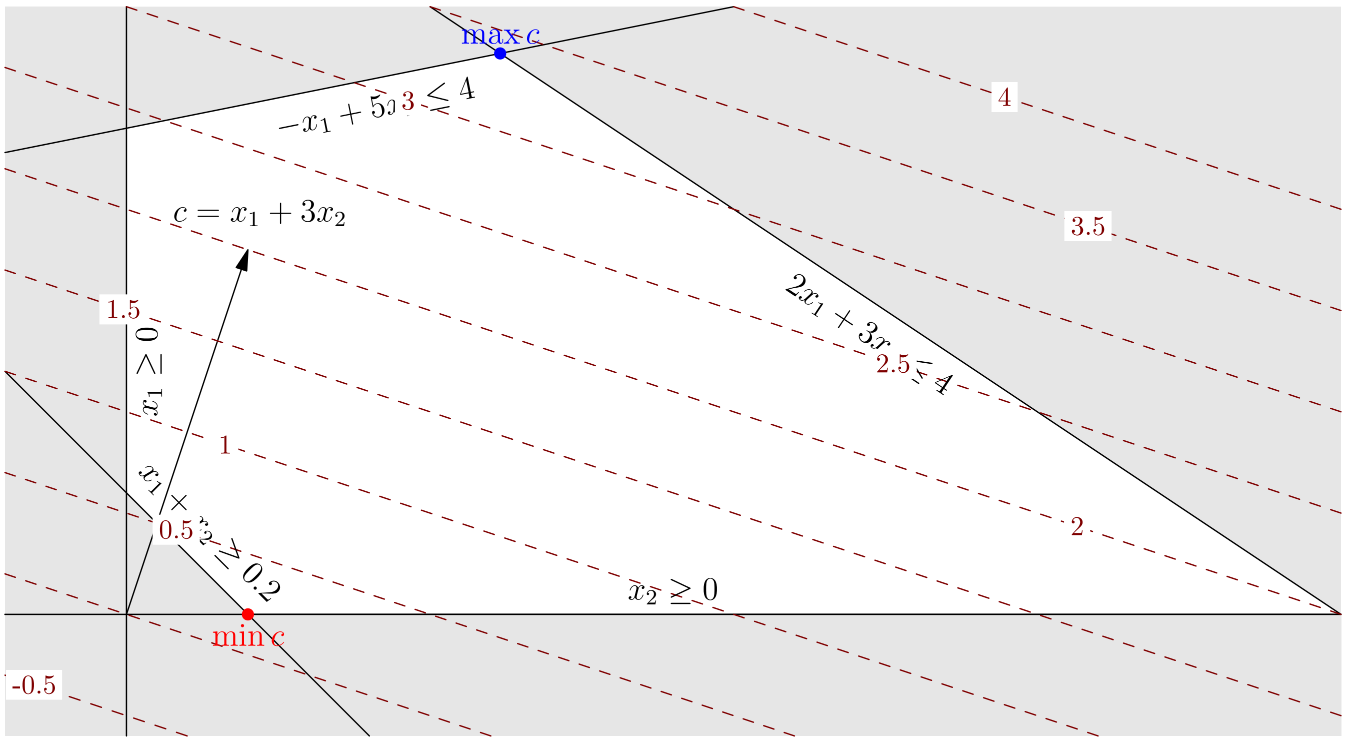
Algorithms and Analysis

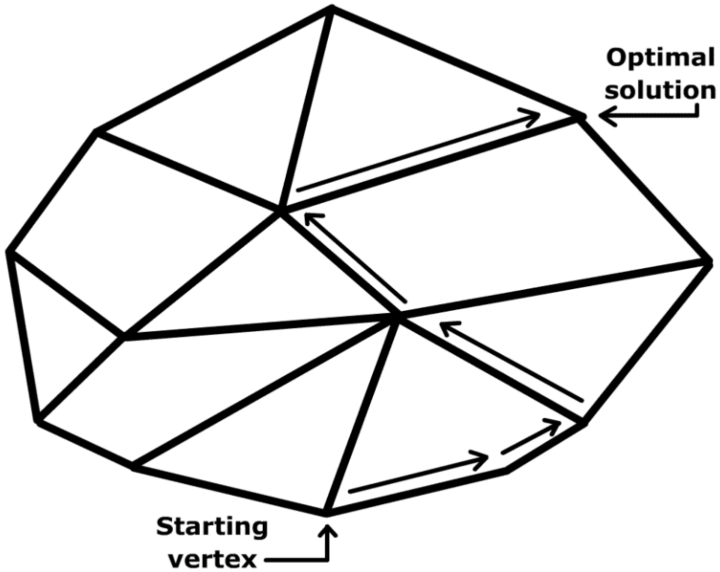
Lesson 27: *Use Linear Programmings*



linear programming, applications

Outline

1. **Examples**
2. Linear Programs
3. Properties of Solution
4. Normal Form



Going Shopping

- Suppose we have a number of food stuffs which we label with indices $f \in \mathcal{F}$
- The price of food stuff f per kilogram we denote p_f
- We are interested in buying a selection of foods $\boldsymbol{x} = (x_f | f \in \mathcal{F})$ where x_f is the quantity (in kg) of food f
- We want to minimise the total price $\sum_f p_f \, x_f = \boldsymbol{p} \cdot \boldsymbol{x}$
- However we want to ensure that the food has enough vitamins

Going Shopping

- Suppose we have a number of food stuffs which we label with indices $f \in \mathcal{F}$
- The price of food stuff f per kilogram we denote p_f
- We are interested in buying a selection of foods $\boldsymbol{x} = (x_f | f \in \mathcal{F})$ where x_f is the quantity (in kg) of food f
- We want to minimise the total price $\sum_f p_f \, x_f = \boldsymbol{p} \cdot \boldsymbol{x}$
- However we want to ensure that the food has enough vitamins

Going Shopping

- Suppose we have a number of food stuffs which we label with indices $f \in \mathcal{F}$
- The price of food stuff f per kilogram we denote p_f
- We are interested in buying a selection of foods $\boldsymbol{x} = (x_f | f \in \mathcal{F})$ where x_f is the quantity (in kg) of food f
- We want to minimise the total price $\sum_f p_f \, x_f = \boldsymbol{p} \cdot \boldsymbol{x}$
- However we want to ensure that the food has enough vitamins

Going Shopping

- Suppose we have a number of food stuffs which we label with indices $f \in \mathcal{F}$
- The price of food stuff f per kilogram we denote p_f
- We are interested in buying a selection of foods $\boldsymbol{x} = (x_f | f \in \mathcal{F})$ where x_f is the quantity (in kg) of food f
- We want to minimise the total price $\sum_f p_f \, x_f = \boldsymbol{p} \cdot \boldsymbol{x}$
- However we want to ensure that the food has enough vitamins

Going Shopping

- Suppose we have a number of food stuffs which we label with indices $f \in \mathcal{F}$
- The price of food stuff f per kilogram we denote p_f
- We are interested in buying a selection of foods $\boldsymbol{x} = (x_f | f \in \mathcal{F})$ where x_f is the quantity (in kg) of food f
- We want to minimise the total price $\sum_f p_f \, x_f = \boldsymbol{p} \cdot \boldsymbol{x}$
- However we want to ensure that the food has enough vitamins

Nutrition

- We consider the set of vitamins $\mathcal{V}$
- Let $A_{v\,f}$ be the quantity of vitamin v in food stuff f
- Let b_v be the minimum daily requirement of vitamin v
- We therefore require

$$\forall v \in \mathcal{V} \qquad \sum_{f \in \mathcal{F}} A_{v\,f}\, x_f \geq b_v$$

Nutrition

- We consider the set of vitamins $\mathcal{V}$
- Let A_{vf} be the quantity of vitamin v in food stuff f
- Let b_v be the minimum daily requirement of vitamin v
- We therefore require

$$\forall v \in \mathcal{V} \qquad \sum_{f \in \mathcal{F}} A_{vf}\, x_f \geq b_v$$

Nutrition

- We consider the set of vitamins $\mathcal{V}$
- Let A_{vf} be the quantity of vitamin v in food stuff f
- Let b_v be the minimum daily requirement of vitamin v
- We therefore require

$$\forall v \in \mathcal{V} \qquad \sum_{f \in \mathcal{F}} A_{vf}\, x_f \geq b_v$$

Nutrition

- We consider the set of vitamins $\mathcal{V}$
- Let A_{vf} be the quantity of vitamin v in food stuff f
- Let b_v be the minimum daily requirement of vitamin v
- We therefore require

$$\forall v \in \mathcal{V} \qquad \sum_{f \in \mathcal{F}} A_{vf}\, x_f \geq b_v$$

Optimisation Problem

- We can write the food shopping problem as

$$\min_{\boldsymbol{x}} \boldsymbol{p} \cdot \boldsymbol{x} \quad \text{subject to} \quad \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \quad \text{and} \quad \boldsymbol{x} \geq \mathbf{0}$$

- Note that the inequalities involving vectors means that each component must be satisfied, i.e.

$$\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \quad \Rightarrow \quad \forall v \in \mathcal{V} \quad \sum_{f \in \mathcal{F}} A_{vf}\, x_f \geq b_v$$

$$\boldsymbol{x} \geq \mathbf{0} \quad \Rightarrow \quad \forall f \in \mathcal{F} \quad x_f \geq 0$$

- This is an example of a "**linear program**"

Optimisation Problem

- We can write the food shopping problem as

$$\min_{\boldsymbol{x}} \boldsymbol{p} \cdot \boldsymbol{x} \quad \text{subject to} \quad \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \quad \text{and} \quad \boldsymbol{x} \geq \mathbf{0}$$

- Note that the inequalities involving vectors means that each component must be satisfied, i.e.

$$\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \quad \Rightarrow \quad \forall v \in \mathcal{V} \quad \sum_{f \in \mathcal{F}} A_{v\,f}\, x_f \geq b_v$$

$$\boldsymbol{x} \geq \mathbf{0} \quad \Rightarrow \quad \forall f \in \mathcal{F} \quad x_f \geq 0$$

- This is an example of a "**linear program**"

Optimisation Problem

- We can write the food shopping problem as

$$\min_{\boldsymbol{x}} \boldsymbol{p} \cdot \boldsymbol{x} \quad \text{subject to} \quad \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \quad \text{and} \quad \boldsymbol{x} \geq \mathbf{0}$$

- Note that the inequalities involving vectors means that each component must be satisfied, i.e.

$$\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \quad \Rightarrow \quad \forall v \in \mathcal{V} \quad \sum_{f \in \mathcal{F}} A_{vf}\, x_f \geq b_v$$

$$\boldsymbol{x} \geq \mathbf{0} \quad \Rightarrow \quad \forall f \in \mathcal{F} \quad x_f \geq 0$$

- This is an example of a "**linear program**"

Transportation

- We consider a set of factories $\mathcal{F}$ producing a set of commodities $\mathcal{C}$
- The amount of commodity c produced by factory f we denote by x_{cf}
- The shipping cost of commodity c from factory f to the retailer of c we denote by p_{cf}
- We want to choose x_{cf} to minimise the transportation costs

$$\sum_{c \in \mathcal{C}, f \in \mathcal{F}} p_{cf} \, x_{cf}$$

- However, we have constraints. . .

Transportation

- We consider a set of factories $\mathcal{F}$ producing a set of commodities $\mathcal{C}$
- The amount of commodity c produced by factory f we denote by x_{cf}
- The shipping cost of commodity c from factory f to the retailer of c we denote by p_{cf}
- We want to choose x_{cf} to minimise the transportation costs

$$\sum_{c \in \mathcal{C}, f \in \mathcal{F}} p_{cf} \, x_{cf}$$

- However, we have constraints. . .

Transportation

- We consider a set of factories $\mathcal{F}$ producing a set of commodities $\mathcal{C}$
- The amount of commodity c produced by factory f we denote by x_{cf}
- The shipping cost of commodity c from factory f to the retailer of c we denote by p_{cf}
- We want to choose x_{cf} to minimise the transportation costs

$$\sum_{c\in\mathcal{C}, f\in\mathcal{F}} p_{cf}\, x_{cf}$$

- However, we have constraints. . .

Transportation

- We consider a set of factories $\mathcal{F}$ producing a set of commodities $\mathcal{C}$
- The amount of commodity c produced by factory f we denote by x_{cf}
- The shipping cost of commodity c from factory f to the retailer of c we denote by p_{cf}
- We want to choose x_{cf} to minimise the transportation costs

$$\sum_{c \in \mathcal{C}, f \in \mathcal{F}} p_{cf} \, x_{cf}$$

- However, we have constraints. . .

Transportation

- We consider a set of factories $\mathcal{F}$ producing a set of commodities $\mathcal{C}$
- The amount of commodity c produced by factory f we denote by x_{cf}
- The shipping cost of commodity c from factory f to the retailer of c we denote by p_{cf}
- We want to choose x_{cf} to minimise the transportation costs

$$\sum_{c \in \mathcal{C}, f \in \mathcal{F}} p_{cf} \, x_{cf}$$

- However, we have constraints. . .

Constraints

- Each factory can only produce a certain overall tonnage of commodities

$$\sum_{c \in \mathcal{C}} x_{cf} \le b_f \qquad \forall f \in \mathcal{F}$$

  where b_f is the maximum production capacity of factory f

- The total demand for each commodity is d_c so

$$\sum_{f \in \mathcal{F}} x_{cf} = d_c \qquad \forall c \in \mathcal{C}$$

- We can only produce positive amounts, i.e. $x_{cf} \ge 0$

Constraints

- Each factory can only produce a certain overall tonnage of commodities

$$\sum_{c \in \mathcal{C}} x_{cf} \leq b_f \qquad \forall f \in \mathcal{F}$$

where b_f is the maximum production capacity of factory f

- The total demand for each commodity is d_c so

$$\sum_{f \in \mathcal{F}} x_{cf} = d_c \qquad \forall c \in \mathcal{C}$$

- We can only produce positive amounts, i.e. $x_{cf} \geq 0$

Constraints

- Each factory can only produce a certain overall tonnage of commodities

$$\sum_{c \in \mathcal{C}} x_{cf} \leq b_f \qquad \forall f \in \mathcal{F}$$

where b_f is the maximum production capacity of factory f

- The total demand for each commodity is d_c so

$$\sum_{f \in \mathcal{F}} x_{cf} = d_c \qquad \forall c \in \mathcal{C}$$

- We can only produce positive amounts, i.e. $x_{cf} \geq 0$

Constraints

- Each factory can only produce a certain overall tonnage of commodities

$$\sum_{c \in \mathcal{C}} x_{cf} \leq b_f \qquad \forall f \in \mathcal{F}$$

  where b_f is the maximum production capacity of factory f

- The total demand for each commodity is d_c so

$$\sum_{f \in \mathcal{F}} x_{cf} = d_c \qquad \forall c \in \mathcal{C}$$

- We can only produce positive amounts, i.e. $x_{cf} \geq 0$

Linear Program

- We can write the full problem as

$$\min_{\boldsymbol{x}} \sum_{c\in\mathcal{C},f\in\mathcal{F}} p_{cf}\, x_{cf}$$

subject to

$$\sum_{c\in\mathcal{C}} x_{cf} \le b_f \qquad \forall f \in \mathcal{F}$$

$$\sum_{f\in\mathcal{F}} x_{cf} = d_c \qquad \forall c \in \mathcal{C}$$

$$x_{cf} \ge 0 \qquad \forall c \in \mathcal{C}, \quad \forall f \in \mathcal{F}$$

Linear Program

- We can write the full problem as

$$\min_{\boldsymbol{x}} \sum_{c\in\mathcal{C},f\in\mathcal{F}} p_{cf}\, x_{cf}$$

subject to

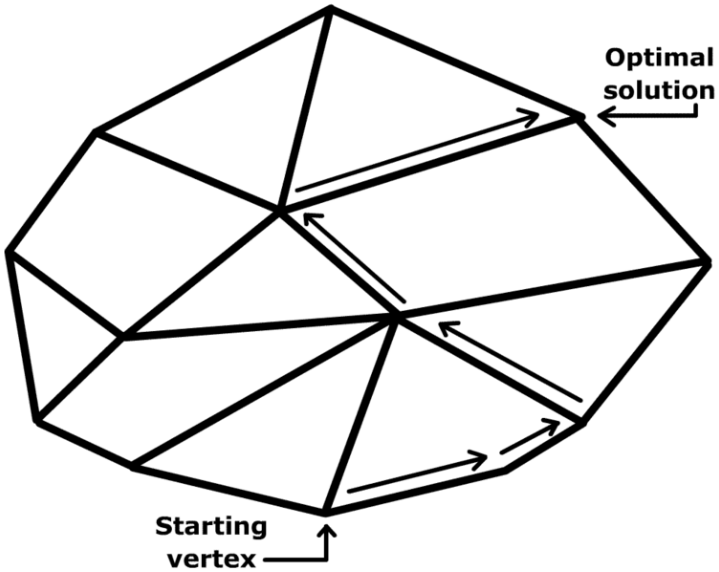
$$\sum_{c\in\mathcal{C}} x_{cf} \le b_f \qquad \forall f \in \mathcal{F}$$

$$\sum_{f\in\mathcal{F}} x_{cf} = d_c \qquad \forall c \in \mathcal{C}$$

$$x_{cf} \ge 0 \qquad \forall c \in \mathcal{C}, \quad \forall f \in \mathcal{F}$$

Outline

1. Examples
2. **Linear Programs**
3. Properties of Solution
4. Normal Form



General Linear Programs

- Linear programs are problems that can be formulated as follows

$$\min_{\boldsymbol{x}} \boldsymbol{c} \cdot \boldsymbol{x}$$

subject to

$$\mathbf{A}^{\leq} \boldsymbol{x} \leq \boldsymbol{b}^{\leq}, \quad \mathbf{A}^{\geq} \boldsymbol{x} \geq \boldsymbol{b}^{\geq}, \quad \mathbf{A}^{=} \boldsymbol{x} = \boldsymbol{b}^{=}, \quad \boldsymbol{x} \geq \mathbf{0}$$

- Note in the previous example it was convenient to use two indices c and f to denote the components x_{cf}, however, it still has this structure

General Linear Programs

- Linear programs are problems that can be formulated as follows

$$\min_{\boldsymbol{x}} \boldsymbol{c} \cdot \boldsymbol{x}$$

subject to

$$\mathbf{A}^{\leq} \boldsymbol{x} \leq \boldsymbol{b}^{\leq}, \quad \mathbf{A}^{\geq} \boldsymbol{x} \geq \boldsymbol{b}^{\geq}, \quad \mathbf{A}^{=} \boldsymbol{x} = \boldsymbol{b}^{=}, \quad \boldsymbol{x} \geq \mathbf{0}$$

- Note in the previous example it was convenient to use two indices c and f to denote the components x_{cf}, however, it still has this structure

Maximising

- We can also maximise rather than minimise
- Whether we want to maximise or minimise will depend on the application
- Note that

$$\max_{\boldsymbol{x}} \boldsymbol{c} \cdot \boldsymbol{x} \quad \equiv \quad \min_{\boldsymbol{x}} (-\boldsymbol{c}) \cdot \boldsymbol{x}$$

- We can thus always reformulate a maximisation problem as a minimisation problem and vice versa

Maximising

- We can also maximise rather than minimise
- Whether we want to maximise or minimise will depend on the application
- Note that

$$\max_{\boldsymbol{x}} \boldsymbol{c} \cdot \boldsymbol{x} \quad \equiv \quad \min_{\boldsymbol{x}} (-\boldsymbol{c}) \cdot \boldsymbol{x}$$

- We can thus always reformulate a maximisation problem as a minimisation problem and vice versa

Maximising

- We can also maximise rather than minimise
- Whether we want to maximise or minimise will depend on the application
- Note that

$$\max_{\boldsymbol{x}} \boldsymbol{c} \cdot \boldsymbol{x} \quad \equiv \quad \min_{\boldsymbol{x}} (-\boldsymbol{c}) \cdot \boldsymbol{x}$$

- We can thus always reformulate a maximisation problem as a minimisation problem and vice versa

Maximising

- We can also maximise rather than minimise
- Whether we want to maximise or minimise will depend on the application
- Note that

$$\max_{\boldsymbol{x}} \boldsymbol{c} \cdot \boldsymbol{x} \quad \equiv \quad \min_{\boldsymbol{x}} (-\boldsymbol{c}) \cdot \boldsymbol{x}$$

- We can thus always reformulate a maximisation problem as a minimisation problem and vice versa

Linear Program Applications

- A huge number of problems can be mapped to linear programming problems
- Or modelled as linear (even when they're not, e.g. oil extraction)
- Realistic problems might have many more constraints and large number of variables
- State of the art solvers can deal with problems with hundreds of thousands or even millions of variables

Linear Program Applications

- A huge number of problems can be mapped to linear programming problems
- Or modelled as linear (even when they're not, e.g. oil extraction)
- Realistic problems might have many more constraints and large number of variables
- State of the art solvers can deal with problems with hundreds of thousands or even millions of variables

Linear Program Applications

- A huge number of problems can be mapped to linear programming problems
- Or modelled as linear (even when they're not, e.g. oil extraction)
- Realistic problems might have many more constraints and large number of variables
- State of the art solvers can deal with problems with hundreds of thousands or even millions of variables

Linear Program Applications

- A huge number of problems can be mapped to linear programming problems
- Or modelled as linear (even when they're not, e.g. oil extraction)
- Realistic problems might have many more constraints and large number of variables
- State of the art solvers can deal with problems with hundreds of thousands or even millions of variables

Key Features

- There are three key features of linear programs

  1. The cost (objective function) is linear in x_i ($\boldsymbol{c} \cdot \boldsymbol{x}$)
  2. The constraints are linear in x_i (e.g. $\mathbf{A}_1 \boldsymbol{x} \leq b_1$)
  3. The component of $\boldsymbol{x}$ are non-negative (i.e. $x_i \geq 0$)

- These are very special features, very often they don't apply, but a surprising large number of problems can be formulated as linear programming problems

Key Features

- There are three key features of linear programs
  1. The cost (objective function) is linear in x_i ($\boldsymbol{c} \cdot \boldsymbol{x}$)
  2. The constraints are linear in x_i (e.g. $\mathbf{A}_1 \boldsymbol{x} \leq b_1$)
  3. The component of $\boldsymbol{x}$ are non-negative (i.e. $x_i \geq 0$)
- These are very special features, very often they don't apply, but a surprising large number of problems can be formulated as linear programming problems

History

- Linear programming was "invented" by Leonid Kantorovich in 1939 to help Soviet Russia maximise its production
- It was kept secret during the war, but was finally made public in 1947 when George Dantzig published the **simplex method** which still today is a standard method for solving linear programs
- John von Neumann developed the idea of duality (you can turn a maximisation problem for a set of variables $\boldsymbol{x}$ into a minimisation problem for a dual set of variables $\boldsymbol{\lambda}$ associated with each constraint)
- von Neumann used this idea as the basis for "game theory"

History

- Linear programming was “invented” by Leonid Kantorovich in 1939 to help Soviet Russia maximise its production

- It was kept secret during the war, but was finally made public in 1947 when George Dantzig published the **simplex method** which still today is a standard method for solving linear programs

- John von Neumann developed the idea of duality (you can turn a maximisation problem for a set of variables $\boldsymbol{x}$ into a minimisation problem for a dual set of variables $\boldsymbol{\lambda}$ associated with each constraint)

- von Neumann used this idea as the basis for “game theory”

History

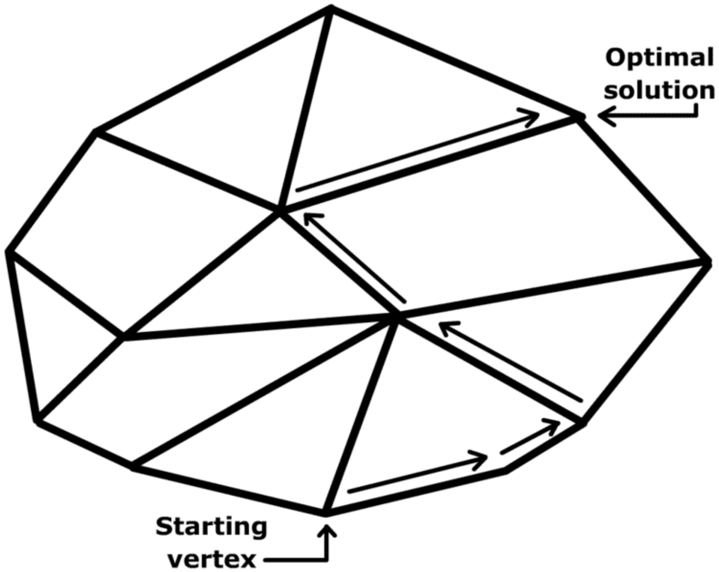
- Linear programming was "invented" by Leonid Kantorovich in 1939 to help Soviet Russia maximise its production
- It was kept secret during the war, but was finally made public in 1947 when George Dantzig published the **simplex method** which still today is a standard method for solving linear programs
- John von Neumann developed the idea of duality (you can turn a maximisation problem for a set of variables $\boldsymbol{x}$ into a minimisation problem for a dual set of variables $\boldsymbol{\lambda}$ associated with each constraint)
- von Neumann used this idea as the basis for "game theory"

History

- Linear programming was "invented" by Leonid Kantorovich in 1939 to help Soviet Russia maximise its production
- It was kept secret during the war, but was finally made public in 1947 when George Dantzig published the **simplex method** which still today is a standard method for solving linear programs
- John von Neumann developed the idea of duality (you can turn a maximisation problem for a set of variables $\boldsymbol{x}$ into a minimisation problem for a dual set of variables $\boldsymbol{\lambda}$ associated with each constraint)
- von Neumann used this idea as the basis for "game theory"

Outline

1. Examples
2. Linear Programs
3. **Properties of Solution**
4. Normal Form



Structure of Linear Programs

- Before we go into the details of solving linear programs its useful to consider the structure of the solutions
- The set of x that satisfy all the constraints is known as the set of **feasible solutions**
- The set of feasible solutions may be empty in which case it is impossible to satisfy all the constraints
- This is rather disappointing, but usually doesn't happen if we have formulated a sensible problem

Structure of Linear Programs

- Before we go into the details of solving linear programs its useful to consider the structure of the solutions
- The set of x that satisfy all the constraints is known as the set of **feasible solutions**
- The set of feasible solutions may be empty in which case it is impossible to satisfy all the constraints
- This is rather disappointing, but usually doesn't happen if we have formulated a sensible problem

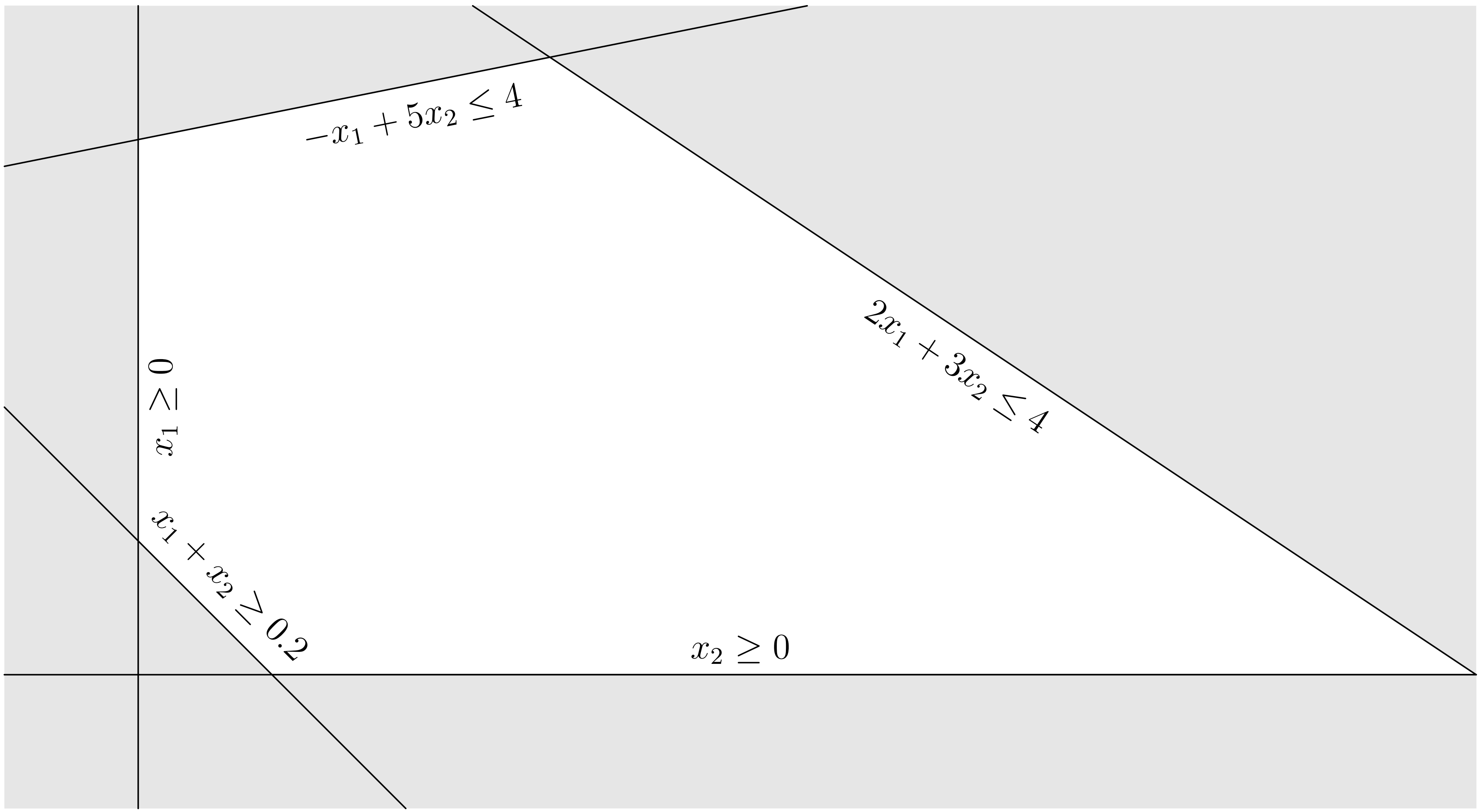
Structure of Linear Programs

- Before we go into the details of solving linear programs its useful to consider the structure of the solutions
- The set of x that satisfy all the constraints is known as the set of **feasible solutions**
- The set of feasible solutions may be empty in which case it is impossible to satisfy all the constraints
- This is rather disappointing, but usually doesn't happen if we have formulated a sensible problem

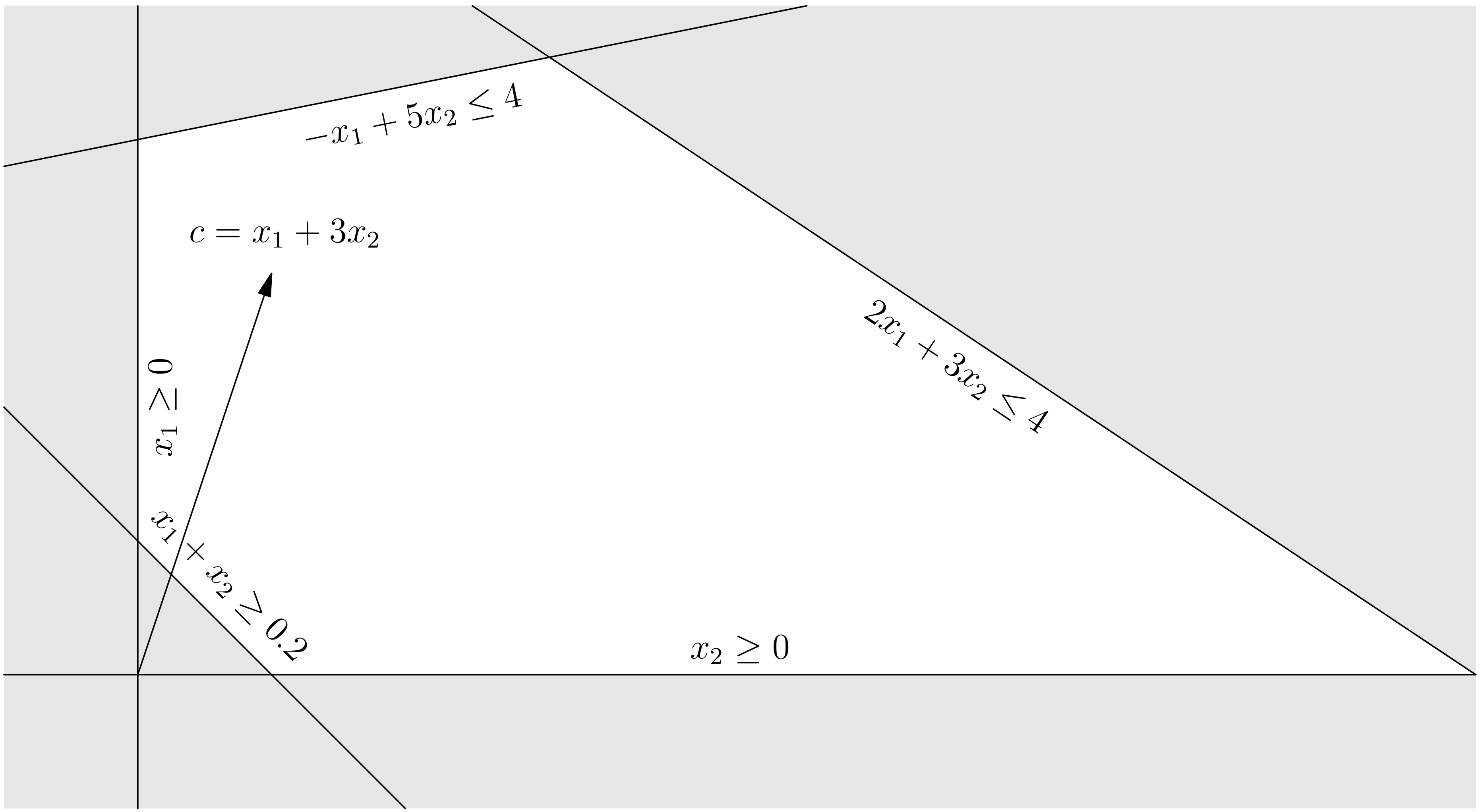
Structure of Linear Programs

- Before we go into the details of solving linear programs its useful to consider the structure of the solutions
- The set of x that satisfy all the constraints is known as the set of **feasible solutions**
- The set of feasible solutions may be empty in which case it is impossible to satisfy all the constraints
- This is rather disappointing, but usually doesn't happen if we have formulated a sensible problem

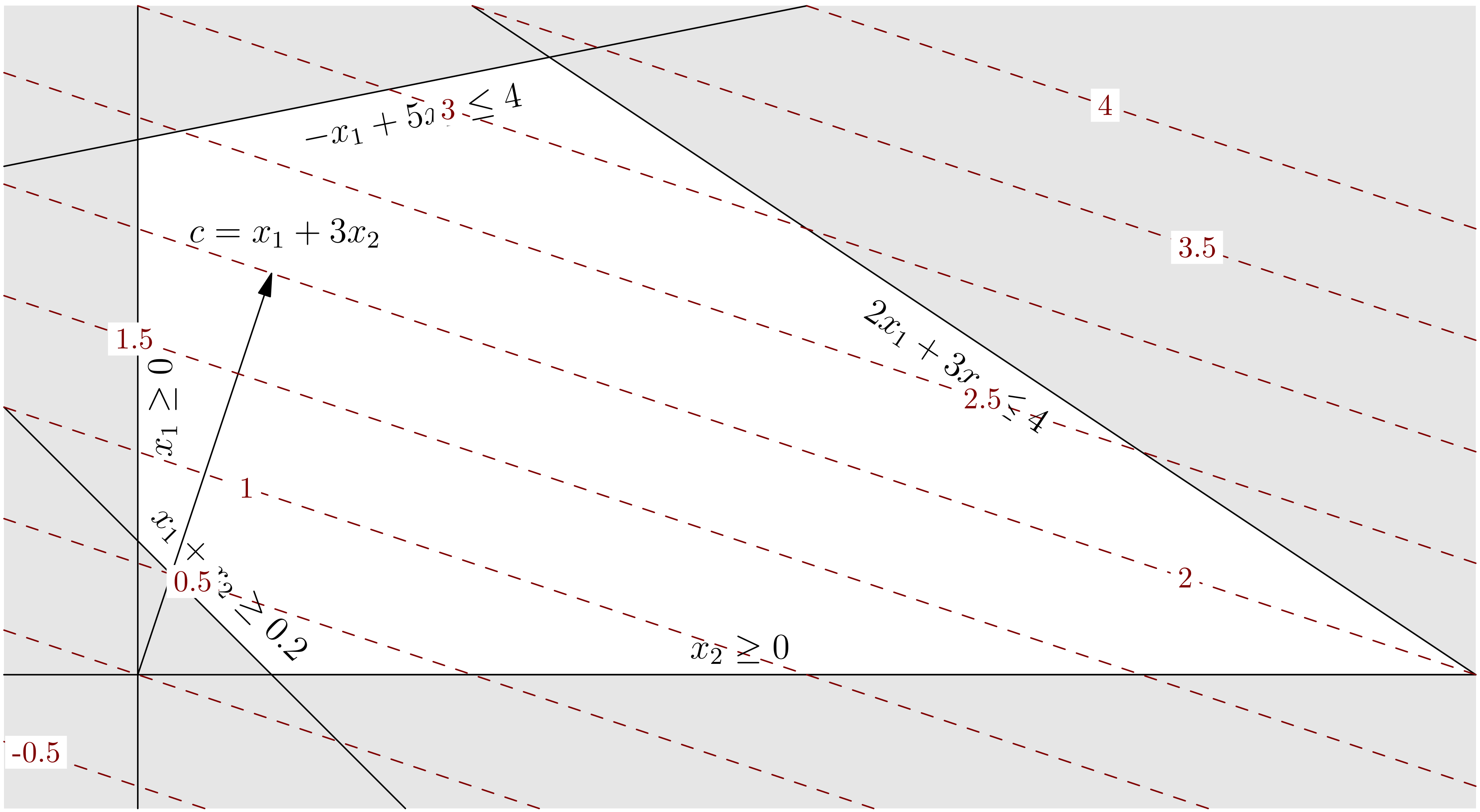
The Space of Feasible Solutions



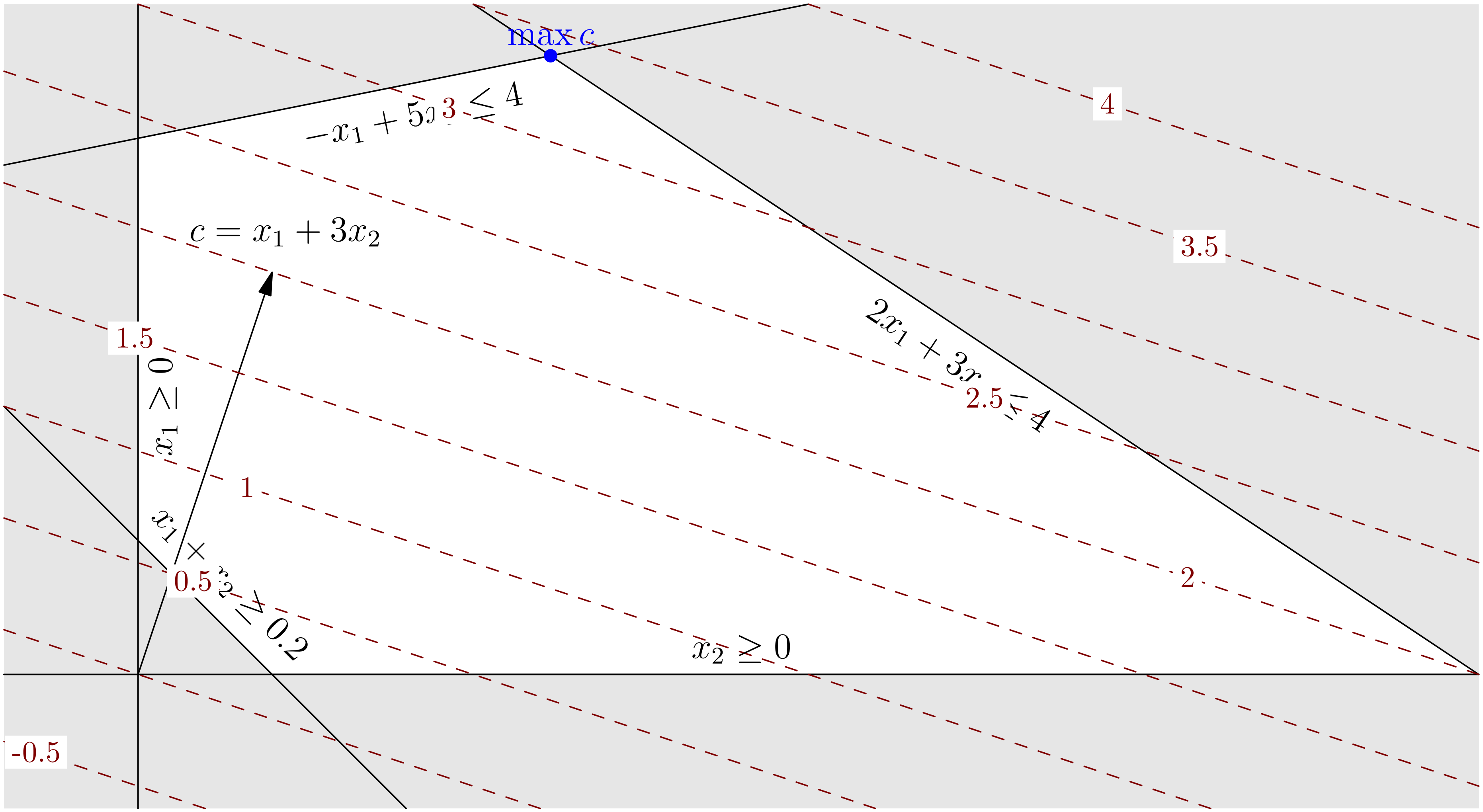
The Space of Feasible Solutions



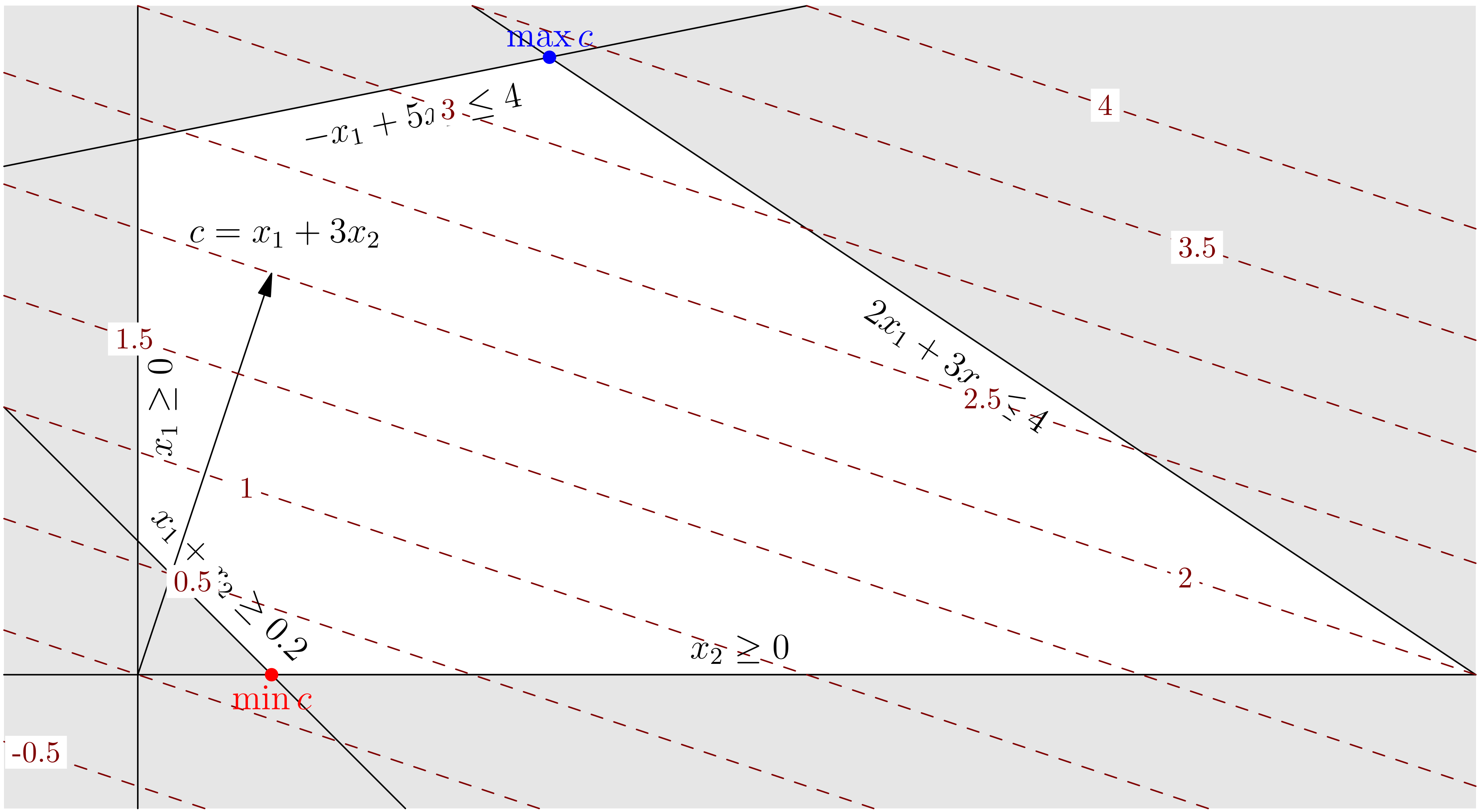
The Space of Feasible Solutions



The Space of Feasible Solutions



The Space of Feasible Solutions



Vertices of Polytope

- The space of feasible solutions is a polyhedra or polytope
- The maximum or minimum solution will always lie at a vertex of the polytope
- Our solution policy will be to start at a vertex and move to a neighbouring vertex that gives the best improvement in cost
- When this isn't possible then we are finished
- However, there is still a lot of work to realise this solution strategy

Vertices of Polytope

- The space of feasible solutions is a polyhedra or polytope
- The maximum or minimum solution will always lie at a vertex of the polytope
- Our solution policy will be to start at a vertex and move to a neighbouring vertex that gives the best improvement in cost
- When this isn't possible then we are finished
- However, there is still a lot of work to realise this solution strategy

Vertices of Polytope

- The space of feasible solutions is a polyhedra or polytope
- The maximum or minimum solution will always lie at a vertex of the polytope
- Our solution policy will be to start at a vertex and move to a neighbouring vertex that gives the best improvement in cost
- When this isn't possible then we are finished
- However, there is still a lot of work to realise this solution strategy

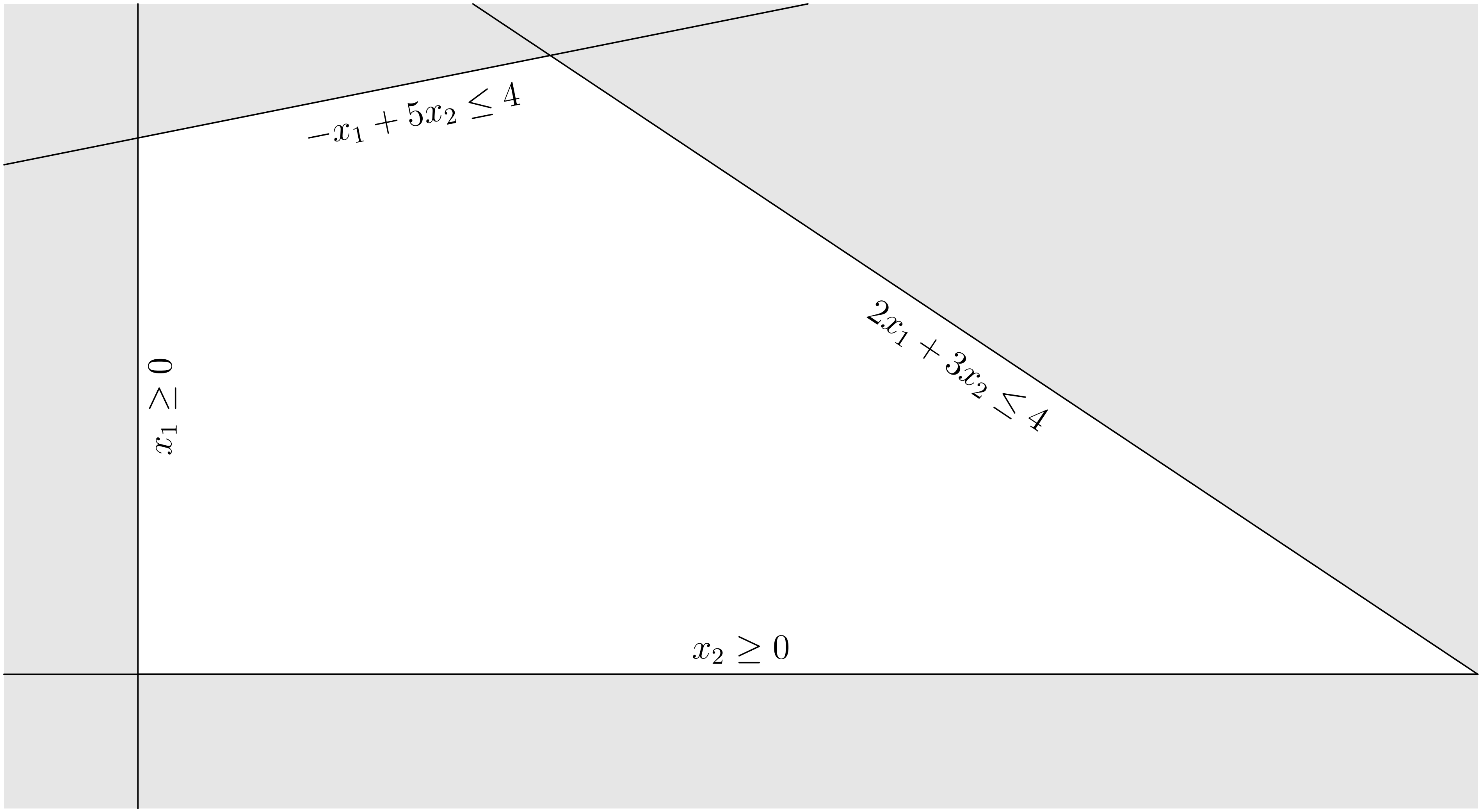
Vertices of Polytope

- The space of feasible solutions is a polyhedra or polytope
- The maximum or minimum solution will always lie at a vertex of the polytope
- Our solution policy will be to start at a vertex and move to a neighbouring vertex that gives the best improvement in cost
- When this isn't possible then we are finished
- However, there is still a lot of work to realise this solution strategy

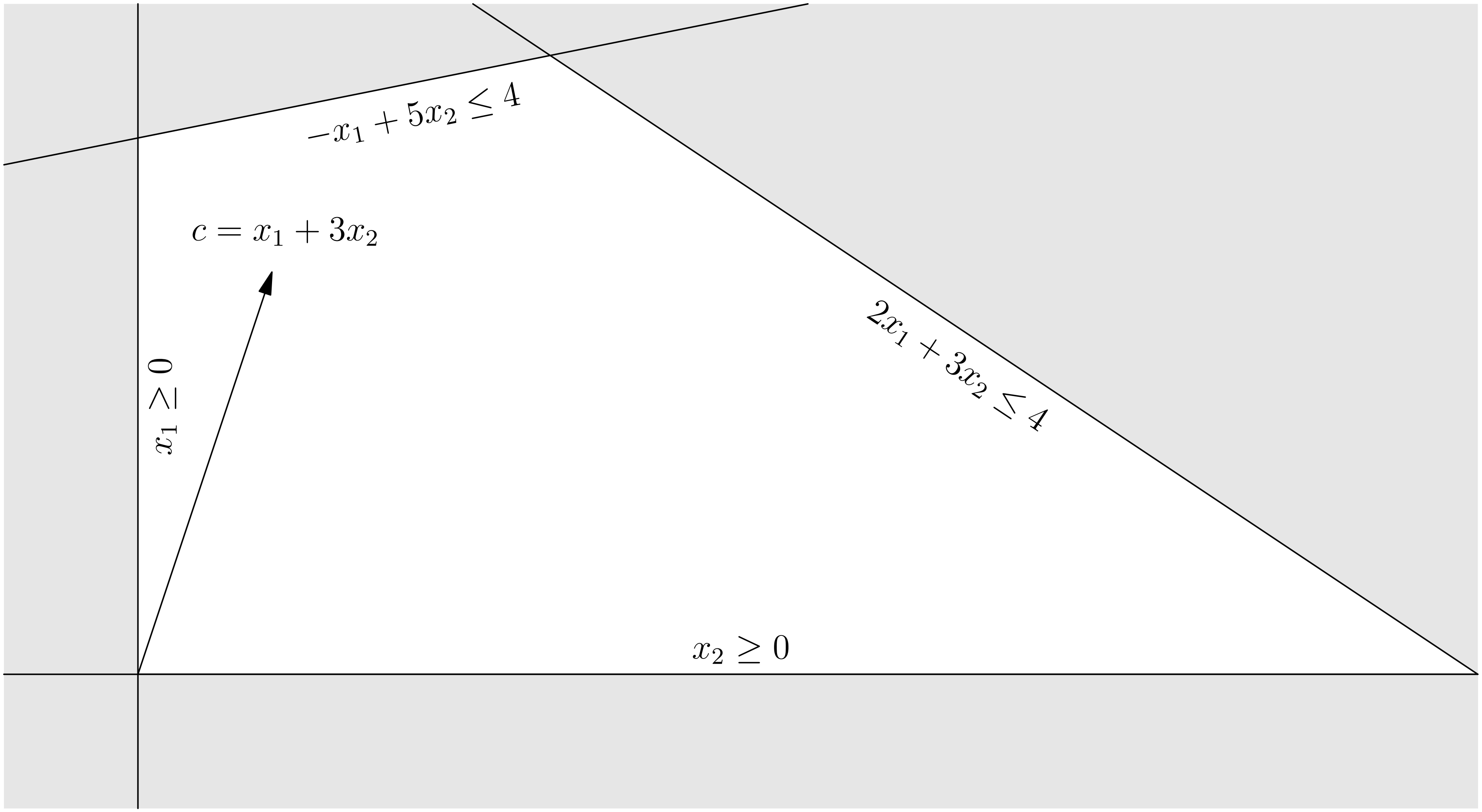
Vertices of Polytope

- The space of feasible solutions is a polyhedra or polytope
- The maximum or minimum solution will always lie at a vertex of the polytope
- Our solution policy will be to start at a vertex and move to a neighbouring vertex that gives the best improvement in cost
- When this isn't possible then we are finished
- However, there is still a lot of work to realise this solution strategy

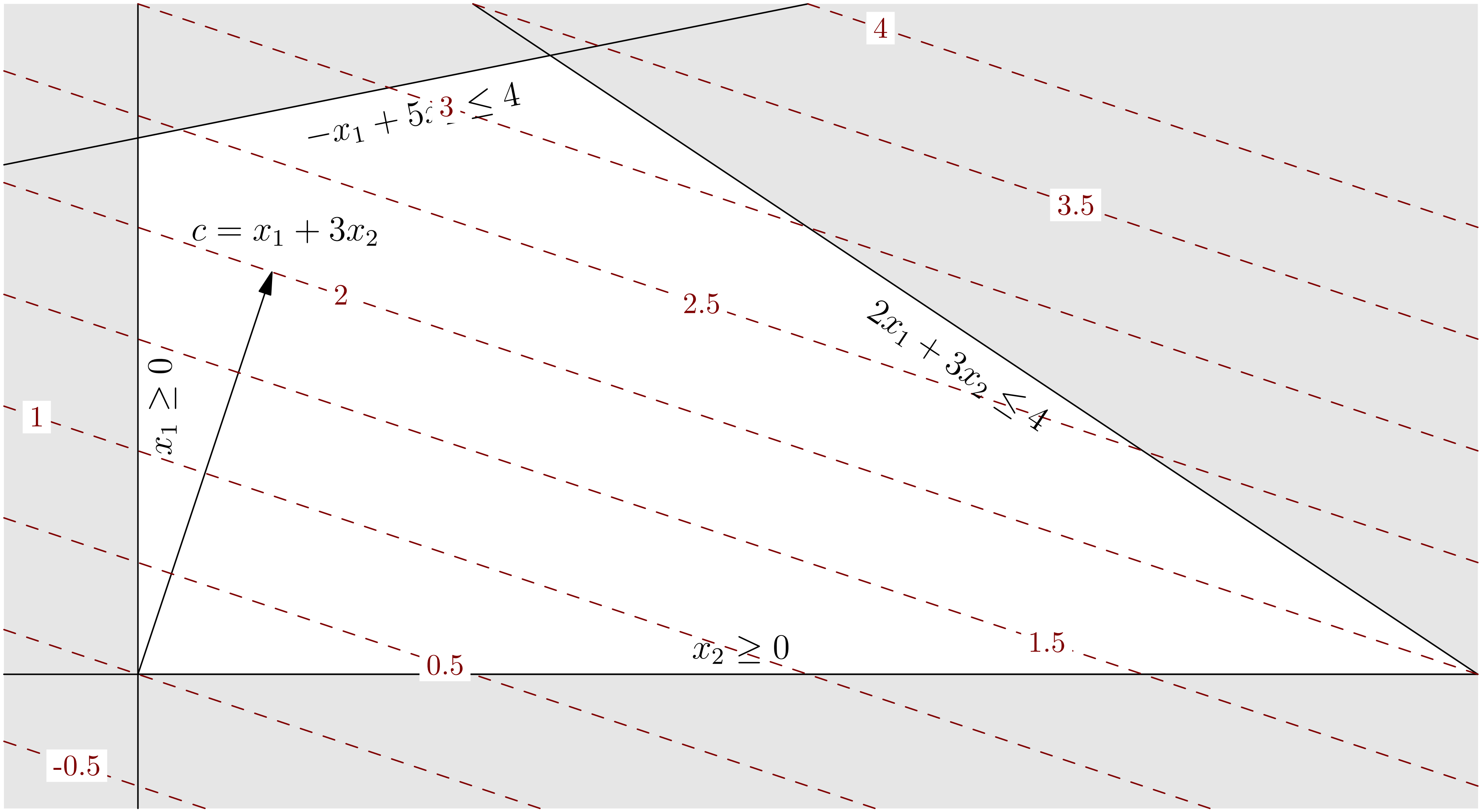
Optimal Solution



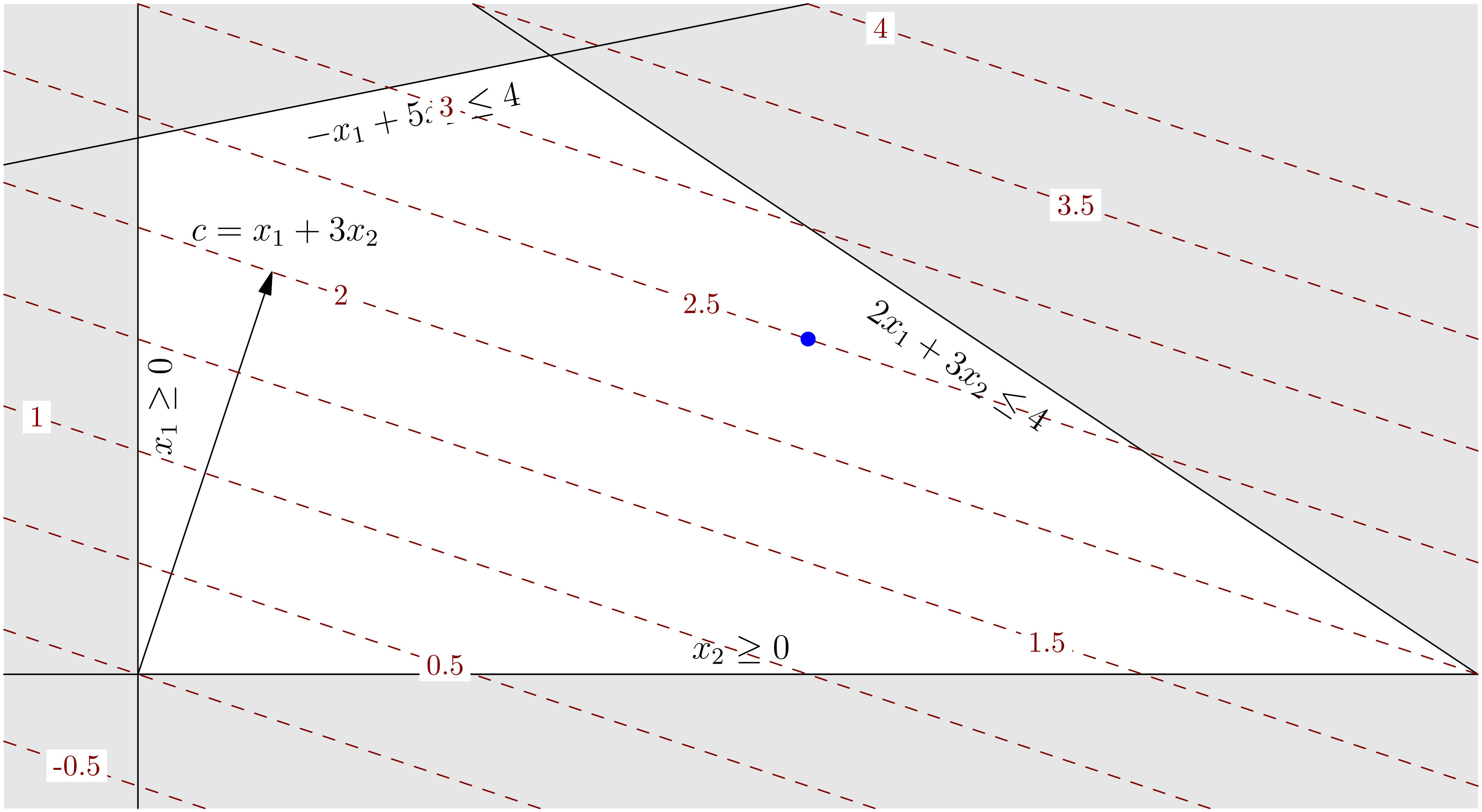
Optimal Solution



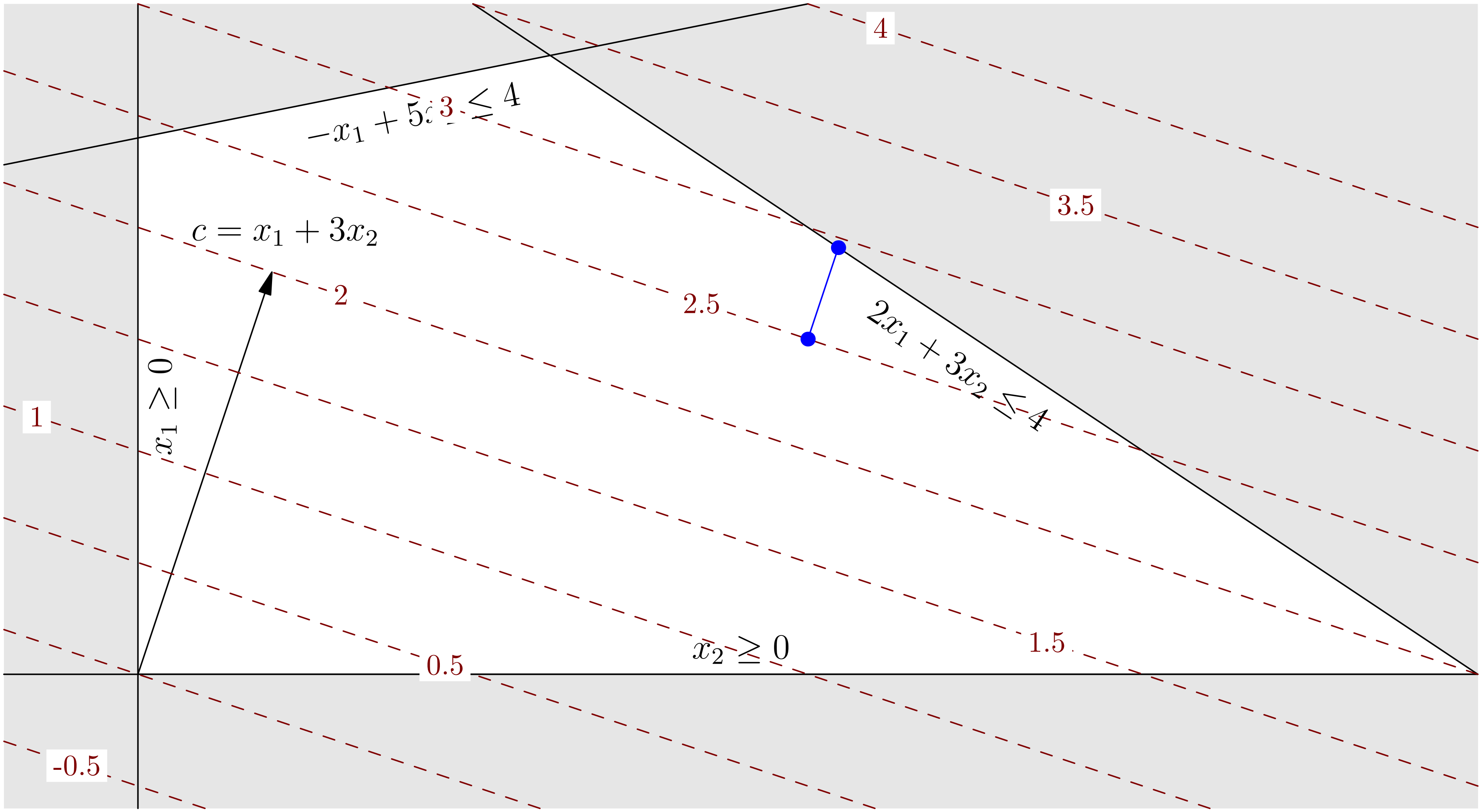
Optimal Solution



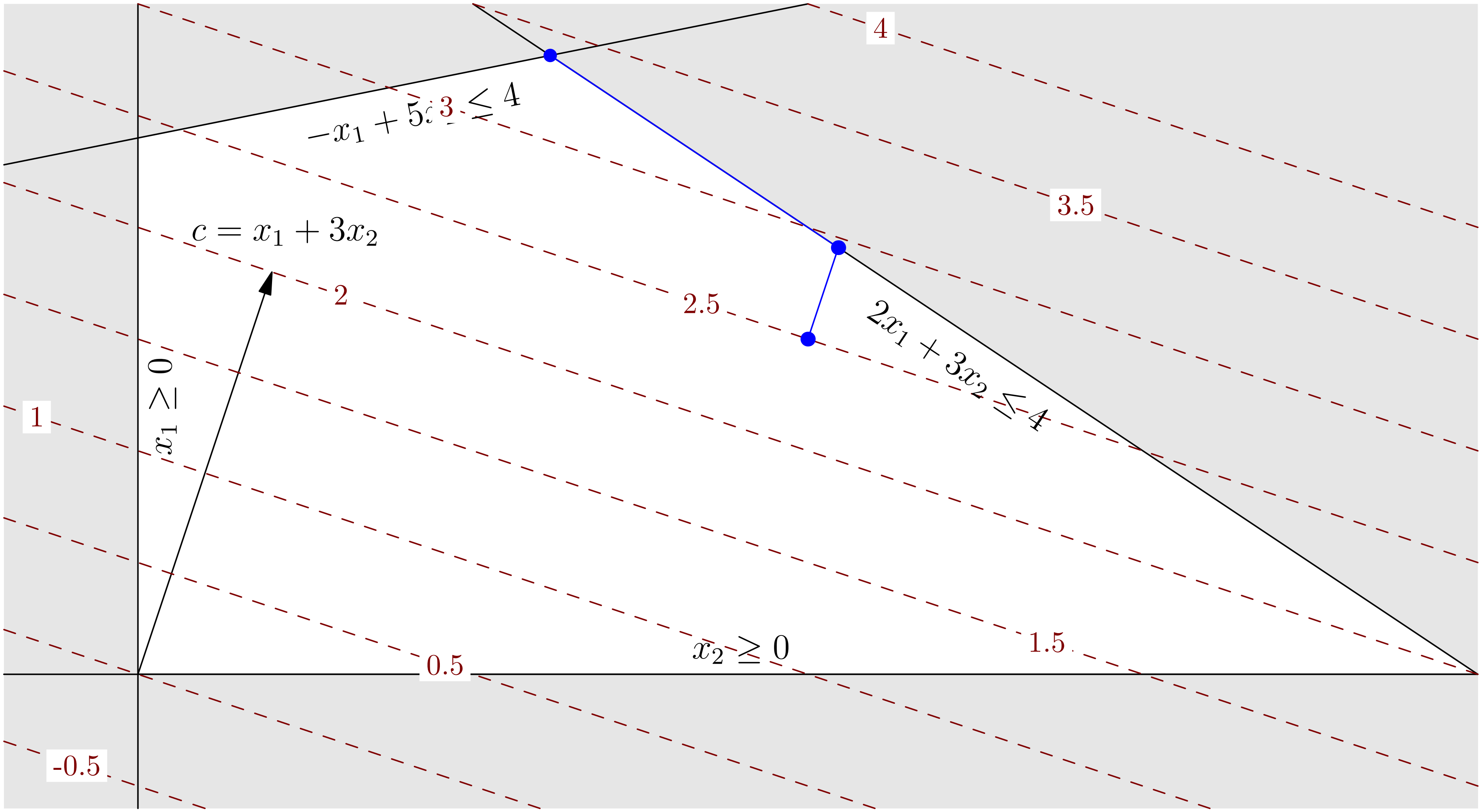
Optimal Solution



Optimal Solution

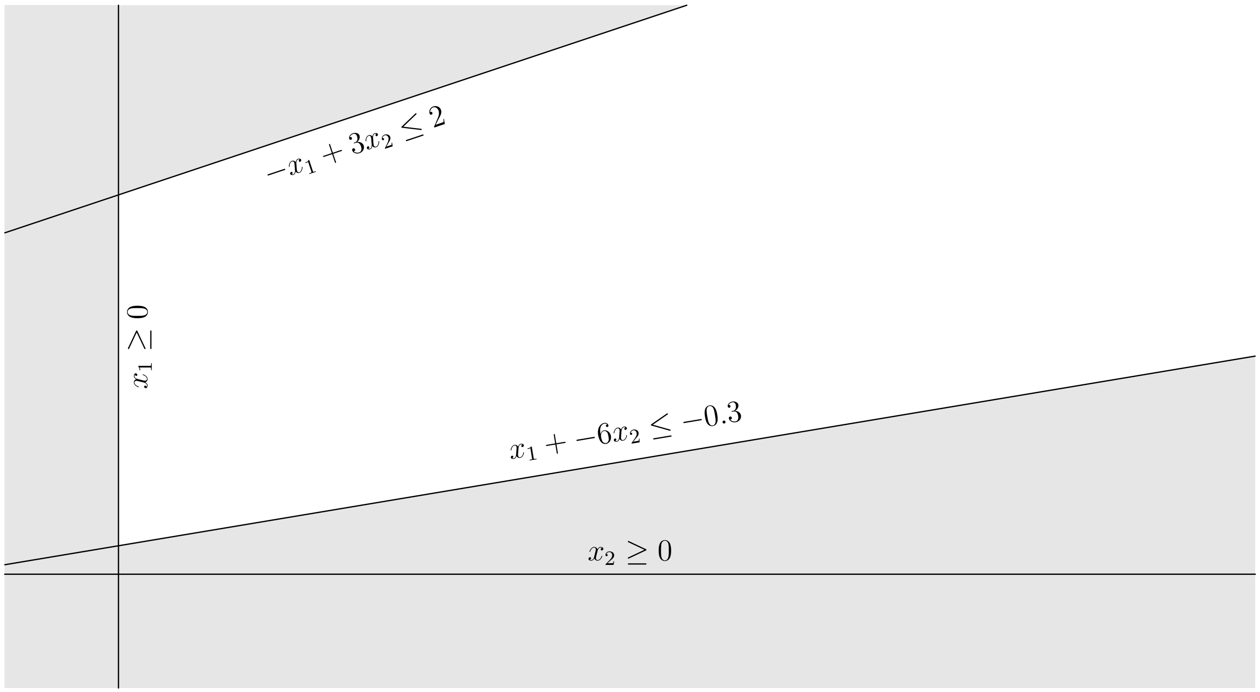


Optimal Solution



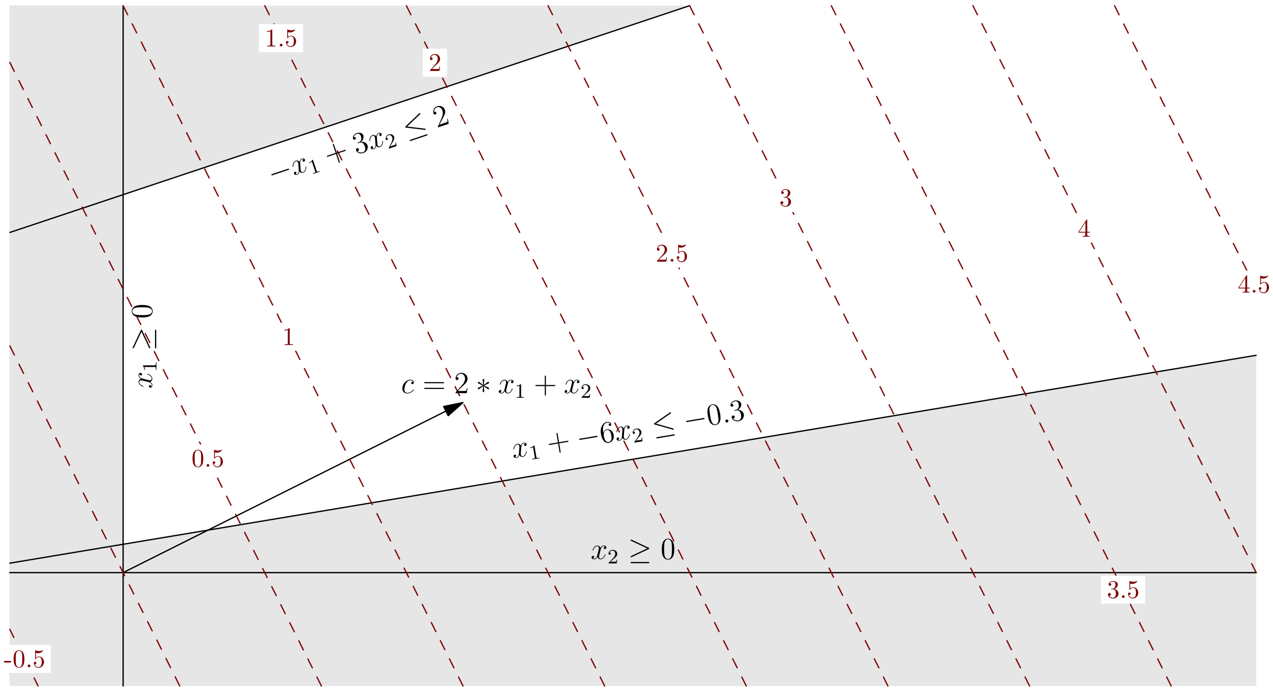
Unbounded Solutions

- If you are unlucky you might not have a bounded solution



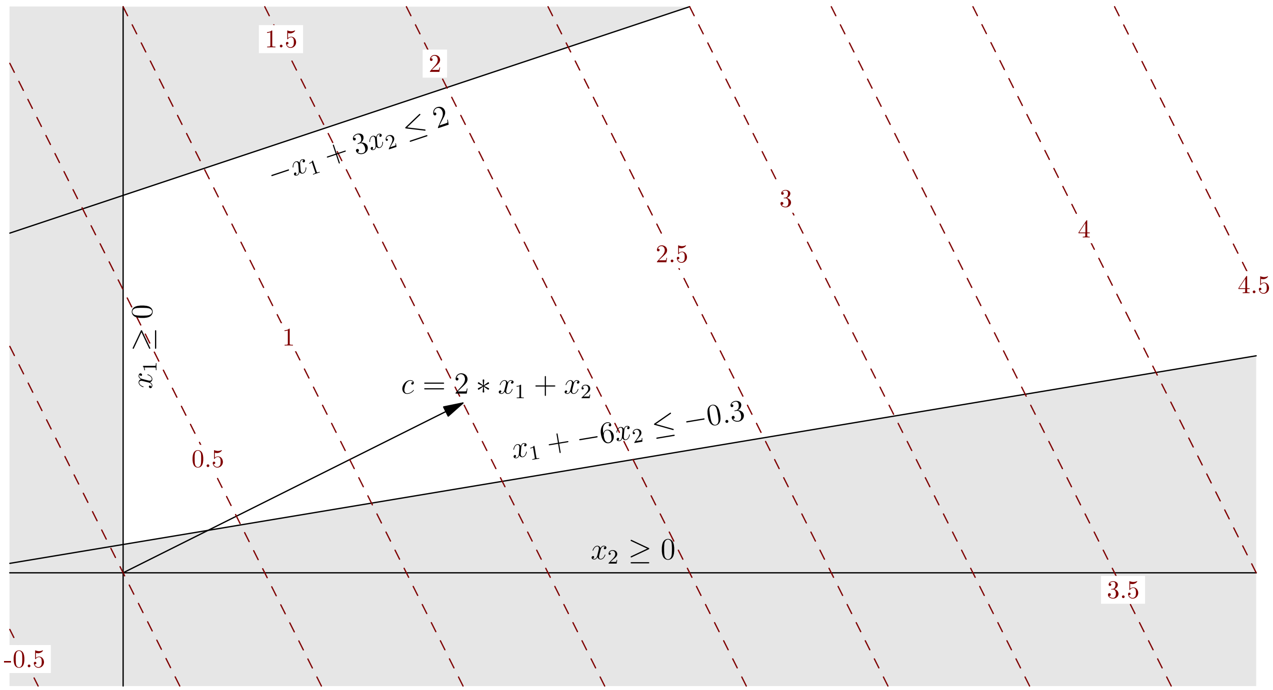
Unbounded Solutions

- If you are unlucky you might not have a bounded solution



Unbounded Solutions

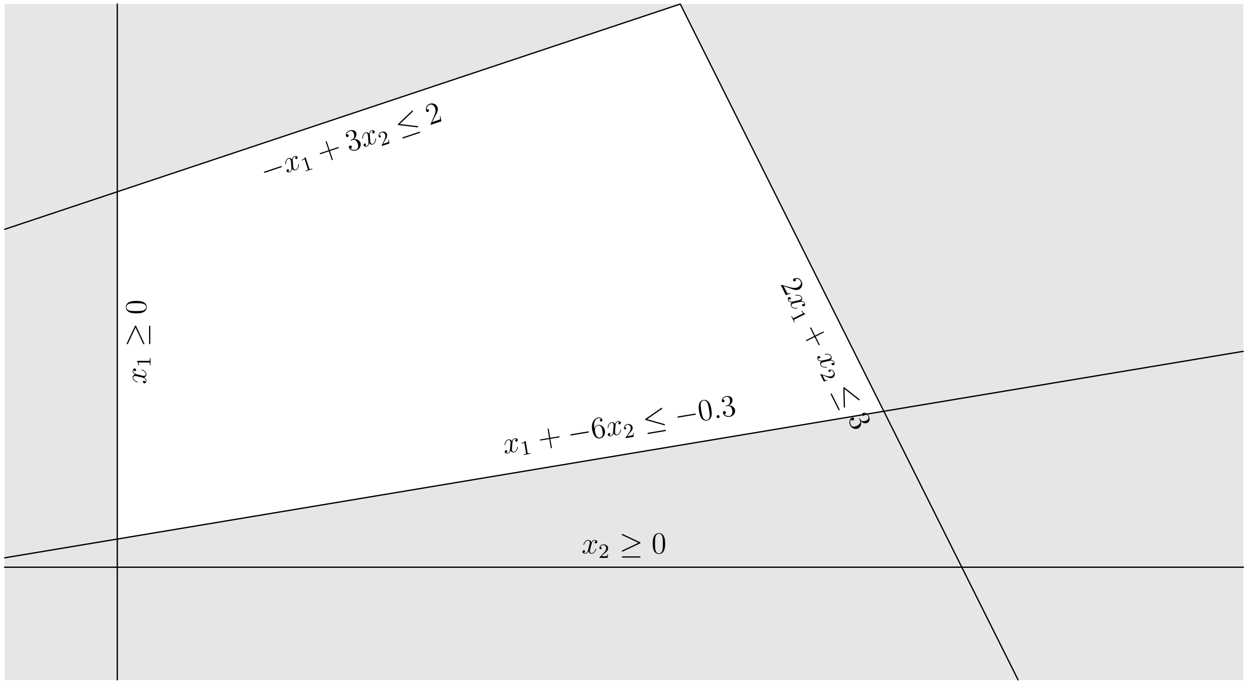
- If you are unlucky you might not have a bounded solution



- But usually this would not happen because of the problem definition

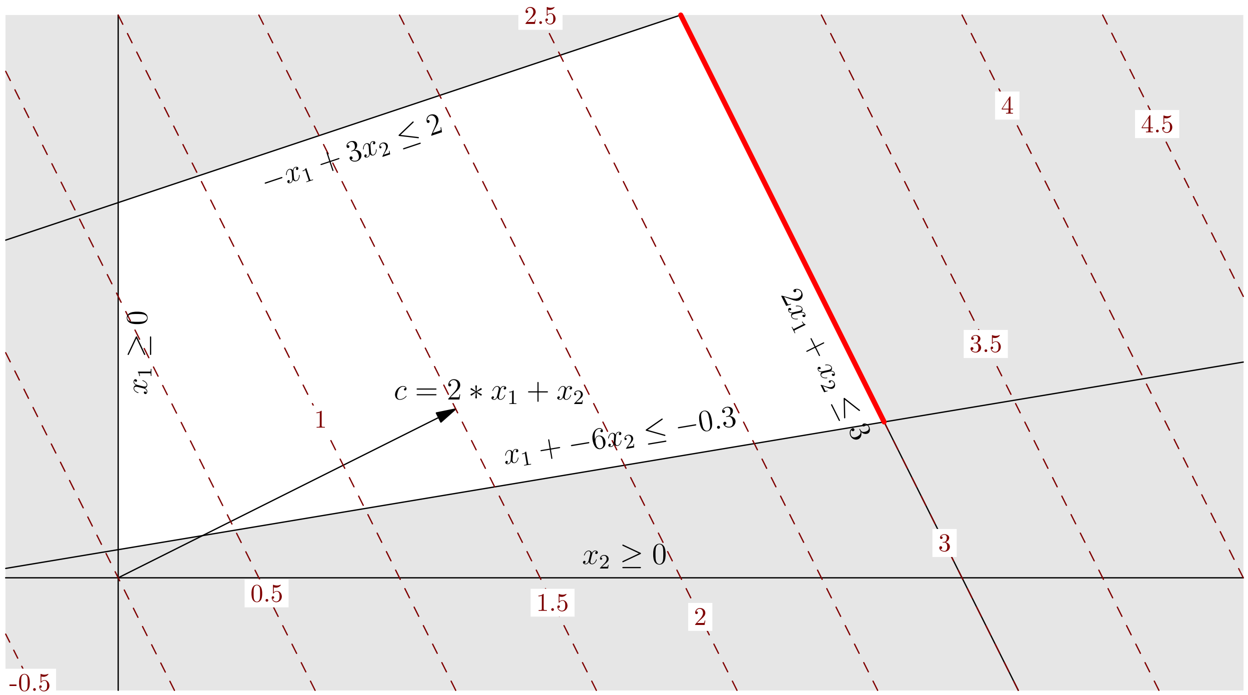
Multiple Solutions

- You can also get multiple solutions if a constraint is orthogonal to the objective function



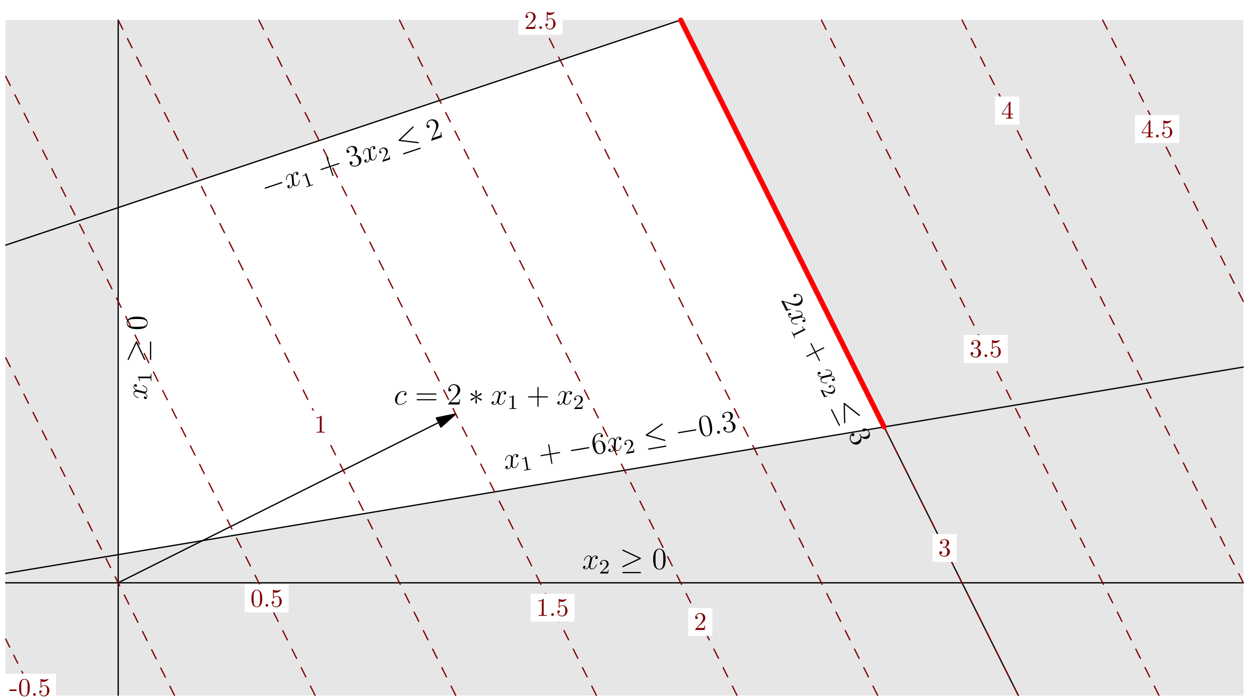
Multiple Solutions

- You can also get multiple solutions if a constraint is orthogonal to the objective function



Multiple Solutions

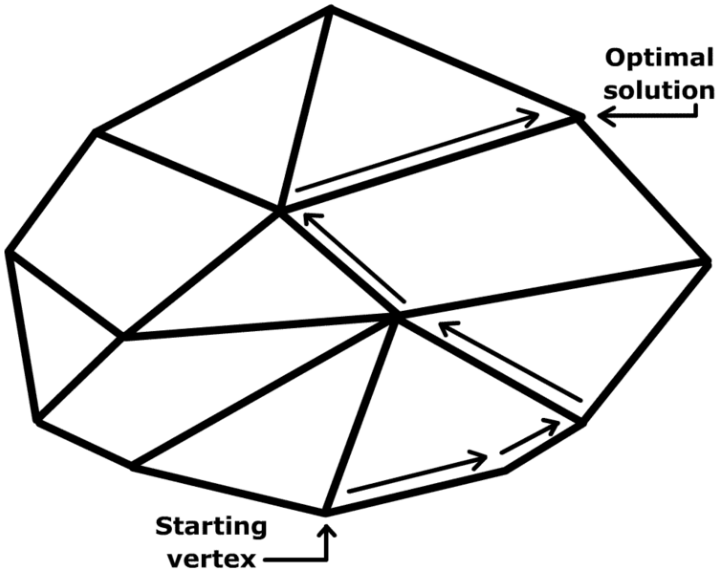
- You can also get multiple solutions if a constraint is orthogonal to the objective function



- Nevertheless the optimal will be at a vertex

Outline

1. Examples
2. Linear Programs
3. Properties of Solution
4. **Normal Form**



Converting Linear Programs

- Solving full linear programs is difficult
- However, it is much easier to solve linear programs in **normal form**
- This is basically a form where we get rid of all inequalities and rewriting the equalities
- Fortunately its rather easy to convert linear programs to normal form

Converting Linear Programs

- Solving full linear programs is difficult
- However, it is much easier to solve linear programs in **normal form**
- This is basically a form where we get rid of all inequalities and rewriting the equalities
- Fortunately its rather easy to convert linear programs to normal form

Converting Linear Programs

- Solving full linear programs is difficult
- However, it is much easier to solve linear programs in **normal form**
- This is basically a form where we get rid of all inequalities and rewriting the equalities
- Fortunately its rather easy to convert linear programs to normal form

Converting Linear Programs

- Solving full linear programs is difficult
- However, it is much easier to solve linear programs in **normal form**
- This is basically a form where we get rid of all inequalities and rewriting the equalities
- Fortunately its rather easy to convert linear programs to normal form

Slack Variables

- We can change an inequality into an equality by introducing a new "**slack**" variable
- E.g.

$$
\begin{array}{lcll}
\boldsymbol{a}_1 \cdot \boldsymbol{x} \geq 0 & \Rightarrow & \boldsymbol{a}_1 \cdot \boldsymbol{x} - z_1 = 0 & z_1 \geq 0 \\
\boldsymbol{a}_2 \cdot \boldsymbol{x} \leq 0 & \Rightarrow & \boldsymbol{a}_2 \cdot \boldsymbol{x} + z_2 = 0 & z_2 \geq 0
\end{array}
$$

  z_1 (the excess) and z_2 (the deficit) are known as slack variables

- We eliminate inequalities at the expense of increasing the number of variables
- We can treat the slack variables on an equivalent footing to the normal variables (they just provide a different way of describing the original problem)

Slack Variables

- We can change an inequality into an equality by introducing a new "**slack**" variable

- E.g.

$$\begin{aligned} \boldsymbol{a}_1 \cdot \boldsymbol{x} \geq 0 \qquad &\Rightarrow \qquad \boldsymbol{a}_1 \cdot \boldsymbol{x} - z_1 = 0 \quad z_1 \geq 0 \\ \boldsymbol{a}_2 \cdot \boldsymbol{x} \leq 0 \qquad &\Rightarrow \qquad \boldsymbol{a}_2 \cdot \boldsymbol{x} + z_2 = 0 \quad z_2 \geq 0 \end{aligned}$$

  z_1 (the excess) and z_2 (the deficit) are known as slack variables

- We eliminate inequalities at the expense of increasing the number of variables

- We can treat the slack variables on an equivalent footing to the normal variables (they just provide a different way of describing the original problem)

Slack Variables

- We can change an inequality into an equality by introducing a new "**slack**" variable

- E.g.

$$\begin{array}{lcll} \boldsymbol{a}_1 \cdot \boldsymbol{x} \geq 0 & \Rightarrow & \boldsymbol{a}_1 \cdot \boldsymbol{x} - z_1 = 0 & z_1 \geq 0 \\ \boldsymbol{a}_2 \cdot \boldsymbol{x} \leq 0 & \Rightarrow & \boldsymbol{a}_2 \cdot \boldsymbol{x} + z_2 = 0 & z_2 \geq 0 \end{array}$$

  z_1 (the excess) and z_2 (the deficit) are known as slack variables

- We eliminate inequalities at the expense of increasing the number of variables

- We can treat the slack variables on an equivalent footing to the normal variables (they just provide a different way of describing the original problem)

Slack Variables

- We can change an inequality into an equality by introducing a new "**slack**" variable

- E.g.

$$\begin{array}{lcll} \boldsymbol{a}_1 \cdot \boldsymbol{x} \geq 0 & \Rightarrow & \boldsymbol{a}_1 \cdot \boldsymbol{x} - z_1 = 0 & z_1 \geq 0 \\ \boldsymbol{a}_2 \cdot \boldsymbol{x} \leq 0 & \Rightarrow & \boldsymbol{a}_2 \cdot \boldsymbol{x} + z_2 = 0 & z_2 \geq 0 \end{array}$$

  z_1 (the excess) and z_2 (the deficit) are known as slack variables

- We eliminate inequalities at the expense of increasing the number of variables

- We can treat the slack variables on an equivalent footing to the normal variables (they just provide a different way of describing the original problem)

Normal Form

- A linear program with only equality constraints is said to be in **normal form**
- We will find in the next lecture that this is a convenient form for solving linear programs
- An equality constraint restricts the solutions to a subspace (some lower dimensional space)

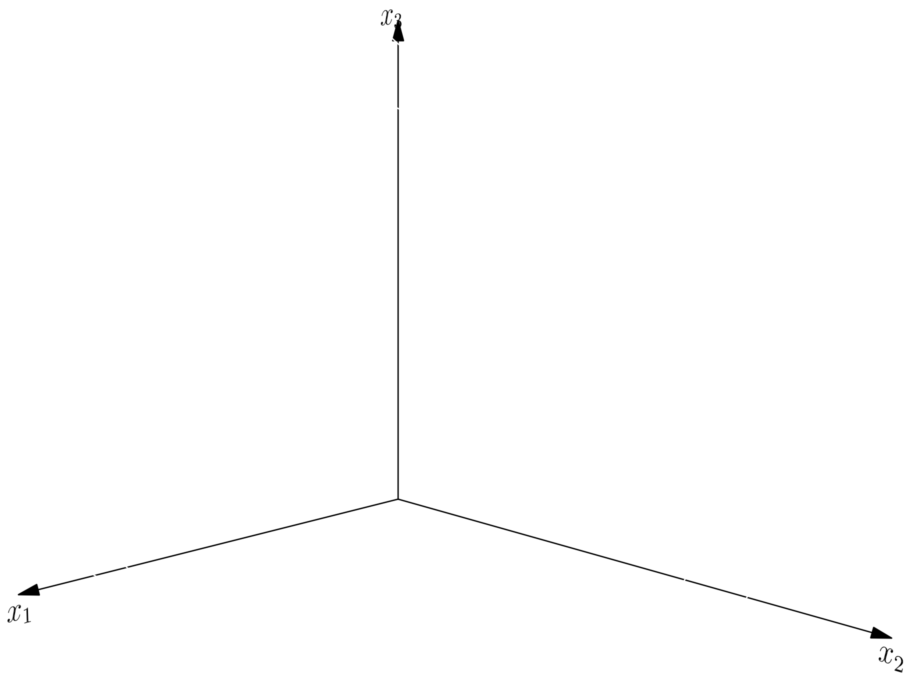
Normal Form

- A linear program with only equality constraints is said to be in **normal form**
- We will find in the next lecture that this is a convenient form for solving linear programs
- An equality constraint restricts the solutions to a subspace (some lower dimensional space)

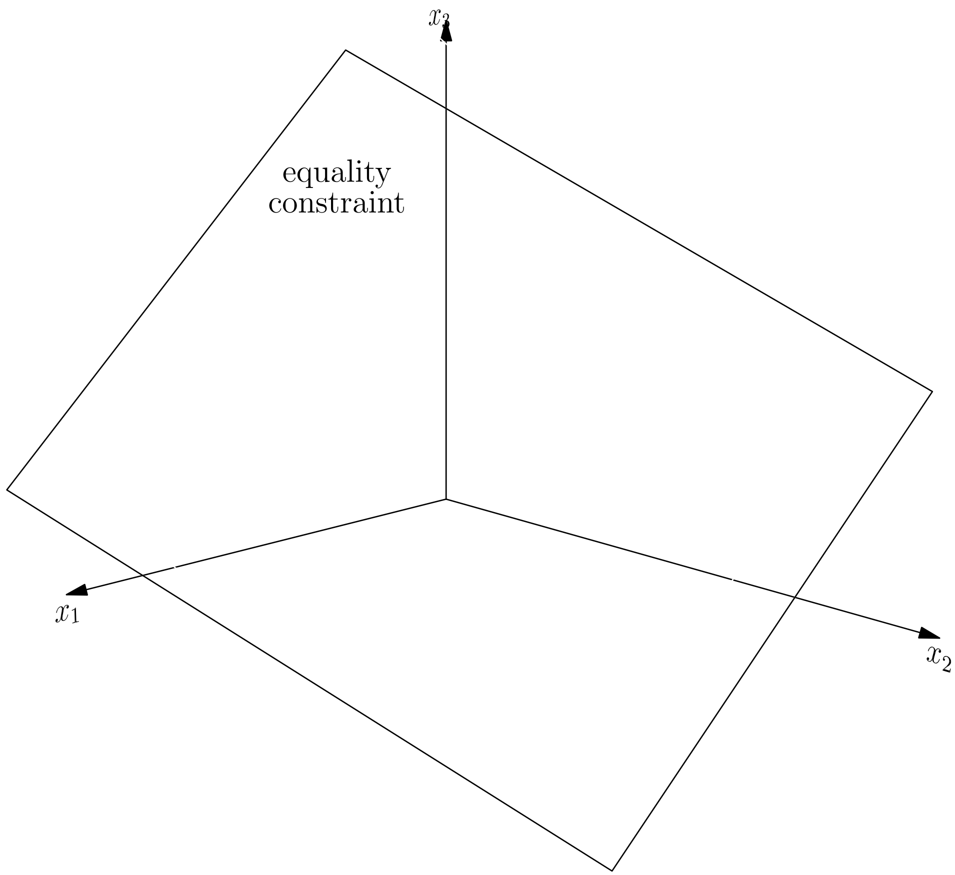
Normal Form

- A linear program with only equality constraints is said to be in **normal form**
- We will find in the next lecture that this is a convenient form for solving linear programs
- An equality constraint restricts the solutions to a subspace (some lower dimensional space)

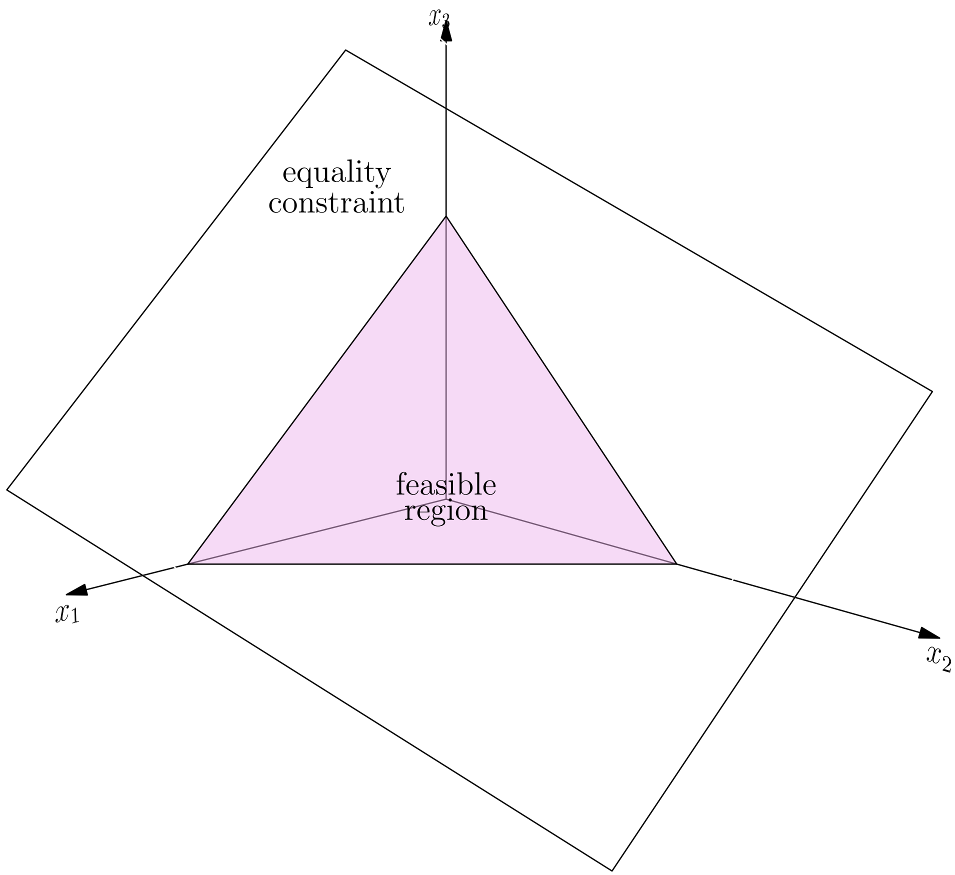
Solving Linear Programming



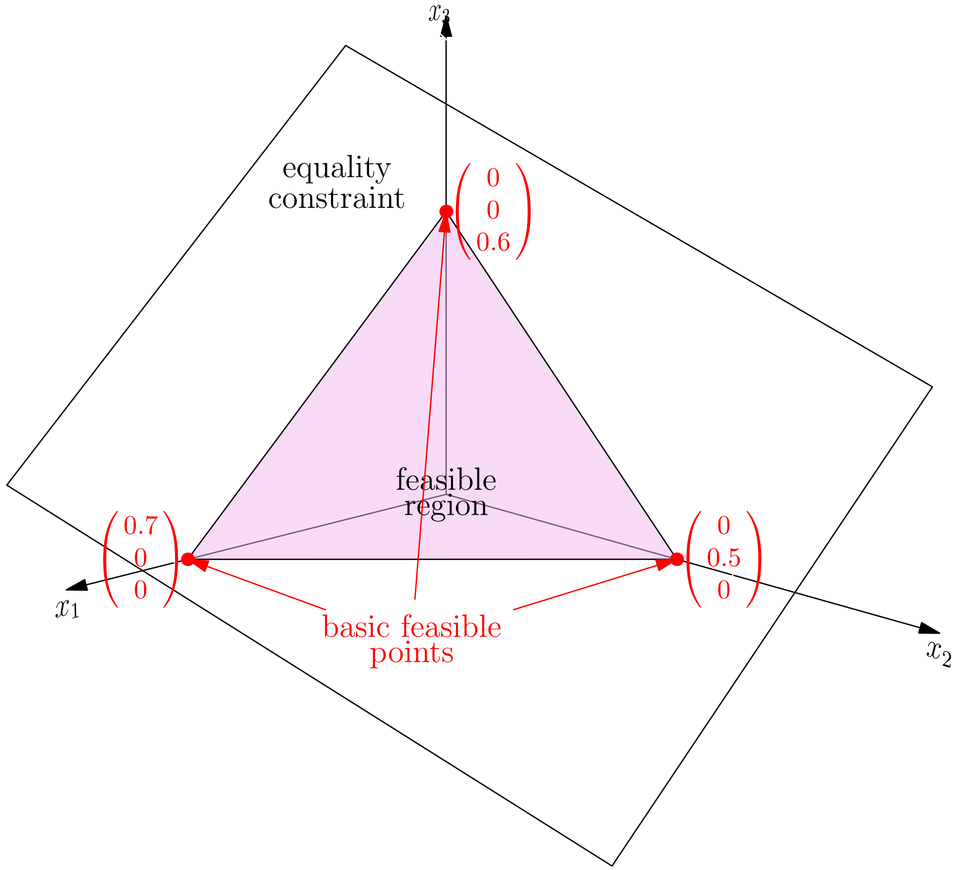
Solving Linear Programming



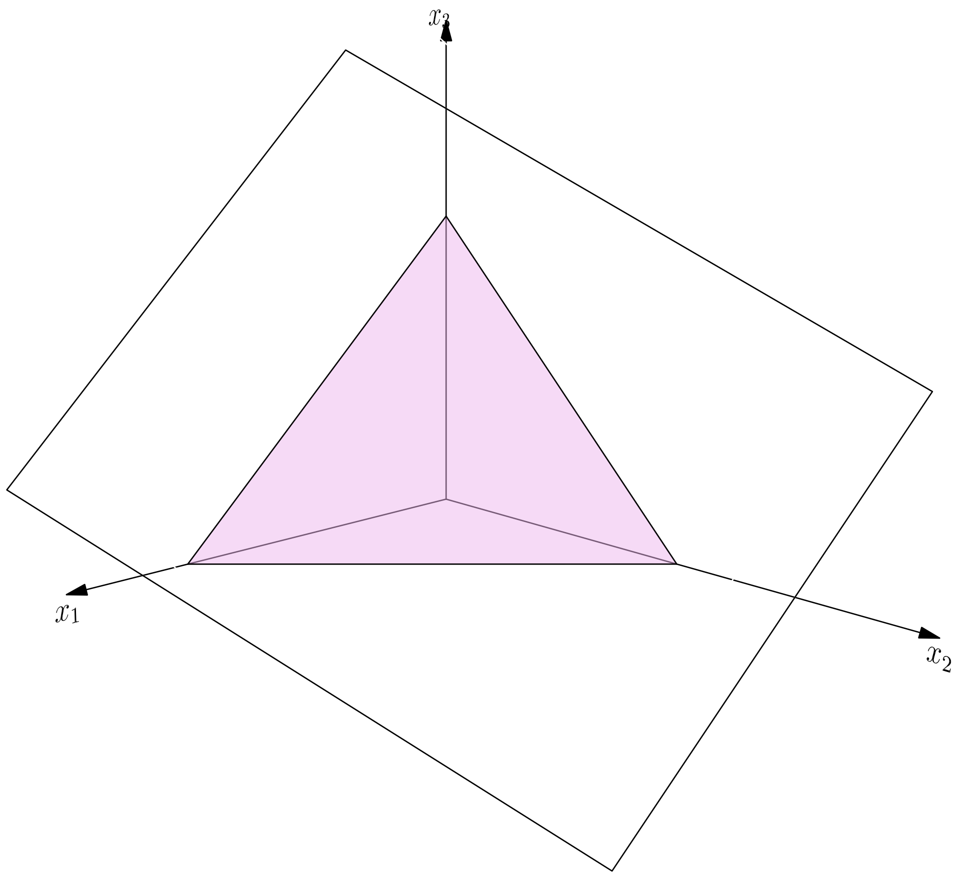
Solving Linear Programming



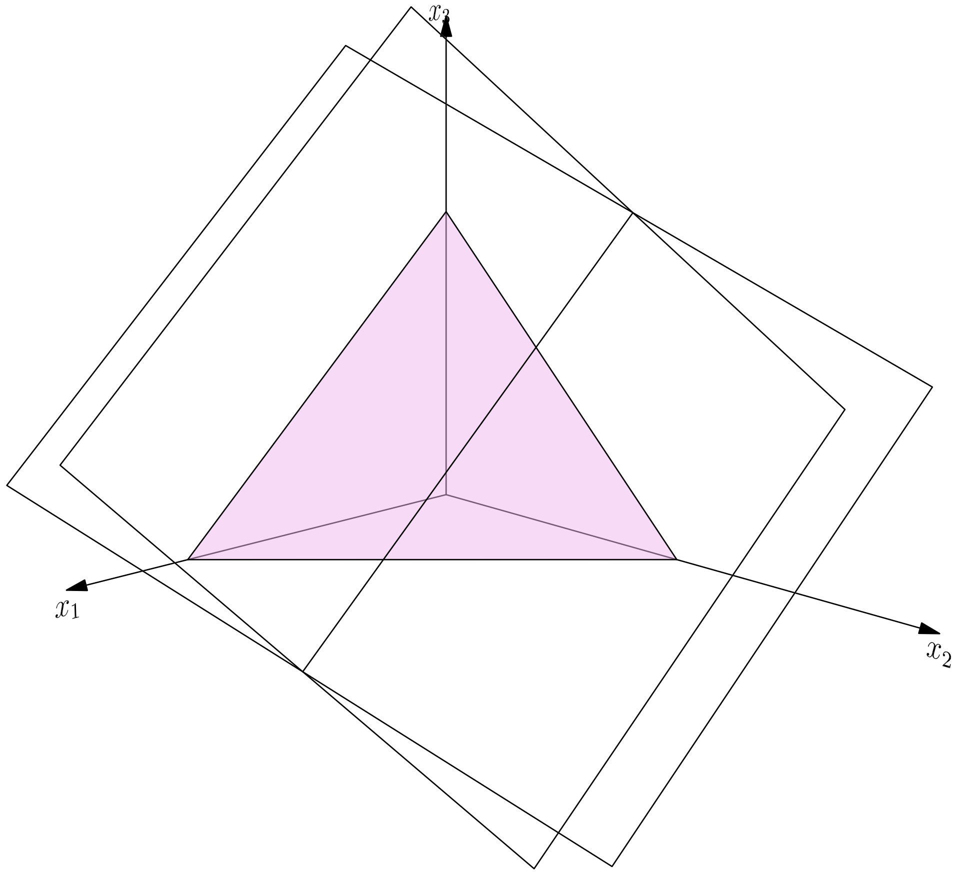
Solving Linear Programming



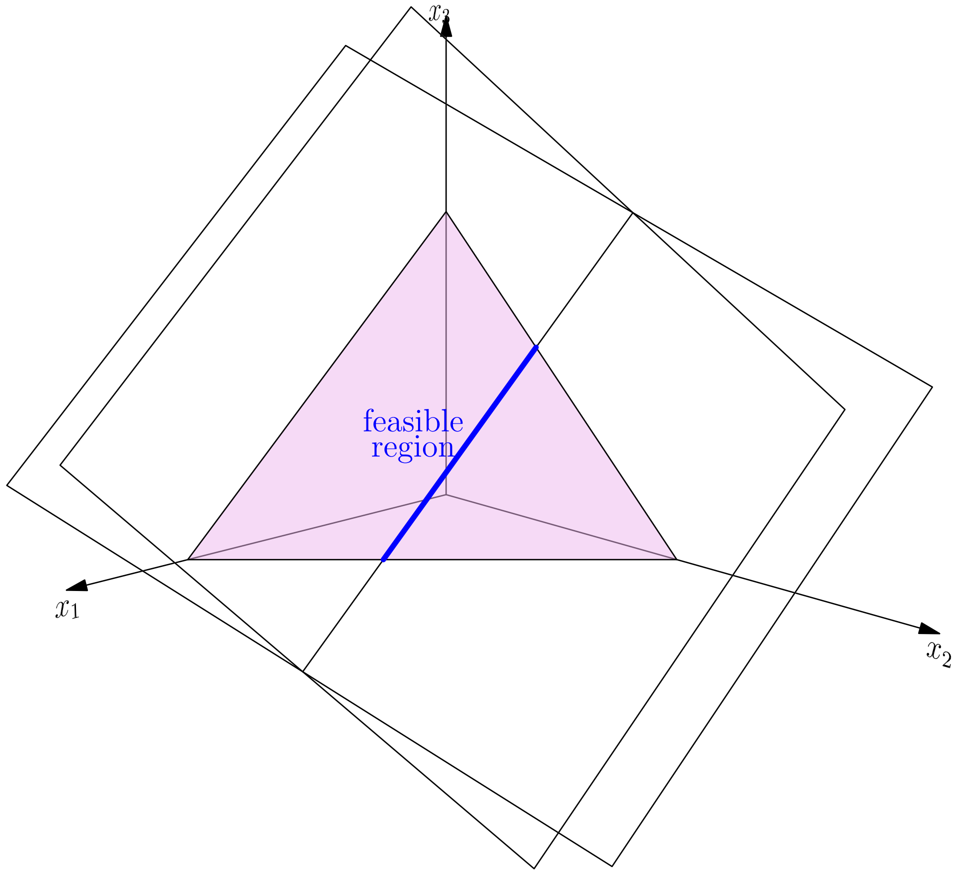
Solving Linear Programming



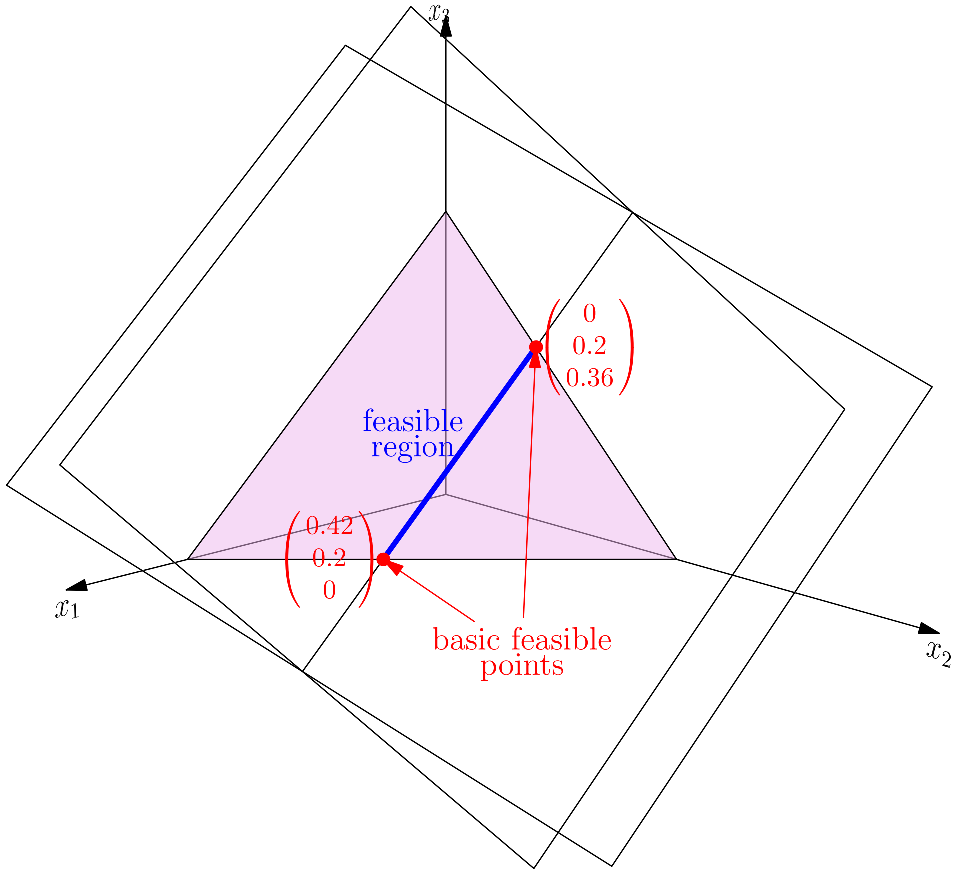
Solving Linear Programming



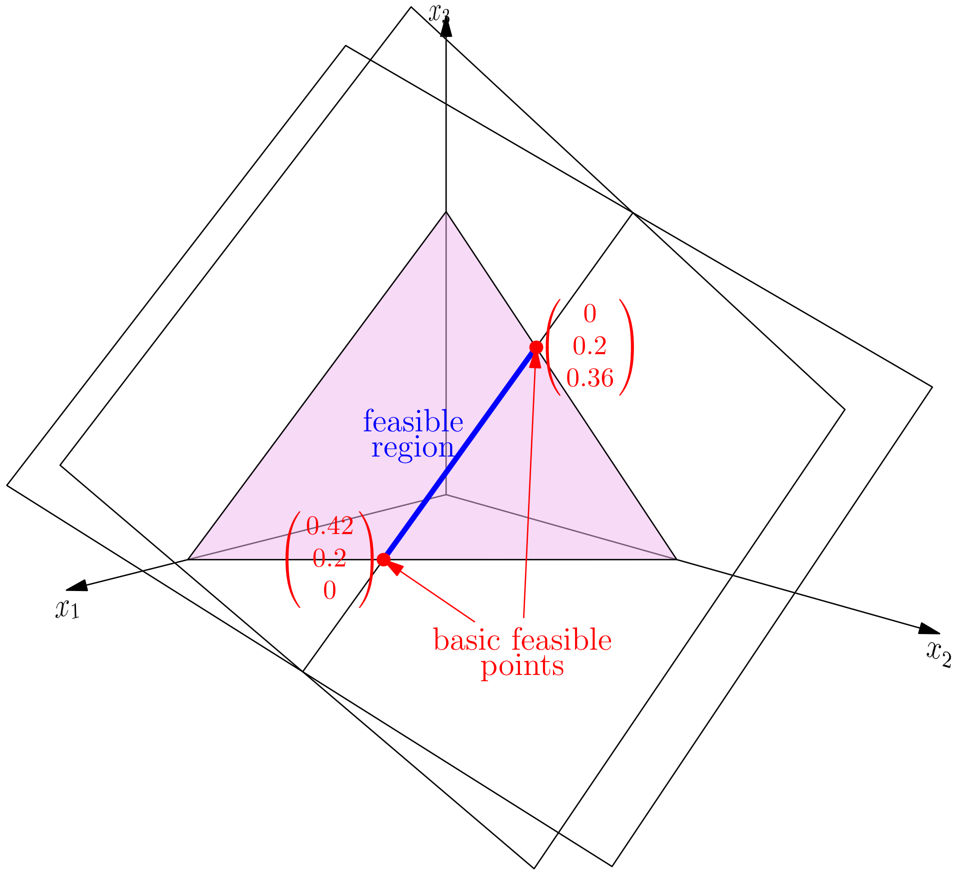
Solving Linear Programming



Solving Linear Programming

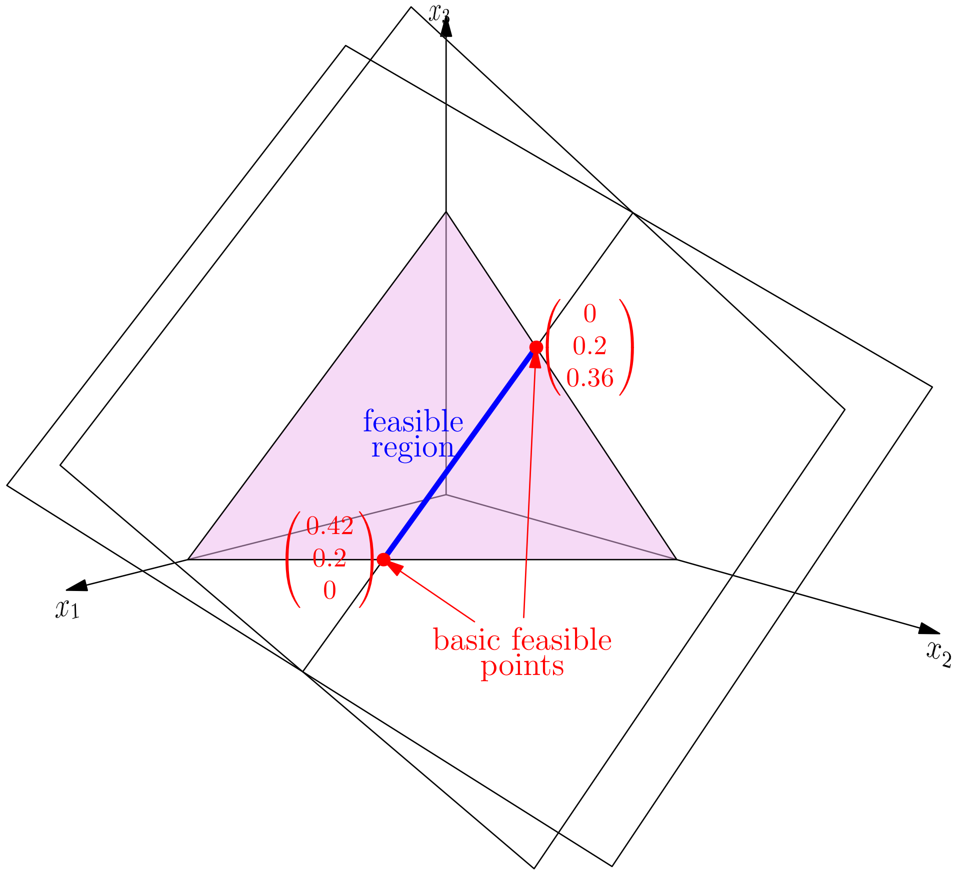


Solving Linear Programming



- The basic feasible points for LP problems with n variables and m constraints have at least $n - m$ zero variables

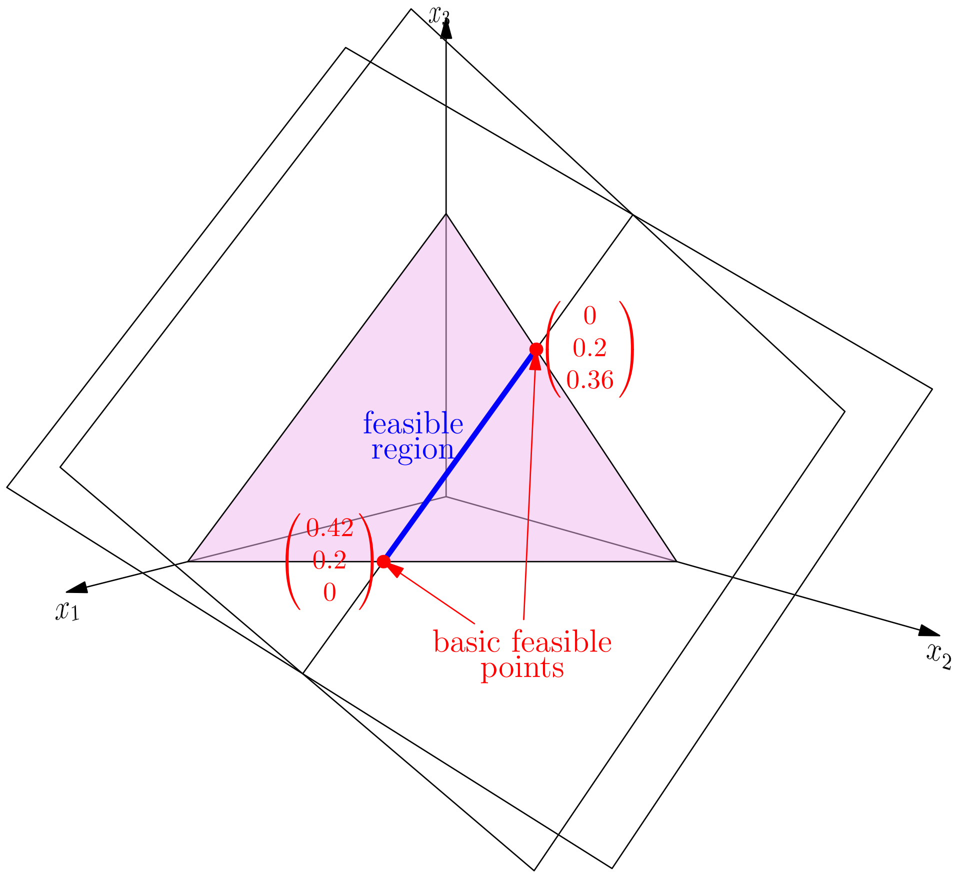
Solving Linear Programming



- The basic feasible points for LP problems with n variables and m constraints have at least $n-m$ zero variables

- Typical number of basic feasible solutions is $\binom{n}{m} \geq \left(\frac{n}{m}\right)^m$

Solving Linear Programming



- The basic feasible points for LP problems with n variables and m constraints have at least $n-m$ zero variables
- Typical number of basic feasible solutions is $\binom{n}{m} \geq \left(\frac{n}{m}\right)^m$
- Simplex algorithm organises iterative search for global solutions

Lessons

- There are a huge number of problems that can be set up as linear programs
- They are particularly useful in resource allocation where the resources are all positive
- The solution to linear programming problems is at the vertex of the feasible space (intersection of constraints)
- We can search for solutions by moving from vertex to vertex
- We can transform inequality constraints to equality constraints using slack variables

Lessons

- There are a huge number of problems that can be set up as linear programs
- They are particularly useful in resource allocation where the resources are all positive
- The solution to linear programming problems is at the vertex of the feasible space (intersection of constraints)
- We can search for solutions by moving from vertex to vertex
- We can transform inequality constraints to equality constraints using slack variables

Lessons

- There are a huge number of problems that can be set up as linear programs
- They are particularly useful in resource allocation where the resources are all positive
- The solution to linear programming problems is at the vertex of the feasible space (intersection of constraints)
- We can search for solutions by moving from vertex to vertex
- We can transform inequality constraints to equality constraints using slack variables

Lessons

- There are a huge number of problems that can be set up as linear programs
- They are particularly useful in resource allocation where the resources are all positive
- The solution to linear programming problems is at the vertex of the feasible space (intersection of constraints)
- We can search for solutions by moving from vertex to vertex
- We can transform inequality constraints to equality constraints using slack variables

Lessons

- There are a huge number of problems that can be set up as linear programs
- They are particularly useful in resource allocation where the resources are all positive
- The solution to linear programming problems is at the vertex of the feasible space (intersection of constraints)
- We can search for solutions by moving from vertex to vertex
- We can transform inequality constraints to equality constraints using slack variables