SFMin in an "Assemble to Order" inventory problem

S. Thomas McCormick
(with M. Bolandnazar, W.T. Huh, K. Murota)

Sauder School of Business, UBC
Cargese Workshop on Combinatorial Optimization,
Sept–Oct 2013

Outline

Supply Chain Questions

- A typical supply chain consists of one or more suppliers who manufacture components that are supplied to one or more manufacturers, who assemble the components into products, which are then sent maybe to retailers and/or end customers.

Supply Chain Questions

- A typical supply chain consists of one or more suppliers who manufacture components that are supplied to one or more manufacturers, who assemble the components into products, which are then sent maybe to retailers and/or end customers.
- Some basic questions are:

Supply Chain Questions

- A typical supply chain consists of one or more suppliers who manufacture components that are supplied to one or more manufacturers, who assemble the components into products, which are then sent maybe to retailers and/or end customers.
- Some basic questions are:
  1. When should, e.g., a manufacturer order?

Supply Chain Questions

- A typical supply chain consists of one or more suppliers who manufacture components that are supplied to one or more manufacturers, who assemble the components into products, which are then sent maybe to retailers and/or end customers.
- Some basic questions are:
  1. When should, e.g., a manufacturer order?
  2. How many units should they order?

Supply Chain Questions

- A typical supply chain consists of one or more suppliers who manufacture components that are supplied to one or more manufacturers, who assemble the components into products, which are then sent maybe to retailers and/or end customers.
- Some basic questions are:
  1. When should, e.g., a manufacturer order?
  2. How many units should they order?
  3. Can we say anything useful about the structure of an optimal policy?

Supply Chain Questions

- A typical supply chain consists of one or more suppliers who manufacture components that are supplied to one or more manufacturers, who assemble the components into products, which are then sent maybe to retailers and/or end customers.
- Some basic questions are:
  1. When should, e.g., a manufacturer order?
  2. How many units should they order?
  3. Can we say anything useful about the structure of an optimal policy?
  4. Can we say anything useful about the qualitative sensitivity of an optimal policy? E.g., if there is more stock of product A, does this mean that we should order more or less of product B?

Supply Chain Optimization is Hard

- Most of these questions can be posed as optimization problems. These optimization problems have several difficulties:

Supply Chain Optimization is Hard

- Most of these questions can be posed as optimization problems. These optimization problems have several difficulties:
    1. They are stochastic: Performance depends on customer demand, which is random. What sort of demands can we assume? Normal? Poisson? General?

Supply Chain Optimization is Hard

- Most of these questions can be posed as optimization problems. These optimization problems have several difficulties:
  1. They are stochastic: Performance depends on customer demand, which is random. What sort of demands can we assume? Normal? Poisson? General?
  2. They are discrete: In most cases you can't order .364 of a product.

Supply Chain Optimization is Hard

- Most of these questions can be posed as optimization problems. These optimization problems have several difficulties:

  1. They are stochastic: Performance depends on customer demand, which is random. What sort of demands can we assume? Normal? Poisson? General?
  2. They are discrete: In most cases you can't order .364 of a product.
  3. They are non-separable: The ordering policy for product A will affect product B and vice versa.

Supply Chain Optimization is Hard

- Most of these questions can be posed as optimization problems. These optimization problems have several difficulties:
  1. They are stochastic: Performance depends on customer demand, which is random. What sort of demands can we assume? Normal? Poisson? General?
  2. They are discrete: In most cases you can't order .364 of a product.
  3. They are non-separable: The ordering policy for product A will affect product B and vice versa.
  4. They are big: Real-world supply chains can have thousands of products in hundreds of locations, and need to be optimized over dozens of time periods, or even infinite horizon.

Supply Chain Optimization is Hard

- Most of these questions can be posed as optimization problems. These optimization problems have several difficulties:

  1. They are stochastic: Performance depends on customer demand, which is random. What sort of demands can we assume? Normal? Poisson? General?
  2. They are discrete: In most cases you can't order .364 of a product.
  3. They are non-separable: The ordering policy for product A will affect product B and vice versa.
  4. They are big: Real-world supply chains can have thousands of products in hundreds of locations, and need to be optimized over dozens of time periods, or even infinite horizon.
  5. They are complicated: You can run into capacities, backlogging or lost sales or a mix of these, release dates/due dates/time windows/precedence constraints, etc, etc.

Motivation for Discrete Convexity

- In non-linear optimization, convexity leads to much faster solution times.

Motivation for Discrete Convexity

- In non-linear optimization, convexity leads to much faster solution times.
- Idea: try to find an analogue for optimization of functions defined on integer lattice. Desired properties:

Motivation for Discrete Convexity

- In non-linear optimization, convexity leads to much faster solution times.
- Idea: try to find an analogue for optimization of functions defined on integer lattice. Desired properties:
    1. Local optimality leads to global optimality.

Motivation for Discrete Convexity

- In non-linear optimization, convexity leads to much faster solution times.
- Idea: try to find an analogue for optimization of functions defined on integer lattice. Desired properties:
    1. Local optimality leads to global optimality.
    2. Analogue to Fenchel duality.

Motivation for Discrete Convexity

- In non-linear optimization, convexity leads to much faster solution times.
- Idea: try to find an analogue for optimization of functions defined on integer lattice. Desired properties:
  1. Local optimality leads to global optimality.
  2. Analogue to Fenchel duality.
  3. Separation theorem.

Motivation for Discrete Convexity

- In non-linear optimization, convexity leads to much faster solution times.
- Idea: try to find an analogue for optimization of functions defined on integer lattice. Desired properties:
  1. Local optimality leads to global optimality.
  2. Analogue to Fenchel duality.
  3. Separation theorem.
  4. Reduces to well-know concepts like submodularity or matroids on 0-1 vectors.

Motivation for Discrete Convexity

- In non-linear optimization, convexity leads to much faster solution times.
- Idea: try to find an analogue for optimization of functions defined on integer lattice. Desired properties:
  1. Local optimality leads to global optimality.
  2. Analogue to Fenchel duality.
  3. Separation theorem.
  4. Reduces to well-know concepts like submodularity or matroids on 0-1 vectors.
  5. Has efficient minimization algorithms.

Definitions of Discrete Convexity

These concepts were first defined by Kazuo Murota.

- We first define $\mathrm{L}^\natural$-convex functions.

Definitions of Discrete Convexity

These concepts were first defined by Kazuo Murota.

- We first define $\mathrm{L}^\natural$-convex functions.
- Suppose that $f : \mathbb{Z}^n \to \mathbb{R}$.

Definitions of Discrete Convexity

These concepts were first defined by Kazuo Murota.

- We first define $\mathrm{L}^\natural$-convex functions.
- Suppose that $f : \mathbb{Z}^n \to \mathbb{R}$.
- Then f is $\mathrm{L}^\natural$-convex if it satisfies the discrete midpoint property:

$$f(x) + f(y) \geq f(\lceil \tfrac{1}{2}(x+y) \rceil) + f(\lfloor \tfrac{1}{2}(x+y) \rfloor)$$

for all $x, y \in \mathbb{Z}^n$ with $||x - y||_\infty \leq 2$.

Definitions of Discrete Convexity

These concepts were first defined by Kazuo Murota.

- We first define L$^\natural$-convex functions.
- Suppose that $f : \mathbb{Z}^n \to \mathbb{R}$.
- Then f is L$^\natural$-convex if it satisfies the discrete midpoint property:

$$f(x) + f(y) \geq f(\lceil \tfrac{1}{2}(x+y) \rceil) + f(\lfloor \tfrac{1}{2}(x+y) \rfloor)$$

  for all $x, y \in \mathbb{Z}^n$ with $||x - y||_\infty \leq 2$.
- It can be shown that this implies generalized submodularity:

$$f(x) + f(y) \geq f(\min(x, y)) + f(\max(x, y))$$

  but that submodularity does not imply L$^\natural$-convexity.

Definitions of Discrete Convexity

These concepts were first defined by Kazuo Murota.

- We first define $\mathrm{L}^\natural$-convex functions.
- Suppose that $f : \mathbb{Z}^n \to \mathbb{R}$.
- Then f is $\mathrm{L}^\natural$-convex if it satisfies the discrete midpoint property:

$$f(x) + f(y) \geq f(\lceil \tfrac{1}{2}(x+y) \rceil) + f(\lfloor \tfrac{1}{2}(x+y) \rfloor)$$

  for all $x, y \in \mathbb{Z}^n$ with $||x - y||_\infty \leq 2$.
- It can be shown that this implies generalized submodularity:

$$f(x) + f(y) \geq f(\min(x,y)) + f(\max(x,y))$$

  but that submodularity does not imply $\mathrm{L}^\natural$-convexity.
- There is a dual notion called M-convexity (related to valuated matroids) that doesn't concern us here.

Definitions of Discrete Convexity

These concepts were first defined by Kazuo Murota.

- We first define L$^\natural$-convex functions.
- Suppose that $f : \mathbb{Z}^n \to \mathbb{R}$.
- Then f is L$^\natural$-convex if it satisfies the discrete midpoint property:

$$f(x) + f(y) \geq f(\lceil \tfrac{1}{2}(x+y) \rceil) + f(\lfloor \tfrac{1}{2}(x+y) \rfloor)$$

  for all $x, y \in \mathbb{Z}^n$ with $||x - y||_\infty \leq 2$.
- It can be shown that this implies generalized submodularity:

$$f(x) + f(y) \geq f(\min(x,y)) + f(\max(x,y))$$

  but that submodularity does not imply L$^\natural$-convexity.
- There is a dual notion called M-convexity (related to valuated matroids) that doesn't concern us here.
- We get all items on our wishlist for L- and M-convex functions, including efficient minimization algorithms.

Discrete Convexity to the Rescue?

- Given this definition, why is $L^\natural$-convexity appealing in the supply chain context?

Discrete Convexity to the Rescue?

- Given this definition, why is $L^\natural$-convexity appealing in the supply chain context?
    1. Submodularity: It was already understood that submodularity arises surprisingly and usefully often in supply chain models.

Discrete Convexity to the Rescue?

- Given this definition, why is $L^\natural$-convexity appealing in the supply chain context?
  1. Submodularity: It was already understood that submodularity arises surprisingly and usefully often in supply chain models.
  2. Integer lattice: Many supply chain models have decision variables that are naturally general integer vectors, and where component-wise min and max make sense.

Discrete Convexity to the Rescue?

- Given this definition, why is $L^\natural$-convexity appealing in the supply chain context?

  1. Submodularity: It was already understood that submodularity arises surprisingly and usefully often in supply chain models.
  2. Integer lattice: Many supply chain models have decision variables that are naturally general integer vectors, and where component-wise min and max make sense.
  3. Non-separable costs: Many supply chain models have non-separable costs, and $L^\natural$-convexity can deal gracefully with this.

Discrete Convexity to the Rescue?

- Given this definition, why is $L^\natural$-convexity appealing in the supply chain context?

    1. Submodularity: It was already understood that submodularity arises surprisingly and usefully often in supply chain models.
    2. Integer lattice: Many supply chain models have decision variables that are naturally general integer vectors, and where component-wise min and max make sense.
    3. Non-separable costs: Many supply chain models have non-separable costs, and $L^\natural$-convexity can deal gracefully with this.
    4. Good qualitative properties: If you can prove $L^\natural$-convexity, then you understand a lot about the qualitative sensitivity of your problem.

Discrete Convexity to the Rescue?

- Given this definition, why is $L^\natural$-convexity appealing in the supply chain context?
  1. Submodularity: It was already understood that submodularity arises surprisingly and usefully often in supply chain models.
  2. Integer lattice: Many supply chain models have decision variables that are naturally general integer vectors, and where component-wise min and max make sense.
  3. Non-separable costs: Many supply chain models have non-separable costs, and $L^\natural$-convexity can deal gracefully with this.
  4. Good qualitative properties: If you can prove $L^\natural$-convexity, then you understand a lot about the qualitative sensitivity of your problem.
  5. Efficient solution algorithms: If a problem is $L^\natural$-convex, then there is a polynomial-time minimization algorithm for it.

Outline

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.
- Dell keeps in stock some inventory I_j of each component j, where j belongs to a set J of all possible components.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.
- Dell keeps in stock some inventory I_j of each component j, where j belongs to a set J of all possible components.
- In this context a product is essentially a subset of components.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.
- Dell keeps in stock some inventory I_j of each component j, where j belongs to a set J of all possible components.
- In this context a product is essentially a subset of components.
  - Assume that the time to assemble components into the product is negligible.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.
- Dell keeps in stock some inventory I_j of each component j, where j belongs to a set J of all possible components.
- In this context a product is essentially a subset of components.
  - Assume that the time to assemble components into the product is negligible.
  - Assume that each product uses either zero of one of each component.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.
- Dell keeps in stock some inventory I_j of each component j, where j belongs to a set J of all possible components.
- In this context a product is essentially a subset of components.
  - Assume that the time to assemble components into the product is negligible.
  - Assume that each product uses either zero of one of each component.
- When an order for a product $P \subseteq J$ arrives, Dell takes the components out of inventory and assembles P and sends it to the customer.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.
- Dell keeps in stock some inventory I_j of each component j, where j belongs to a set J of all possible components.
- In this context a product is essentially a subset of components.
  - Assume that the time to assemble components into the product is negligible.
  - Assume that each product uses either zero of one of each component.
- When an order for a product $P \subseteq J$ arrives, Dell takes the components out of inventory and assembles P and sends it to the customer.
  - Assume that each product is ordered only one at a time.

What is Assemble to Order (ATO)?

- We follow the model from the paper "Order-Based Cost Optimization in Assemble-to-Order Systems" by Y. Lu and J-S. Song, *OR* 2005.
- Imagine, e.g., a company like Dell Computers that makes customized products out of components.
- Dell keeps in stock some inventory I_j of each component j, where j belongs to a set J of all possible components.
- In this context a product is essentially a subset of components.
  - Assume that the time to assemble components into the product is negligible.
  - Assume that each product uses either zero of one of each component.
- When an order for a product $P \subseteq J$ arrives, Dell takes the components out of inventory and assembles P and sends it to the customer.
  - Assume that each product is ordered only one at a time.
- This is happening in discrete time periods $t = 0, 1, 2, \ldots$.

Stockouts

- What happens if $j \in P$ but $I_j = 0$, i.e., a stockout?

Stockouts

- What happens if $j \in P$ but $I_j = 0$, i.e., a stockout?
- Then we backorder P in a special way:

Stockouts

- What happens if $j \in P$ but $I_j = 0$, i.e., a stockout?
- Then we backorder P in a special way:
  - We tell the customer to wait.

Stockouts

- What happens if $j \in P$ but $I_j = 0$, i.e., a stockout?
- Then we backorder P in a special way:
  - We tell the customer to wait.
  - We set aside, or earmark, one unit of each component $j \in P$ such that $I_j > 0$.

Stockouts

- What happens if $j \in P$ but $I_j = 0$, i.e., a stockout?
- Then we backorder P in a special way:
  - We tell the customer to wait.
  - We set aside, or earmark, one unit of each component $j \in P$ such that $I_j > 0$.
  - As soon as the missing components arrive in future deliveries from our suppliers, we put them together with the earmarked components and assemble and deliver product P to the patient customer.

Stockouts

- What happens if $j \in P$ but $I_j = 0$, i.e., a stockout?
- Then we backorder P in a special way:
  - We tell the customer to wait.
  - We set aside, or earmark, one unit of each component $j \in P$ such that $I_j > 0$.
  - As soon as the missing components arrive in future deliveries from our suppliers, we put them together with the earmarked components and assemble and deliver product P to the patient customer.
  - Thus demand from backlogged products takes precedence over subsequent orders that use the same component - we satisfy orders in first come, first served (FCFS) fashion.

The Ordering Process

- Assume that each component comes from a different supplier.

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.
- When do we order?

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.
- When do we order?
  - This is a complicated situation where the form of an optimal ordering policy is far from clear.

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.
- When do we order?
  - This is a complicated situation where the form of an optimal ordering policy is far from clear.
  - To try to make things tractable, we will assume that we follow a base stock ordering policy, which is common in practice.

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.
- When do we order?
  - This is a complicated situation where the form of an optimal ordering policy is far from clear.
  - To try to make things tractable, we will assume that we follow a base stock ordering policy, which is common in practice.
  - For each component j we decide on a base stock level $s_j \geq 0$.

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.
- When do we order?
  - This is a complicated situation where the form of an optimal ordering policy is far from clear.
  - To try to make things tractable, we will assume that we follow a base stock ordering policy, which is common in practice.
  - For each component j we decide on a base stock level $s_j \geq 0$.
  - Whenever a customer orders product P with $j \in P$, if the inventory position of $j =$ (inventory on hand) $+$ (inventory on order) $-$ (backorders) is less than s_j, then we immediately order a replacement unit of j.

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.
- When do we order?
  - This is a complicated situation where the form of an optimal ordering policy is far from clear.
  - To try to make things tractable, we will assume that we follow a base stock ordering policy, which is common in practice.
  - For each component j we decide on a base stock level $s_j \geq 0$.
  - Whenever a customer orders product P with $j \in P$, if the inventory position of $j =$ (inventory on hand) $+$ (inventory on order) $-$ (backorders) is less than s_j, then we immediately order a replacement unit of j.
  - Note that "inventory on hand" does not include earmarked components.

The Ordering Process

- Assume that each component comes from a different supplier.
- When we order component j from its supplier, the order arrives after some leadtime L_j, which could be random.
- When do we order?
  - This is a complicated situation where the form of an optimal ordering policy is far from clear.
  - To try to make things tractable, we will assume that we follow a base stock ordering policy, which is common in practice.
  - For each component j we decide on a base stock level $s_j \geq 0$.
  - Whenever a customer orders product P with $j \in P$, if the inventory position of $j =$ (inventory on hand) $+$ (inventory on order) $-$ (backorders) is less than s_j, then we immediately order a replacement unit of j.
  - Note that "inventory on hand" does not include earmarked components.
  - In practice, this means that for each customer order with $j \in P$, we immediately order a replacement unit from j's supplier.

Costs

- There is a per-period holding cost h_j levied on each unit of component j in inventory.

Costs

- There is a per-period holding cost h_j levied on each unit of component j in inventory.
  - We have to be careful about inventory: We have both available (non-earmarked) inventory I_j and earmarked inventory F_j.

Costs

- There is a per-period holding cost h_j levied on each unit of component j in inventory.
  - We have to be careful about inventory: We have both available (non-earmarked) inventory I_j and earmarked inventory F_j.
  - Holding cost is assessed on both of these.

Costs

- There is a per-period holding cost h_j levied on each unit of component j in inventory.
  - We have to be careful about inventory: We have both available (non-earmarked) inventory I_j and earmarked inventory F_j.
  - Holding cost is assessed on both of these.
- There is a per-period backorder cost b_P levied on each unit of product P when it is backordered.

Costs

- There is a per-period holding cost h_j levied on each unit of component j in inventory.
  - We have to be careful about inventory: We have both available (non-earmarked) inventory I_j and earmarked inventory F_j.
  - Holding cost is assessed on both of these.
- There is a per-period backorder cost b_P levied on each unit of product P when it is backordered.
  - The interaction between per-component holding costs, and per-product backorder costs, including that the FCFS fulfillment policy means that the choice of s_j affects not only the costs for component j, but also the costs of other items, makes this a difficult problem.

Demand Process

- Assume that customer orders arrive in a Poisson process at rate λ.

Demand Process

- Assume that customer orders arrive in a Poisson process at rate λ.
- Further assume that the probability of a customer order being for subset P is q_P, so that $\sum_P q_P = 1$.

Demand Process

- Assume that customer orders arrive in a Poisson process at rate λ.
- Further assume that the probability of a customer order being for subset P is q_P, so that $\sum_P q_P = 1$.
- Thus orders for product P arrive as a Poisson process at rate $q_P \lambda$.

Demand Process

- Assume that customer orders arrive in a Poisson process at rate λ.
- Further assume that the probability of a customer order being for subset P is q_P, so that $\sum_P q_P = 1$.
- Thus orders for product P arrive as a Poisson process at rate $q_P \lambda$.
- We now have the broad outlines of our problem: choose the base stock levels s_j for each $j \in J$ so as to minimize the expected sum of holding and backorder costs in the long run.

Demand Process

- Assume that customer orders arrive in a Poisson process at rate λ.
- Further assume that the probability of a customer order being for subset P is q_P, so that $\sum_P q_P = 1$.
- Thus orders for product P arrive as a Poisson process at rate $q_P \lambda$.
- We now have the broad outlines of our problem: choose the base stock levels s_j for each $j \in J$ so as to minimize the expected sum of holding and backorder costs in the long run.
- We have the classic tension between holding costs and backorder penalties here: if s_j is big then we make B_j small and so a small backorder penalty, but we make I_j big, and so a big holding cost.

Demand Process

- Assume that customer orders arrive in a Poisson process at rate λ.
- Further assume that the probability of a customer order being for subset P is q_P, so that $\sum_P q_P = 1$.
- Thus orders for product P arrive as a Poisson process at rate $q_P \lambda$.
- We now have the broad outlines of our problem: choose the base stock levels s_j for each $j \in J$ so as to minimize the expected sum of holding and backorder costs in the long run.
- We have the classic tension between holding costs and backorder penalties here: if s_j is big then we make B_j small and so a small backorder penalty, but we make I_j big, and so a big holding cost.
- Our decision vector s takes values on the integer lattice, and is non-separable.

Demand Process

- Assume that customer orders arrive in a Poisson process at rate λ.
- Further assume that the probability of a customer order being for subset P is q_P, so that $\sum_P q_P = 1$.
- Thus orders for product P arrive as a Poisson process at rate $q_P \lambda$.
- We now have the broad outlines of our problem: choose the base stock levels s_j for each $j \in J$ so as to minimize the expected sum of holding and backorder costs in the long run.
- We have the classic tension between holding costs and backorder penalties here: if s_j is big then we make B_j small and so a small backorder penalty, but we make I_j big, and so a big holding cost.
- Our decision vector s takes values on the integer lattice, and is non-separable.
- Therefore classic optimization techniques will not work unless we can prove that there is additional structure here.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.
- Notice that $I_j = (s_j - X_j)^+$ and $B_j = (X_j - s_j)^+$.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.
- Notice that $I_j = (s_j - X_j)^+$ and $B_j = (X_j - s_j)^+$.
- Thus $I_j - B_j = s_j - X_j$, or $I_j = s_j - X_j + B_j$.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.
- Notice that $I_j = (s_j - X_j)^+$ and $B_j = (X_j - s_j)^+$.
- Thus $I_j - B_j = s_j - X_j$, or $I_j = s_j - X_j + B_j$.
- Holding costs are also assessed on earmarked units, denoted by F_j.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.
- Notice that $I_j = (s_j - X_j)^+$ and $B_j = (X_j - s_j)^+$.
- Thus $I_j - B_j = s_j - X_j$, or $I_j = s_j - X_j + B_j$.
- Holding costs are also assessed on earmarked units, denoted by F_j.
- Define B_j^P as the number of backorders for j due to product P, so that $B_j = \sum_{P \ni j} B_j^P$. Also define B^P as the total number of backorders for product P.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.
- Notice that $I_j = (s_j - X_j)^+$ and $B_j = (X_j - s_j)^+$.
- Thus $I_j - B_j = s_j - X_j$, or $I_j = s_j - X_j + B_j$.
- Holding costs are also assessed on earmarked units, denoted by F_j.
- Define B_j^P as the number of backorders for j due to product P, so that $B_j = \sum_{P \ni j} B_j^P$. Also define B^P as the total number of backorders for product P.
- Then $F_j = \sum_{P \ni j}(B^P - B_j^P) = \sum_{P \ni j} B^P - B_j$.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.
- Notice that $I_j = (s_j - X_j)^+$ and $B_j = (X_j - s_j)^+$.
- Thus $I_j - B_j = s_j - X_j$, or $I_j = s_j - X_j + B_j$.
- Holding costs are also assessed on earmarked units, denoted by F_j.
- Define B_j^P as the number of backorders for j due to product P, so that $B_j = \sum_{P \ni j} B_j^P$. Also define B^P as the total number of backorders for product P.
- Then $F_j = \sum_{P \ni j}(B^P - B_j^P) = \sum_{P \ni j} B^P - B_j$.
- Thus $I_j + F_j =$ $(s_j - X_j + B_j) + \sum_{P \ni j} B^P - B_j = s_j - X_j + \sum_{P \ni j} B^P$.

The Objective Function 1

- Define $X_j(t)$ to be the number of outstanding orders for component j at time t (and suppress t), and B_j to be the number of units of j that are backordered.
- Notice that $I_j = (s_j - X_j)^+$ and $B_j = (X_j - s_j)^+$.
- Thus $I_j - B_j = s_j - X_j$, or $I_j = s_j - X_j + B_j$.
- Holding costs are also assessed on earmarked units, denoted by F_j.
- Define B_j^P as the number of backorders for j due to product P, so that $B_j = \sum_{P \ni j} B_j^P$. Also define B^P as the total number of backorders for product P.
- Then $F_j = \sum_{P \ni j}(B^P - B_j^P) = \sum_{P \ni j} B^P - B_j$.
- Thus $I_j + F_j =$ $(s_j - X_j + B_j) + \sum_{P \ni j} B^P - B_j = s_j - X_j + \sum_{P \ni j} B^P$.
- Thus $C(s) = \sum_j h_j E(I_j + F_j) + \sum_P b^P E(B^P) =$ $\sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) - \sum_j h_j E(X_j)$, where $\tilde{b}^P = b^P + \sum_{j \in P} h_j$.

The Objective Function 2

- Recall $C(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) - \sum_j h_j E(X_j)$.

The Objective Function 2

- Recall $C(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) - \sum_j h_j E(X_j)$.
- Think carefully: what depends on s?

The Objective Function 2

- Recall $C(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) - \sum_j h_j E(X_j)$.
- Think carefully: what depends on s?
- Answer: not $\sum_j h_j E(X_j)$.

The Objective Function 2

- Recall $C(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) - \sum_j h_j E(X_j)$.
- Think carefully: what depends on s?
- Answer: not $\sum_j h_j E(X_j)$.
- So we want to solve $\min_s \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) = \min_s \tilde{C}(s)$.

The Objective Function 2

- Recall $C(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) - \sum_j h_j E(X_j)$.
- Think carefully: what depends on s?
- Answer: not $\sum_j h_j E(X_j)$.
- So we want to solve
  $\min_s \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) = \min_s \tilde{C}(s)$.
- The term $\sum_j h_j s_j$ is separable and linear, so easy.

The Objective Function 2

- Recall $C(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) - \sum_j h_j E(X_j)$.
- Think carefully: what depends on s?
- Answer: not $\sum_j h_j E(X_j)$.
- So we want to solve $\min_s \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P) = \min_s \tilde{C}(s)$.
- The term $\sum_j h_j s_j$ is separable and linear, so easy.
- The term $\sum_P \tilde{b}^P E(B^P)$ is non-separable and non-linear, so (maybe) difficult.

The Main Claim

- A main result in Lu and Song's paper is:

The Main Claim

- A main result in Lu and Song's paper is:
- **Proposition 1 (c):** $\tilde{C}(s)$ is L$^\natural$-convex.

The Main Claim

- A main result in Lu and Song's paper is:
- **Proposition 1 (c):** $\tilde{C}(s)$ is L$^\natural$-convex.
- Recall that this is equivalent to having the discrete midpoint property that for all s', s'' with $||s' - s''||_\infty \leq 2$:

$$\tilde{C}(s') + \tilde{C}(s'') \geq \tilde{C}\left(\left\lfloor \frac{s' + s''}{2} \right\rfloor\right) + \tilde{C}\left(\left\lceil \frac{s' + s''}{2} \right\rceil\right).$$

Outline

The Data

- Start with $J = \{1, 2\}$, and two products: $P = \{1, 2\}$ and $Q = \{1\}$. We use superscript "12" in place of "P" and "1" in place of "Q".

The Data

- Start with $J = \{1, 2\}$, and two products: $P = \{1, 2\}$ and $Q = \{1\}$. We use superscript "12" in place of "P" and "1" in place of "Q".
- The general objective $\tilde{C}(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P)$ is now

$$\begin{aligned}\tilde{C}(s_1, s_2) &= h_1 s_1 + h_2 s_2 + (b^{12} + h_1 + h_2) E(B^{12}(s_1, s_2)) \\ &\quad + (b^1 + h_1) E(B^1(s_1, s_2))\end{aligned}$$

The Data

- Start with $J = \{1, 2\}$, and two products: $P = \{1, 2\}$ and $Q = \{1\}$. We use superscript "12" in place of "P" and "1" in place of "Q".
- The general objective $\tilde{C}(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P)$ is now

$$\begin{aligned}\tilde{C}(s_1, s_2) \quad = \quad & h_1 s_1 + h_2 s_2 + (b^{12} + h_1 + h_2) E(B^{12}(s_1, s_2)) \\ & + (b^1 + h_1) E(B^1(s_1, s_2))\end{aligned}$$

- Let's further simplify by setting $b^1 = h_1 = h_2 = 0$, so that $\tilde{C}$ becomes
$$\tilde{C}(s_1, s_2) = b^{12} E(B^{12}(s_1, s_2)).$$

The Data

- Start with $J = \{1, 2\}$, and two products: $P = \{1, 2\}$ and $Q = \{1\}$. We use superscript "12" in place of "P" and "1" in place of "Q".
- The general objective $\tilde{C}(s) = \sum_j h_j s_j + \sum_P \tilde{b}^P E(B^P)$ is now

$$\begin{aligned} \tilde{C}(s_1, s_2) &= h_1 s_1 + h_2 s_2 + (b^{12} + h_1 + h_2) E(B^{12}(s_1, s_2)) \\ &\quad + (b^1 + h_1) E(B^1(s_1, s_2)) \end{aligned}$$

- Let's further simplify by setting $b^1 = h_1 = h_2 = 0$, so that $\tilde{C}$ becomes
$$\tilde{C}(s_1, s_2) = b^{12} E(B^{12}(s_1, s_2)).$$
- Now verifying the discrete midpoint property for $\tilde{C}$ reduces to verifying it for $E(B^{12}(s_1, s_2))$.

The Instance

- We assume that both leadtimes are deterministic, and equal L.

The Instance

- We assume that both leadtimes are deterministic, and equal L.
- Now set $(s_1', s_2') = (0, 0)$ and $(s_1'', s_2'') = (2, 1)$.

The Instance

- We assume that both leadtimes are deterministic, and equal L.
- Now set $(s_1', s_2') = (0, 0)$ and $(s_1'', s_2'') = (2, 1)$.
- Thus $\left\lfloor \frac{s' + s''}{2} \right\rfloor = (1, 0)$ and $\left\lceil \frac{s' + s''}{2} \right\rceil = (1, 1)$.

The Instance

- We assume that both leadtimes are deterministic, and equal L.
- Now set $(s'_1, s'_2) = (0,0)$ and $(s''_1, s''_2) = (2,1)$.
- Thus $\left\lfloor \frac{s'+s''}{2} \right\rfloor = (1,0)$ and $\left\lceil \frac{s'+s''}{2} \right\rceil = (1,1)$.
- Thus we need to verify that $E(B^{12}(0,0)) + E(B^{12}(2,1)) \geq E(B^{12}(1,0)) + E(B^{12}(1,1))$.

The Instance

- We assume that both leadtimes are deterministic, and equal L.
- Now set $(s'_1, s'_2) = (0, 0)$ and $(s''_1, s''_2) = (2, 1)$.
- Thus $\left\lfloor \frac{s'+s''}{2} \right\rfloor = (1, 0)$ and $\left\lceil \frac{s'+s''}{2} \right\rceil = (1, 1)$.
- Thus we need to verify that $E(B^{12}(0, 0)) + E(B^{12}(2, 1)) \geq E(B^{12}(1, 0)) + E(B^{12}(1, 1))$.
- Instead we will show that $E(B^{12}(0, 0)) + E(B^{12}(2, 1)) < E(B^{12}(1, 0)) + E(B^{12}(1, 1))$.

Proving the Counterexample 1

- First focus on $E(B^{12}(0,0))$ and $E(B^{12}(1,0))$ and recall that these are expected backorders for $P = \{1,2\}$.

Proving the Counterexample 1

- First focus on $E(B^{12}(0,0))$ and $E(B^{12}(1,0))$ and recall that these are expected backorders for $P = \{1,2\}$.
- Both $(0,0)$ and $(1,0)$ keep zero units of component 2 in stock. Thus every time that a customer orders P, a unit of component 2 is ordered, and so the order for P can't be filled until the component 2 arrives in L time periods.

Proving the Counterexample 1

- First focus on $E(B^{12}(0,0))$ and $E(B^{12}(1,0))$ and recall that these are expected backorders for $P = \{1,2\}$.
- Both $(0,0)$ and $(1,0)$ keep zero units of component 2 in stock. Thus every time that a customer orders P, a unit of component 2 is ordered, and so the order for P can't be filled until the component 2 arrives in L time periods.
- Therefore, under every demand scenario, both $(0,0)$ and $(1,0)$ generate exactly the same sequence of backorders of P, and so $E(B^{12}(0,0)) = E(B^{12}(1,0))$.

Proving the Counterexample 2

- Now focus instead on $E(B^{12}(2,1))$ and $E(B^{12}(1,1))$. More stock always reduces backorders, so $E(B^{12}(2,1)) \leq E(B^{12}(1,1))$. We'll show that in fact $E(B^{12}(2,1)) < E(B^{12}(1,1))$.

Proving the Counterexample 2

- Now focus instead on $E(B^{12}(2,1))$ and $E(B^{12}(1,1))$. More stock always reduces backorders, so $E(B^{12}(2,1)) \leq E(B^{12}(1,1))$. We'll show that in fact $E(B^{12}(2,1)) < E(B^{12}(1,1))$.
- At any given time t, there is a positive probability that the demand stream in $(t-L,t]$ will be one order for $Q=\{1\}$ followed by one order for $P=\{1,2\}$.

Proving the Counterexample 2

- Now focus instead on $E(B^{12}(2,1))$ and $E(B^{12}(1,1))$. More stock always reduces backorders, so $E(B^{12}(2,1)) \leq E(B^{12}(1,1))$. We'll show that in fact $E(B^{12}(2,1)) < E(B^{12}(1,1))$.
- At any given time t, there is a positive probability that the demand stream in $(t-L,t]$ will be one order for $Q=\{1\}$ followed by one order for $P=\{1,2\}$.
- In this scenario, the $(2,1)$ system will not have a backorder for $P=\{1,2\}$, whereas the $(1,1)$ system will have a backorder for $P=\{1,2\}$ (since the prior order for $Q=\{1\}$ "used up" the stock of component 1 before it could be used to satisfy the order for P).

Proving the Counterexample 2

- Now focus instead on $E(B^{12}(2,1))$ and $E(B^{12}(1,1))$. More stock always reduces backorders, so $E(B^{12}(2,1)) \leq E(B^{12}(1,1))$. We'll show that in fact $E(B^{12}(2,1)) < E(B^{12}(1,1))$.
- At any given time t, there is a positive probability that the demand stream in $(t-L,t]$ will be one order for $Q=\{1\}$ followed by one order for $P=\{1,2\}$.
- In this scenario, the $(2,1)$ system will not have a backorder for $P=\{1,2\}$, whereas the $(1,1)$ system will have a backorder for $P=\{1,2\}$ (since the prior order for $Q=\{1\}$ "used up" the stock of component 1 before it could be used to satisfy the order for P).
- This proves that $E(B^{12}(2,1)) < E(B^{12}(1,1))$.

Proving the Counterexample 2

- Now focus instead on $E(B^{12}(2,1))$ and $E(B^{12}(1,1))$. More stock always reduces backorders, so $E(B^{12}(2,1)) \le E(B^{12}(1,1))$. We'll show that in fact $E(B^{12}(2,1)) < E(B^{12}(1,1))$.
- At any given time t, there is a positive probability that the demand stream in $(t-L,t]$ will be one order for $Q=\{1\}$ followed by one order for $P=\{1,2\}$.
- In this scenario, the $(2,1)$ system will not have a backorder for $P=\{1,2\}$, whereas the $(1,1)$ system will have a backorder for $P=\{1,2\}$ (since the prior order for $Q=\{1\}$ "used up" the stock of component 1 before it could be used to satisfy the order for P).
- This proves that $E(B^{12}(2,1)) < E(B^{12}(1,1))$.
- Since we had $E(B^{12}(0,0)) = E(B^{12}(1,0))$, we get $E(B^{12}(0,0)) + E(B^{12}(2,1)) < E(B^{12}(1,0)) + E(B^{12}(1,1))$, and so $\tilde{C}(s)$ is not in general L$^\natural$-convex.

Implications of the Counterexample

- We have communicated this proposed counterexample to Lu and Song and they agree with it.

Implications of the Counterexample

- We have communicated this proposed counterexample to Lu and Song and they agree with it.
- The counterexample is robust:

Implications of the Counterexample

- We have communicated this proposed counterexample to Lu and Song and they agree with it.
- The counterexample is robust:
  - We show that the counterexample translates to “big” s.

Implications of the Counterexample

- We have communicated this proposed counterexample to Lu and Song and they agree with it.
- The counterexample is robust:
  - We show that the counterexample translates to “big” s.
  - We show that the counterexample can be adapted to any definition of discrete convexity in the class of “D-convex” functions (Ui).

Implications of the Counterexample

- We have communicated this proposed counterexample to Lu and Song and they agree with it.
- The counterexample is robust:
  - We show that the counterexample translates to "big" s.
  - We show that the counterexample can be adapted to any definition of discrete convexity in the class of "D-convex" functions (Ui).
- There does not appear to be any meaningful change to the model that would make $\tilde{C}(s)$ L$^\natural$-convex, due to an inherent flaw in the proof.

Implications of the Counterexample

- We have communicated this proposed counterexample to Lu and Song and they agree with it.
- The counterexample is robust:
  - We show that the counterexample translates to "big" s.
  - We show that the counterexample can be adapted to any definition of discrete convexity in the class of "D-convex" functions (Ui).
- There does not appear to be any meaningful change to the model that would make $\tilde{C}(s)$ $\mathrm{L}^{\natural}$-convex, due to an inherent flaw in the proof.
- Although $\tilde{C}(s)$ is not $\mathrm{L}^{\natural}$-convex, it is (uncontroversially) submodular and is convex in each coordinate direction.

Implications of the Counterexample

- We have communicated this proposed counterexample to Lu and Song and they agree with it.
- The counterexample is robust:
  - We show that the counterexample translates to "big" s.
  - We show that the counterexample can be adapted to any definition of discrete convexity in the class of "D-convex" functions (Ui).
- There does not appear to be any meaningful change to the model that would make $\tilde{C}(s)$ L$^\natural$-convex, due to an inherent flaw in the proof.
- Although $\tilde{C}(s)$ is not L$^\natural$-convex, it is (uncontroversially) submodular and is convex in each coordinate direction.
- We now show how to use these properties to get a pseudo-polynomial algorithm.

Outline

Submodularity on a box in $\mathbb{R}^n$

- Lu and Song give nice bound l and u such that the optimal solution is contained in the box $[l, u] \equiv \{s \in \mathbb{R}^n \mid l \leq s \leq u\}$.

Submodularity on a box in $\mathbb{R}^n$

- Lu and Song give nice bound l and u such that the optimal solution is contained in the box $[l, u] \equiv \{s \in \mathbb{R}^n \mid l \leq s \leq u\}$.
- Thus we want to solve $\min_{s \in [l,u]} \tilde{C}(s)$, where $\tilde{C}(s)$ is submodular on the integer lattice $[l, u]$ (with component-wise min and max as the lattice operations).

Submodularity on a box in $\mathbb{R}^n$

- Lu and Song give nice bound l and u such that the optimal solution is contained in the box $[l, u] \equiv \{s \in \mathbb{R}^n \mid l \le s \le u\}$.
- Thus we want to solve $\min_{s \in [l,u]} \tilde{C}(s)$, where $\tilde{C}(s)$ is submodular on the integer lattice $[l, u]$ (with component-wise min and max as the lattice operations).
- There is a general technique for solving such problems descended from Birkhoff's Theorem on distributive lattices.

Submodularity on a box in $\mathbb{R}^n$

- Lu and Song give nice bound l and u such that the optimal solution is contained in the box $[l, u] \equiv \{s \in \mathbb{R}^n \mid l \leq s \leq u\}$.
- Thus we want to solve $\min_{s \in [l,u]} \tilde{C}(s)$, where $\tilde{C}(s)$ is submodular on the integer lattice $[l, u]$ (with component-wise min and max as the lattice operations).
- There is a general technique for solving such problems descended from Birkhoff's Theorem on distributive lattices.
  - The technique was developed by Iri in '70, '84 as part of his theory of "principal partitions".

Submodularity on a box in $\mathbb{R}^n$

- Lu and Song give nice bound l and u such that the optimal solution is contained in the box $[l, u] \equiv \{s \in \mathbb{R}^n \mid l \leq s \leq u\}$.
- Thus we want to solve $\min_{s \in [l,u]} \tilde{C}(s)$, where $\tilde{C}(s)$ is submodular on the integer lattice $[l, u]$ (with component-wise min and max as the lattice operations).
- There is a general technique for solving such problems descended from Birkhoff's Theorem on distributive lattices.
  - The technique was developed by Iri in '70, '84 as part of his theory of "principal partitions".
  - Another version was developed by Queyranne and Tardella '92.

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.
- For the vector lattice $[l, u]$ it can be shown that the set J of join-irreducible elements is $J \equiv \{x \in [l, u] \mid \exists\ 1 \leq j \leq n \text{ s.t. } x_i = l_i \forall i \neq j, \text{ and } x_j > l_j\}$.

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.
- For the vector lattice $[l, u]$ it can be shown that the set J of join-irreducible elements is $J \equiv \{x \in [l, u] \mid \exists\ 1 \le j \le n$ s.t. $x_i = l_i \forall i \neq j$, and $x_j > l_j\}$.
- Any distributive L has a partial order "$\preceq$"; for $[l, u]$ this is just "$\le$".

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.
- For the vector lattice $[l, u]$ it can be shown that the set J of join-irreducible elements is $J \equiv \{x \in [l, u] \mid \exists\ 1 \leq j \leq n \text{ s.t. } x_i = l_i \forall i \neq j, \text{ and } x_j > l_j\}$.
- Any distributive L has a partial order "$\preceq$"; for $[l, u]$ this is just "$\leq$".
- Then it can be shown that for $x \in L$, the set $\phi(x) \equiv \{j \in J \mid j \preceq x\}$ satisfies

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.
- For the vector lattice $[l, u]$ it can be shown that the set J of join-irreducible elements is $J \equiv \{x \in [l, u] \mid \exists\ 1 \le j \le n \text{ s.t. } x_i = l_i \forall i \ne j, \text{ and } x_j > l_j\}$.
- Any distributive L has a partial order "$\preceq$"; for $[l, u]$ this is just "$\le$".
- Then it can be shown that for $x \in L$, the set $\phi(x) \equiv \{j \in J \mid j \preceq x\}$ satisfies
    1. $x = \bigvee_{j \in \phi(x)} j$.

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.
- For the vector lattice $[l, u]$ it can be shown that the set J of join-irreducible elements is $J \equiv \{x \in [l, u] \mid \exists\ 1 \leq j \leq n \text{ s.t. } x_i = l_i \forall i \neq j, \text{ and } x_j > l_j\}$.
- Any distributive L has a partial order "$\preceq$"; for $[l, u]$ this is just "$\leq$".
- Then it can be shown that for $x \in L$, the set $\phi(x) \equiv \{j \in J \mid j \preceq x\}$ satisfies
  1. $x = \bigvee_{j \in \phi(x)} j$.
  2. $\mathcal{J} \equiv \{\phi(x) \mid x \in L\}$ is a ring family (closed under $\cap$, $\cup$).

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.
- For the vector lattice $[l, u]$ it can be shown that the set J of join-irreducible elements is $J \equiv \{x \in [l, u] \mid \exists\ 1 \leq j \leq n \text{ s.t. } x_i = l_i \forall i \neq j, \text{ and } x_j > l_j\}$.
- Any distributive L has a partial order "$\preceq$"; for $[l, u]$ this is just "$\leq$".
- Then it can be shown that for $x \in L$, the set $\phi(x) \equiv \{j \in J \mid j \preceq x\}$ satisfies
  1. $x = \bigvee_{j \in \phi(x)} j$.
  2. $\mathcal{J} \equiv \{\phi(x) \mid x \in L\}$ is a ring family (closed under $\cap$, $\cup$).
  3. $\phi(x \wedge y) = \phi(x) \cap \phi(y)$ and $\phi(x \vee y) = \phi(x) \cup \phi(y)$, and so (lattice) submodularity on L carries over to (ordinary) submodularity on $\mathcal{J}$.

Join-irreducible elements

- A key idea: For L a distributive lattice with $x \in L$, we call x join-irreducible if $x = y \vee z$ implies that $y = x$ or $z = x$.
- For the vector lattice $[l, u]$ it can be shown that the set J of join-irreducible elements is $J \equiv \{x \in [l, u] \mid \exists\ 1 \leq j \leq n \text{ s.t. } x_i = l_i \forall i \neq j, \text{ and } x_j > l_j\}$.
- Any distributive L has a partial order "$\preceq$"; for $[l, u]$ this is just "$\leq$".
- Then it can be shown that for $x \in L$, the set $\phi(x) \equiv \{j \in J \mid j \preceq x\}$ satisfies
  1. $x = \bigvee_{j \in \phi(x)} j$.
  2. $\mathcal{J} \equiv \{\phi(x) \mid x \in L\}$ is a ring family (closed under $\cap$, $\cup$).
  3. $\phi(x \wedge y) = \phi(x) \cap \phi(y)$ and $\phi(x \vee y) = \phi(x) \cup \phi(y)$, and so (lattice) submodularity on L carries over to (ordinary) submodularity on $\mathcal{J}$.
- Therefore we can minimize $\tilde{C}(s)$ over $[l, u]$ via minimizing $\tilde{C}(\phi(s))$ over $\mathcal{J}$ using a version of SFMin adapted to ring families.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the “price” we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?
  - No: Look at interval $[l, u] \in \mathbb{Z}^1$; any f is submodular, and we have to look at every point to minimize.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?
  - No: Look at interval $[l, u] \in \mathbb{Z}^1$; any f is submodular, and we have to look at every point to minimize.
- But $|u - l|_1$ might not be big in practice, so pseudo-polynomial might not be bad.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?
  - No: Look at interval $[l, u] \in \mathbb{Z}^1$; any f is submodular, and we have to look at every point to minimize.
- But $|u - l|_1$ might not be big in practice, so pseudo-polynomial might not be bad.
  - At least this is better than brute-force enumeration.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?
  - No: Look at interval $[l, u] \in \mathbb{Z}^1$; any f is submodular, and we have to look at every point to minimize.
- But $|u - l|_1$ might not be big in practice, so pseudo-polynomial might not be bad.
  - At least this is better than brute-force enumeration.
- We could probably do better by exploiting the component-wise convexity via an algorithm from Favati-Tardella to shrink $|u - l|_1$ between SFMin steps.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?
  - No: Look at interval $[l, u] \in \mathbb{Z}^1$; any f is submodular, and we have to look at every point to minimize.
- But $|u - l|_1$ might not be big in practice, so pseudo-polynomial might not be bad.
  - At least this is better than brute-force enumeration.
- We could probably do better by exploiting the component-wise convexity via an algorithm from Favati-Tardella to shrink $|u - l|_1$ between SFMin steps.
  - Natural question again: Is there a polynomial algorithm?

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?
  - No: Look at interval $[l, u] \in \mathbb{Z}^1$; any f is submodular, and we have to look at every point to minimize.
- But $|u - l|_1$ might not be big in practice, so pseudo-polynomial might not be bad.
  - At least this is better than brute-force enumeration.
- We could probably do better by exploiting the component-wise convexity via an algorithm from Favati-Tardella to shrink $|u - l|_1$ between SFMin steps.
  - Natural question again: Is there a polynomial algorithm?
  - Tardella conjecture: no.

Implications of the algorithm

- Notice that $|J| = |u - l|_1$, and so this is only a pseudo-polynomial algorithm.
  - This is the "price" we pay for not being L$^\natural$-convex.
  - Natural question: does there exist a polynomial algorithm?
  - No: Look at interval $[l, u] \in \mathbb{Z}^1$; any f is submodular, and we have to look at every point to minimize.
- But $|u - l|_1$ might not be big in practice, so pseudo-polynomial might not be bad.
  - At least this is better than brute-force enumeration.
- We could probably do better by exploiting the component-wise convexity via an algorithm from Favati-Tardella to shrink $|u - l|_1$ between SFMin steps.
  - Natural question again: Is there a polynomial algorithm?
  - Tardella conjecture: no.
- It's cool that we can use all these sophisticated discrete optimization tools to get an algorithm for this supply chain problem.