as an ILP?
x_v : 1 if in MVC, 0 otherwise.
min $\sum_v x_v$ s.t. $\forall (u, v) \in E : x_u + x_v \geq 1$

Vertex cover

A subset V' of the edges of a graph are said to be a vertex cover if every edge is incident on some vertex in V'.

No polytime algorithm known for finding a minimum vertex cover.

Can you express this as an ILP?
x_v : 1 if in MVC, 0 otherwise.
$\min \sum_v x_v$ s.t. $\forall (u, v) \in E : x_u + x_v \geq 1$

Will you be able to write programs that (a) read in the description of a graph and outputs the matrix A and the vectors b, c (b) Take the output of the ILP solver and print out the set of edges constituting the minimum edge cover.

Timetable construction as ILP

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j.
Integer k, indicating number of available slots.

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j.
Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j. Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Variables: For all courses c and slots s define variable x_{cs} which is to take value 1 if course c is scheduled in slot s.

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j. Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Variables: For all courses c and slots s define variable x_{cs} which is to take value 1 if course c is scheduled in slot s.

Total nk variables.

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j. Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Variables: For all courses c and slots s define variable x_{cs} which is to take value 1 if course c is scheduled in slot s.

Total nk variables.

Constraints:

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j. Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Variables: For all courses c and slots s define variable x_{cs} which is to take value 1 if course c is scheduled in slot s.

Total nk variables.

Constraints:

- Every course c must be assigned exactly 1 slot:

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j. Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Variables: For all courses c and slots s define variable x_{cs} which is to take value 1 if course c is scheduled in slot s.

Total nk variables.

Constraints:

- Every course c must be assigned exactly 1 slot:
  $\forall c : \sum_s x_{cs} = 1$.

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j. Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Variables: For all courses c and slots s define variable x_{cs} which is to take value 1 if course c is scheduled in slot s.

Total nk variables.

Constraints:

- Every course c must be assigned exactly 1 slot:
  $\forall c : \sum_s x_{cs} = 1.$
- If some student s takes courses c, c' then they must not be in the same slot:
  $\forall s, c, c' \ s.t. \ a_{sc} = a_{sc'} = 1 : x_{cs} + x_{c's} \leq 1$

Timetable construction as ILP

Input: Matrix A, where $a_{ij} = 1$ if student i is enrolled for course j.
Integer k, indicating number of available slots.
Output: $S[1..n]$ where $S[i] \in \{1, \ldots, k\}$ for all i and $S[i] \neq S[j]$ if courses i, j have a common student.

Variables: For all courses c and slots s define variable x_{cs} which is to take value 1 if course c is scheduled in slot s.

Total nk variables.

Constraints:

- Every course c must be assigned exactly 1 slot:
  $\forall c : \sum_s x_{cs} = 1.$
- If some student s takes courses c, c' then they must not be in the same slot:
  $\forall s, c, c' \ s.t. \ a_{sc} = a_{sc'} = 1 : x_{cs} + x_{c's} \leq 1$

Prof. Jayendran Venkateswaran and his team have used this as the central idea in creating the institute timetable.

Remarks

Remarks

- An important idea in expressing a problem as in ILP: interpret a variable as a logical variable ($0 =$ no, $1 =$ yes) and an arithmetic variable at the same time.

Remarks

- An important idea in expressing a problem as in ILP: interpret a variable as a logical variable ($0 =$ no, $1 =$ yes) and an arithmetic variable at the same time.
- ILP can be thought of as a language: problems of different kinds can be expressed using the language of ILP.

Remarks

- An important idea in expressing a problem as in ILP: interpret a variable as a logical variable ($0 =$ no, $1 =$ yes) and an arithmetic variable at the same time.
- ILP can be thought of as a language: problems of different kinds can be expressed using the language of ILP.
- "Expressing MIS using ILP" = "Reducing MIS to ILP"

Remarks

- An important idea in expressing a problem as in ILP: interpret a variable as a logical variable ($0 =$ no, $1 =$ yes) and an arithmetic variable at the same time.
- ILP can be thought of as a language: problems of different kinds can be expressed using the language of ILP.
- "Expressing MIS using ILP" = "Reducing MIS to ILP"
- Reduction = Translation of the instance of the original problem to an instance of ILP + translation of the result of ILP instance to the result of original problem.