SUPPLY CHAIN MODELING

Botond Bertók
Tibor Holczinger

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

Projekt megvalósulása:
2012.04.01-2014.03.31.

Főkedvezményezett neve, címe:
Pannon Egyetem
8200 Veszprém, Egyetem utca 10.

Kedvezményezett neve, címe:
Miskolci Egyetem
3515 Miskolc, Egyetemváros 1.

Az Európai Unió és a Magyar Állam támogatásával nyújtott összes támogatás:
190 417 422.- Ft.

2013.

PANNON Egyetem

MAGYARORSZÁG MEGÚJUL

ÚJ SZÉCHENYI TERV

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

Modeling the supply chain

I. Introduction to supply chain management and supporting information technologies

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Content

- Supply chain management, integrated planning, models
- Information technology

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

1. SUPPLY CHAIN MANAGEMENT, INTEGRATED PLANNING AND MODELS

Management

- Management involves processes for achieving organizational goals by acquiring, divesting and allocation organization resources

Supply chain management

- Supply chain management (SCM) is practiced by organizations that purchase, transform and/or distribute physical products
- It is a relative new term that crystallizes concepts about integrated business planning
  - Logistics
  - Strategy
  - Operation research

Information technology

- The information revolution has accelerated significantly in recent years
  - PCs computing speed
  - e-commerce
  - Data management software
- Widespread implementation of enterprise resource planning (EPR) systems offers integration of supply chain activities

IT ↔ SCM

- Competitive advantage in SCM is not gained simply through faster and cheaper communication of data and ready access data does not automatically lead to better decision making
- To effectively apply IT in managing its supply chain, a company must distinguish between the form and function of transactional IT and analytical IT

IT ↔ SCM

- Manufacturing and distribution companies in a wide range of industries have begun to appreciate this distinction
  - They are seeking to develop or acquire systems that analyze their transactional database to identify plans for redesigning their SC and operating them more efficiently

Optimization models

- Essential components of these systems are optimization models
  - Which can unravel the complex interactions and ripple effects that make SCM difficult and important
  - These are the only analytical tools capable of fully evaluating large numerical databases in helping managers identifying optimal (or demonstrably good) plans

Aim

- Our aim is to examine in detail the roles of data, models and modeling systems in helping companies improve the management of their supply chain
- Principles of modeling system implementation are illustrated by many successful applications

Examples

- Procter & Gamble was able to drive out non value-adding supply chain costs that saved the company over $200 million
- United Parcel Service (UPS) implemented an optimization modeling system that saved $87 million between 2000 and 2002
- Cerestar increased average daily throughput by 20%,which lead to annual benefits in excess of $11 million

Improvements

- Making obvious changes in faulty business procedures
- Strategic investment or divestment of assets
- Better allocation of company resources
- Redesign, reschedule the production
- …

1.1. FUNDAMENTALS OF SUPPLY CHAIN MANAGEMENT

Introduction

- A company’s supply chain contains geographically dispersed
  - Facilities
  - Transportation links
    - Connects facilities along which products flows

Facility

- Where
  - Raw materials and/or
  - Intermediate products and/or
  - Products
- Are
  - Acquired and/or
  - Transformed and/or
  - Stored and/or
  - Sold

Facility

- May be operated by the company
- May be operated by
  - Vendors
  - Costumers
  - Third-party providers
  - Other firms with which the company has business arrangements

Goal

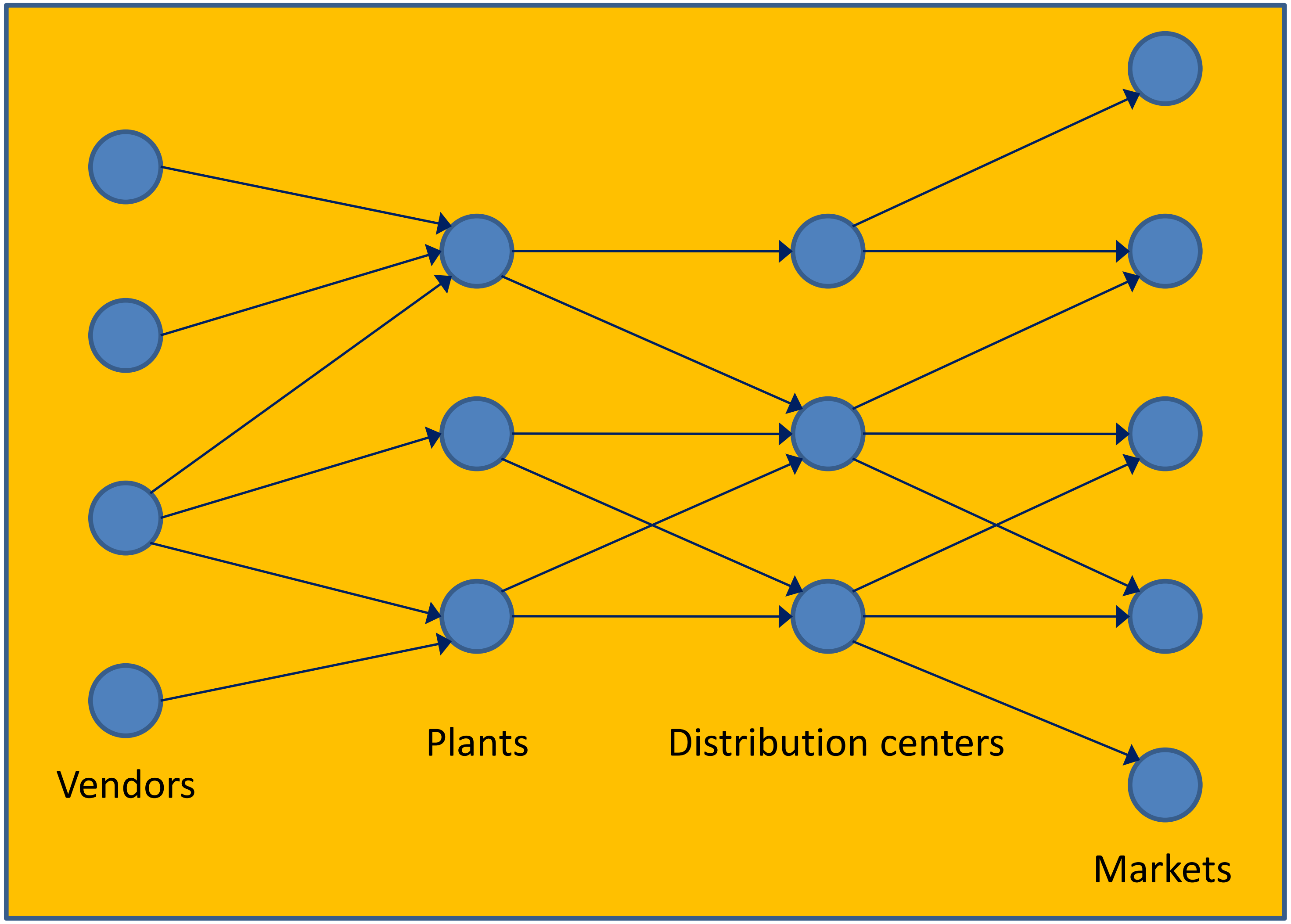
- The company's goal is to add value to its products as they pass through its supply chain and transport them to geographically dispersed markets
  - In the correct quantities
  - With the correct specifications
  - At the correct time
  - At a competitive cost

Facilities

- Plants
  - Manufacturing facilities
- Distribution centers (DCs), where products are
  - Received
  - Sorted
  - Put away in inventory
  - Picked from inventory
  - Dispatched
  - Not physically transformed
- There can exists hybrid facilities

1.1.1. SUPPLY CHAIN NETWORK

Network with four levels



Non-physical products

- The physical products may be unusual
  - Electrical energy
  - Industrial gas
  - …
- Telecommunications network could arguably be considered SC
- Some service companies (banks, insurance companies) operates value chain

1.1.2. INTEGRATED SUPPLY CHAIN PLANNING

Integrations

- Functional integration
- Spatial integration
- Intertemporal integration (hierarchical planning)

Functional integration

- Purchasing
- Manufacturing
- Transportation
- Warehousing
- Inventory management

Spatial integration

- Geographical dispersion
  - Vendors
  - Facilities
  - Markets

Intertemporal integration

- Strategic planning
  - Decisions over long-term planning horizons
    - Resource acquisition
    - Resource divestment
- Tactical planning
  - Decisions over medium-term planning horizons
    - Resource allocation
- Operational planning
  - Decisions over short-term planning horizons

Hierarchical planning

- Requires consistency and coherence among overlapping supply chain decisions at the various levels of planning
  - Efficient operations will not lead to superior profits if the firm’s products are being manufactured in plants with outdated technologies that are poorly located relative to the firm’s vendors and markets

Hierarchical planning

- The product’s supply chain must be optimized over it’s life cycle
  - Design
  - Introduction
  - Growth
  - Maturity
  - Retirement

Example

- Analysis of capital investment decisions in manufacturing equipment during the growth phase of a new product should take into account marketing decisions affecting products sales and gross revenues that may provide future returns sufficient to justify the investment

Collaboration

- Improved collaboration of activities across multiple companies sharing components of a supply chain is a concern of increasing interest and importance
  - Different interests
  - Other products, vendors and customers

1.1.3. OBJECTIVES OF SUPPLY CHAIN MANAGEMENT

Traditional objective

- Minimize total supply chain cost (fixed and given demand)
  - Raw material and other acquisition costs
  - Inbound transportation costs
  - Facility investment costs
  - Direct and indirect manufacturing costs
  - Direct and indirect distribution center costs
  - Inventory-holding costs
  - Interfacility transportation costs
  - Outbound transportation costs

Net revenues

- We might decide to examine only a portion of the company's entire supply chain and associated cost
- The firm does not want to minimize the total cost but maximize the net revenue
  - Net revenue = gross revenue – total cost
  - With fixed demands, the minimization of total cost equivalent of the maximization of net revenue

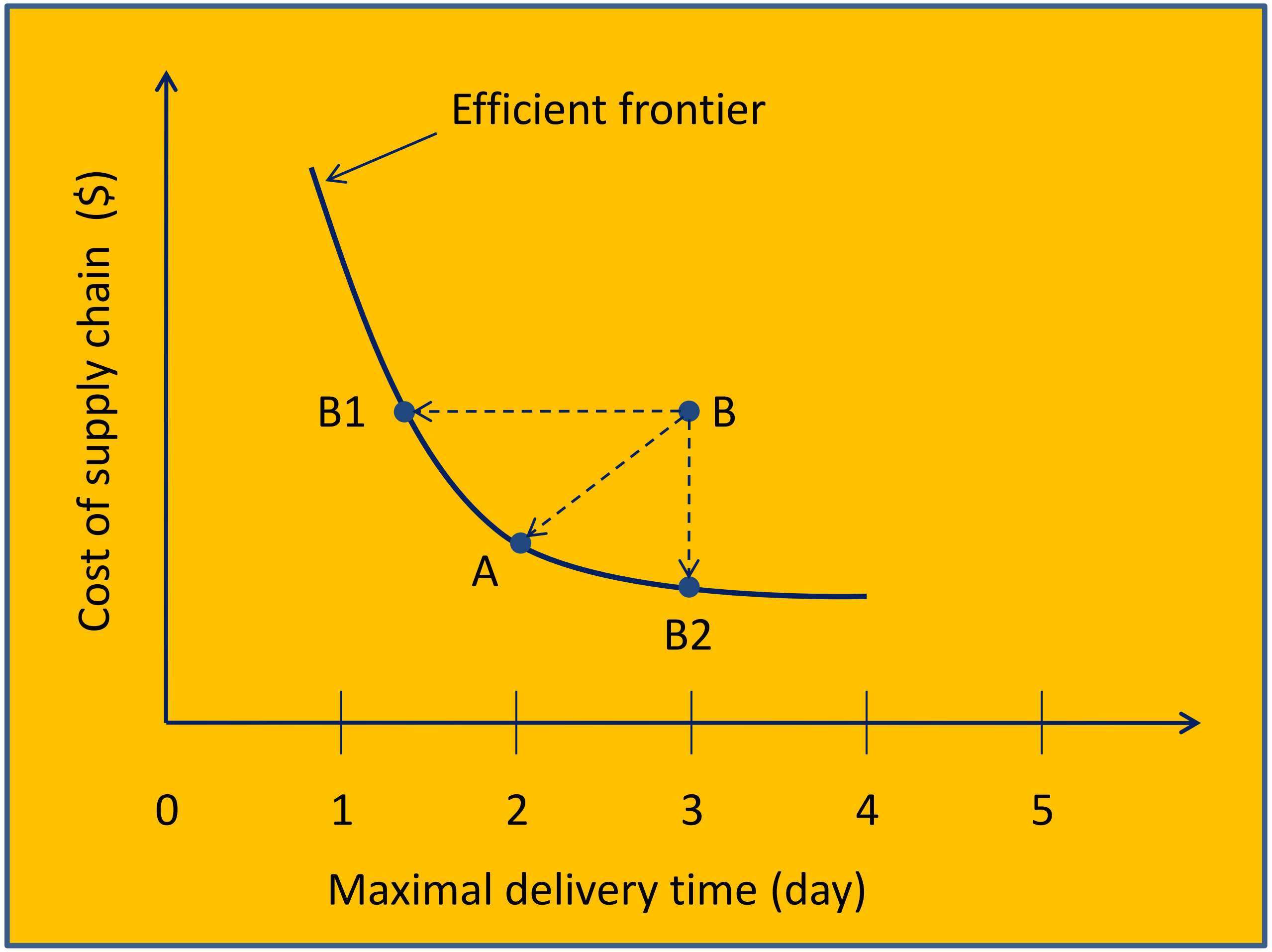
Decisions

- Objectives
  - Cost
  - Net revenue
  - Customer service
  - Product variety
  - Quality
  - Time
- Trade-off analysis

Example

- Objectives
  - Cost of supply chain
  - Longest delivery time
    - The maximal number of days allowed for delivery to customer
    - In range of 1-4 days
- Efficient frontier

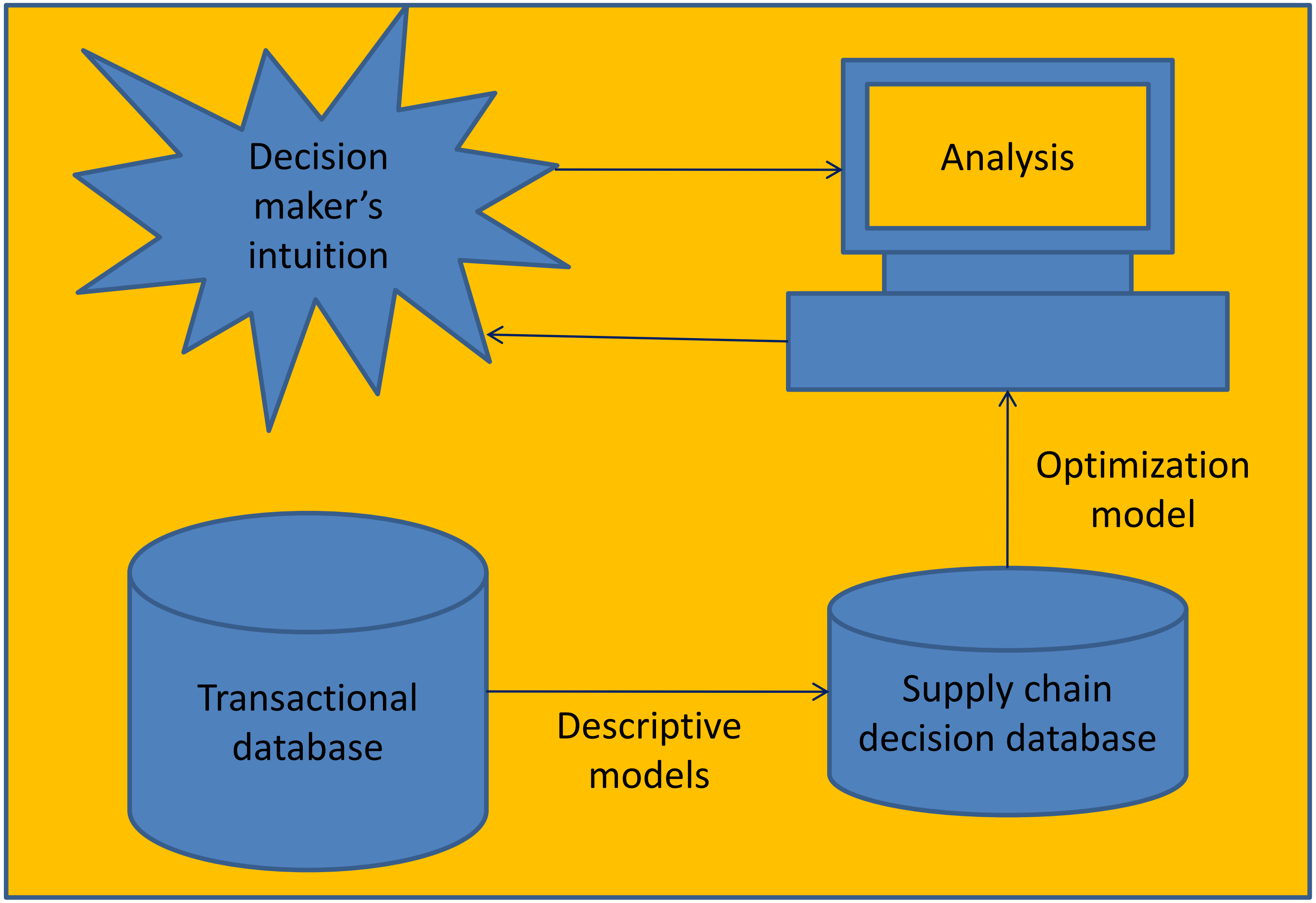
Example


Efficient frontier
Cost of supply chain ($)
B1
B
A
B2
0
1
2
3
4
5
Maximal delivery time (day)

1.2. OVERVIEW OF SUPPLY CHAIN MODELS AND MODELING SYSTEMS

Descriptive and optimization model

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088



Descriptive models

- Functional relationships in the company and the outside world
  - Forecasting models
    - Predict demand for the company's finished products, the cost of raw materials, … based on historical data
  - Cost relationships
    - Direct and indirect cost vary as function of cost drivers
  - Resource utilization relationships
    - How activities consume scarce resources
  - Simulation models
    - How the parts of the company's will operate over time

SC decision database

- High percent of the transactional data are not relevant for decisions
- Descriptive models are used to transform the remaining, relevant transactional data to data and relationship useful for decision making

Optimization model

- The data and the data relationships in the decision database are inputs to an optimization model
- The model integrates them in analyzing a given decision problem by seeking an optimal solution according to the manager’s preferences

Accuracy

- The solution of a model is not better than its input
  - "Garbage-in, garbage-out" problem
- Although some data are not yet accurate using approximate data is better than abandoning the analysis
- Accurate descriptive models are necessary but not sufficient for realizing effective decision making
  - For example, accurate demand forecast must be combined with other data in constructing a holistic optimization model to determine which plants should make which products to serve which distribution centers and markets

Realization

- A model conceptualized on paper must be realized by programs
  - Generation a computer-readable representation of it from input data
  - The results gleaned from the output of the algorithm must be reported in managerial terms

Managers ↔ modeling experts

- Most managers are not modeling experts
  - They can be deceived by mediocre models and methods

1.3. INCORPORATION CONCEPTS FROM MANAGEMENT DISCIPLINES

Introduction

- Models and modeling systems provide the supply chain manager with a framework for representing concepts from various management disciplines

1.3.1. STRATEGY FORMATION AND THEORY OF THE FIRM

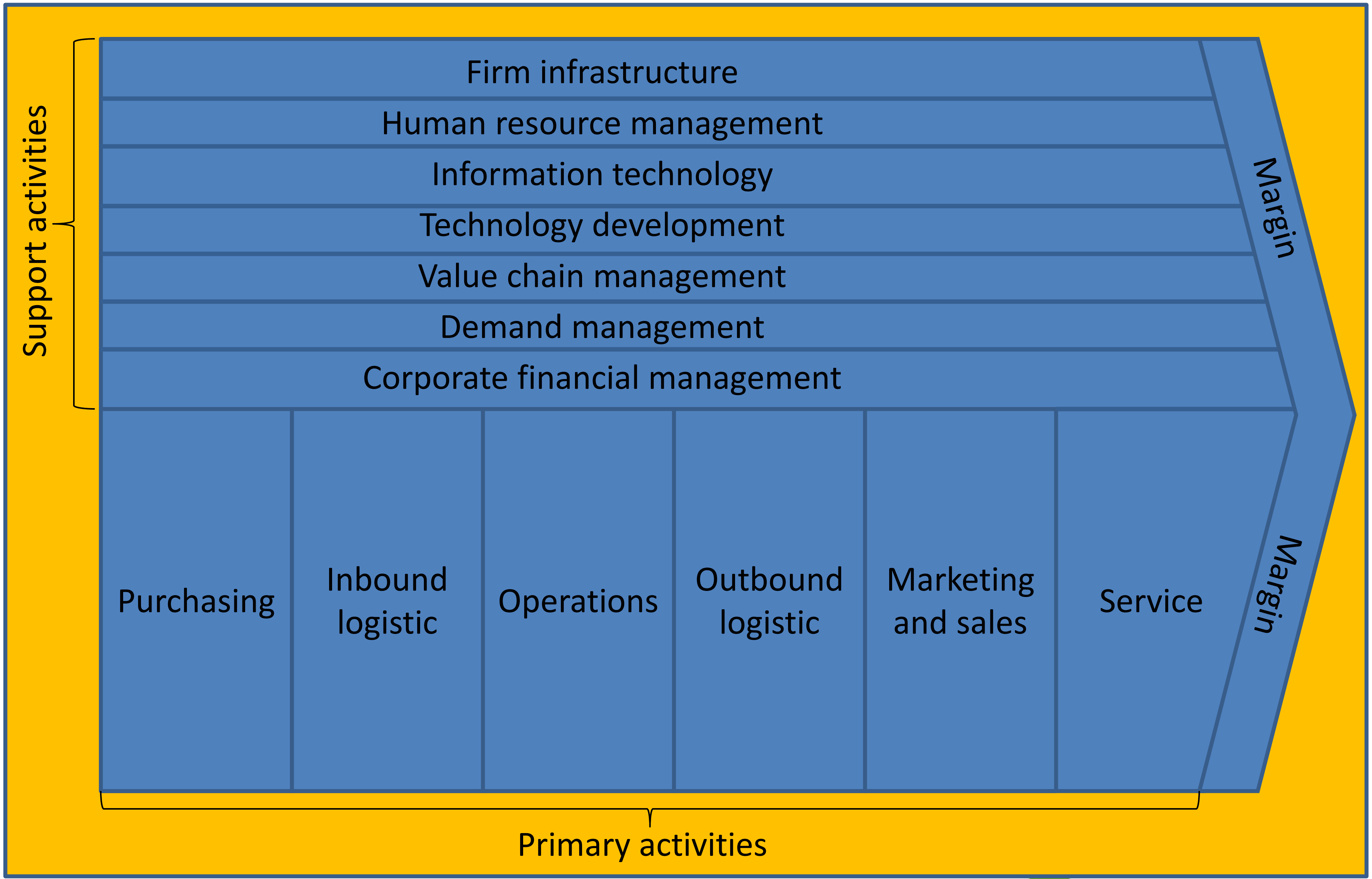
Value chain

- Every firm is a collection of its activities that are performed to design, produce, market, deliver, and support its product
- A firm's value chain and the way it performs individual activities are a reflection of its history, its strategy, its approach to implementing its strategy, and the underlying economics of the activities themselves

Supply chain

- The supply chain is a special case of the value chain for those companies that manufacture and/or distribute physical products
- The value chain has also been called the value-add chain to focus attention on the firm’s ultimate objective of adding value to its products or services at each stage in the chain

Value chain


Support activities
Firm infrastructure
Human resource management
Information technology
Technology development
Value chain management
Demand management
Corporate financial management
Margin
Purchasing
Inbound logistic
Operations
Outbound logistic
Marketing and sales
Service
Margin
Primary activities

Microeconomic models and theories

- Microeconomics, or industrial organization economics is concerned with the construction and interpretation of models that describe in mathematical terms how firm operate, expand, merge, and contract according to economic principles
  - Microeconomic model
- This model is highly relevant to the optimization model

Example

- A company that established a policy restricting the acquisition of a critical raw material to at least two vendors with no vendor providing less than 20% of the total volume
- Management has imposed the policy despite the fact that a particular vendor is offering to sell the raw material in unlimited quantities at significantly lower cost than other vendors
  - External policy constraint
- Optimization models can perform policy analysis pragmatically

Resource-based view

- Resource based models
  - Explicitly analyze the company’s resources in selecting plants to expand efficient resources while contracting inefficient resources
  - Explicitly address strategic uncertainties in demand when determining resource acquisition plans
  - Analyze the firm’s product line from a strategic perspective by simulation considering new product introduction, life-cycle management, and product retirement

1.3.2. LOGISTICS, PRODUCTION MANAGEMENT, AND INVENTORY MANAGEMENT

Introduction

- Till that the top-down, high-level strategic issues examined
- Logistic, production, and inventory management are concerned with managing supply chain operations from the bottom up

Logistics

- Logistics is concerned with managing
  - Transporting activities
  - Warehousing activities
  - Inventory-stocking activities

Transportation

- Transportation planning involves complex decisions
  - Transportation modes (car, truck, train, ship, plane, …)
  - Carrier selection
  - Vehicle scheduling
  - Vehicle routing
  - …

Importance of logistics

- Need to achieve integrated supply chain management
- Need to balance nonmonetary objective such as customer service against cost minimization
- Offer the lot quantity cost as a surrogate for manufacturing cost

Typical consumer product company

- Sales strategies for next year determined by marketing managers
- The plan is passed to manufacturing managers who are asked to develop an appropriate production strategy
- The joint marketing and manufacturing strategy is then passed to logistics managers who are given the responsibility of developing logistic strategy
  - The marketing and manufacturing data cannot be changed
  - The overall supply chain strategy may be significantly suboptimal

Production management

- Production planning varies significantly across industries
- Process manufacturing involves expensive capital equipment that is run continuously with infrequent changeovers
  - For example oil refineries, breweries
- Discrete-part manufacturing involves multiple-stage product lines with setups at each stage for intermediate products
  - For example automobiles, printed circuit boards

Scheduling

- Determine the timetable of production
  - Timing of setups, changeovers, production run length, …
- Schedule determines information for materials requirements planning
- At tactical and strategic planning levels timing details are less important
  - Multistage- and multiperiod-planning decisions

Inventory management

- A company may hold inventories of raw materials, parts, work-in-progress or finished products
- Inventories can serve to
  - Hedge against the uncertainties of supply and demand
  - Take advantage of economic of scale associated with manufacturing or acquiring products in large batches

Characterization

- Holding cost
- Short-age cost
- Demand distribution

1.3.3. DEMAND FORECASTING

Forecasting

- Demand forecasting refers to quantitative methods for predicting future demand for products sold by the company
- Forecasts are essential for the construction of a supply chain model
- Uncertainties in the demand forecast should be used in constructing multiple scenarios

Data mining

- Data mining is defined as "the process of exploration and analysis, by automatic or semiautomatic means, of large quantities of data in order to discover meaningful patterns and rules"
- It has been successfully applied to forecasting demand

Marketing science models

- Marketing science models relate forecasted sales to values decision variables
- The cost of marketing strategies involve the following significant decision variables
  - Price
  - Promotion
  - Advertising
  - Sales effort

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

1.3.4. OPERATION RESEARCH

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Operation research

- Operation research has been called the science and technology of decision making
- Scientific components for modeling decision problems
  - Ideas
  - Methods (algorithms)
- Technical components
  - Software
  - Hardware

1.4. INNOVATION IN INFORMATION TECHNOLOGY

Database

- The ERP’s transactional databases are the foundation from which supply chain modeling systems can be constructed and applied
- Supply chain modeling systems are critically needed to help management extract effective plans from these databases

Modeling system

- To emphasize the need for modeling systems, we highlight two serious problems involving data currently faced by managers

Data problem one

- There is an overabundance of transactional data for the purpose of managerial decision making
- Streaming of transactional data generates very large database
  - Automatic identification systems

Data problem two

- Managers do not know what the data imply about how to efficiently manage the activities, integrate their activities, and coordinate company activities

Technological solution

- Develop and deploy modeling systems for analyzing strategic, tactical, and operational decisions affecting the company’s supply chain

Organizational solution

- Expand company processes
  - The process expansion requirements needed to exploit insights provided by modeling systems are not yet well understood or appreciated by most managers
  - New type of jobs for planners and analysts that combine skills in IT with knowledge about business problems
- Revision managerial incentive scenes
  - Middle and upper-middle managers are encouraged to make decisions and pursue plans that serves to holistically optimize the company’s supply chain

2. INFORMATION TECHNOLOGY

Introduction

- Today’s IT enables the development of supply chain management modeling systems
- There is a need for analytical IT
  - Descriptive and optimization models
  - Data mining
  - Enterprise systems

2.1. ENTERPRISE RESOURCE PLANNING SYSTEMS

ERP systems

- An ERP system includes software and hardware that facilitate the creation and flow of transactional data in a company relating to
  - Manufacturing
  - Logistics
  - Finance
  - Sales
  - Human resources

ERP

- All business applications of the company are integrated in a uniform system environment
  - Centralized database
  - Common platform

Limitations of ERP systems

- Imposed conformity
  - Rigid requirements on data and processes
- Hidden costs
  - Training, integration, testing, customization, …
- Inability to employ software from multiple vendors
- Incompatibility of ERP systems across the supply chain
  - The company could not easily integrate supply chain databases with vendors and customers

2.2. E-COMMERCE

B2C

- Business-to-consumer (B2C) e-commerce is a method of retailing
- The consumer is in direct contact with companies offering physical products

Marketing and sales challenges

- Creating attractive graphics to the website
- Pricing products
- Identifying and exploiting demographics of website customers
- Acceptable and sustainable customer service criteria
- Devising strategies to retain customers
- Number and range of products
- Security payment

B2B

- Business-to-business (B2B) e-commerce
- The realized and potential impact of B2B on SCM is much larger than that for B2C
- Virtual supply chain among industries

Virtual supply chain

- Standardized definition and meanings of data
  - Shift product from a company to a second
  - Middleware between companies
- Establishment a level of intercompany coordination
  - Not share sensitive data
- Optimization modeling systems
  - Faster communication of data does not automatically lead to better decision making

Electronic procurement

- Seller-side sites
  - Suppliers place their catalogs and spec sheets on their website
- Buyer-side sites
  - Buyers have installed software allowing to read and standardize vendor catalogs
- Third-party sites
  - Neutral sites works as marketplaces where buyers and sellers can link up
  - These sites are usually specific to certain industries

Procurement

- Direct procurements over the Internet by manufacturing firms may be complicated
  - Customization of some parts and components
- More flexible software solutions are needed
- For example, a trade association in the automotive industry commissioned the implementation of a standard-based network, the Automotive Network eXchange (ANX)
  - The suppliers would be required to be connected to and use this network

2.3. RADIO-FREQUENCY IDENTIFICATION

Introduction

- The radio-frequency identification (RFID) was invented by the British during World War II to allow their aircraft to be distinguished from enemy aircraft
- Recently retailers, manufacturers and distributors became interested is using RFID tags to track consumer products

Components of FRID technology

- Small tag containing microchip
  - 96 bits electronic product code
- Scanner
  - Activates the tag to induce it to send the data that it carries
  - Not require a line to sight to read
- Networking hardware and software
  - Links the scanner to databases

Benefits of using RFID

- Automatically updating inventory levels
- Speeding up put-away and picking of items from inventory
- Displaying the current location of an item
- Sending an alert when an item is no longer visible
- Enhancing security at ports
- Speeding up checkouts of consumer products

2.4. COMPARISON OF TRANSACTIONAL AND ANALYTICAL IT

Introduction

- In many companies the scope and the flexibility of installed ERP systems have been less than desired or expected
- New ERP systems that are modular and web enabled may lead to significant improvements
  - Advantages cannot be gained simply through faster communication data

Transactional IT

- It is concerned with
  - Acquiring and managing raw data about SC
  - The compilation of reports
- The source of data
  - Internal
    - Ledger system
    - Manufacturing process-control system
  - External
    - Orders (telephone)
    - Trucking rates of a common carrier (over the Internet)

Analytical IT

- Evaluates SC problems with using models
- Descriptive models
  - Forecasting
  - Management accounting models (costs, constraints, …)
- Optimization models
  - Describe the space of supply chain options
  - Decision database

Time frame addressed

- Transactional IT
  - Past and present
- Analytical IT
  - Future

Purpose

- Transactional IT
  - Communication
- Analytical IT
  - Forecasting and decision making
    - Uncertainties

Business scope

- Transactional IT
  - Myopic
- Analytical IT
  - Hierarchical
    - Hierarchy of decisions (operational, tactical, strategic)

Nature of databases

- Transactional IT
  - Raw and lightly transformed objective data
    - For example, average costs for shipping full truckloads last month
- Analytical IT
  - Raw, moderate transformed, and heavily transformed data that are both objective and judgmental
    - For example, components of the cost of a product
    - Judgmental data may not be justified purely by cost
      - For example, risk, limiting distance in transportation

Response time for queries

- Transactional IT
  - Real time
- Analytical IT
  - Real time and batch processing

Implication for business process change

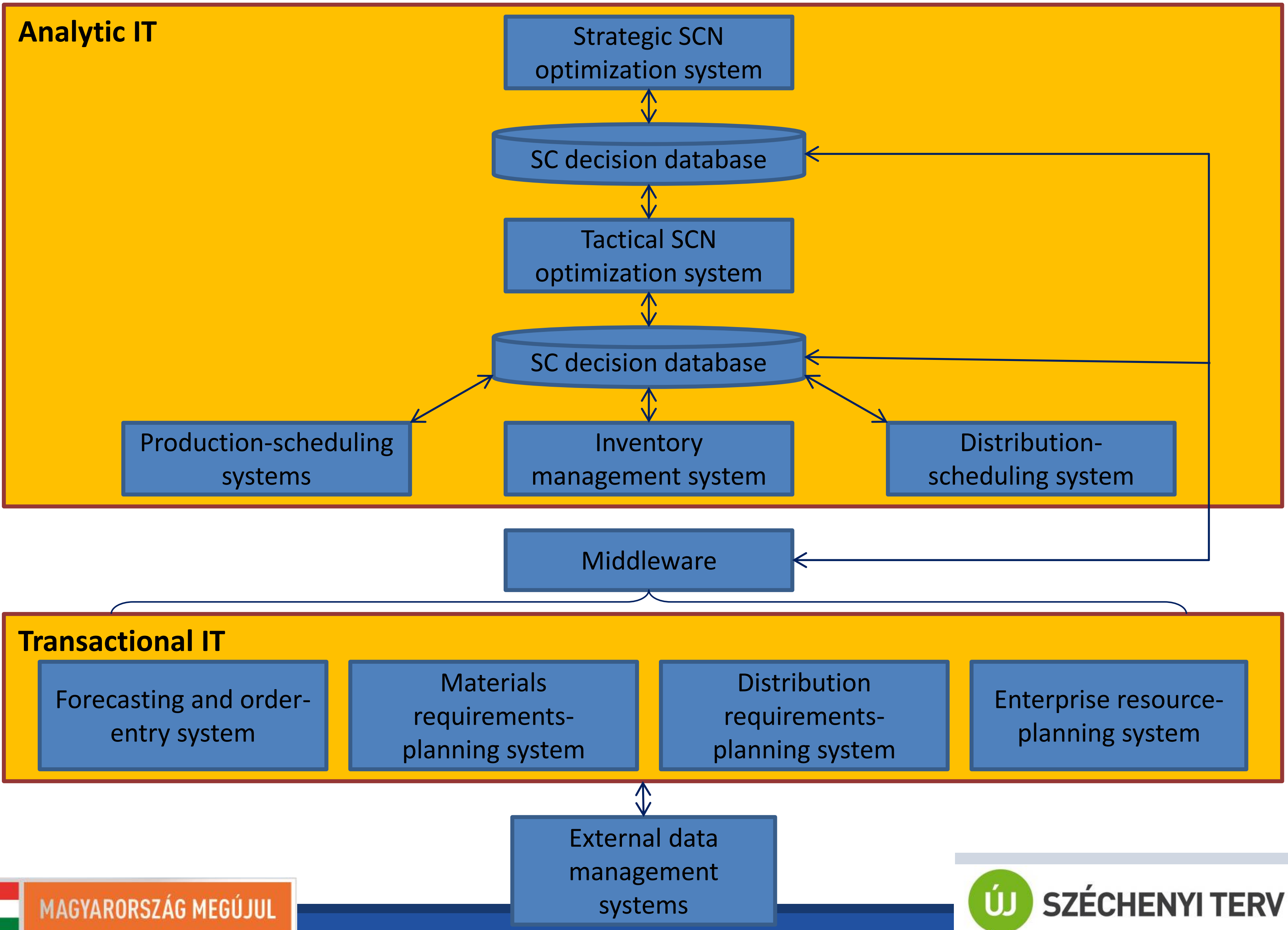
- Transactional IT
  - Substitute for or eliminating inefficient human effort
- Analytical IT
  - Coordinate overlapping management decisions
    - Allows supply chain decisions to be integrated across managerial responsibilities across levels of planning

2.5. HIERARCHY OF SUPPLY CHAIN MANAGEMENT

Introduction

- The supply chain hierarchy is hypothetical
  - To the best of our knowledge, no company has implemented and integrated all modeling systems, supporting programs and adatbases

Supply chain system hierarchy


Analytic IT
Strategic SCN optimization system
SC decision database
Tactical SCN optimization system
SC decision database
Production-scheduling systems
Inventory management system
Distribution-scheduling system
Middleware
Transactional IT
Forecasting and order-entry system
Materials requirements-planning system
Distribution requirements-planning system
Enterprise resource-planning system
External data management systems

2.5.1. COMPONENTS OF THE SUPPLY CHAIN SYSTEMS HIERARCHY

ERP system

- Manages the company’s transactional data
- This system standardizes the company’s data and information systems
- Newer ERP systems have been expanded to inculde the other three transactional systems

DRP system

- The distribution requirements planning (DRP) system analyses product throughout the logistics network
  - Begins with forecast or finished products
  - Using inventory management data
  - Schedules inbound, interfacility and outbound shipments

MRP system

- Material requirement planning (MRP) system analyzes products at all stages of manufacturing
  - One MRP system at each manufacturing site
  - Begins with a master production schedule
  - Develops net requirements of raw materials (and intermediate products to b manufactured) to meet demands

Forecasting and order-entry system

- Order-entry system
  - Keeps track of current orders
    - Quantities ordered, delivery locations, promised delivery dates, …
- Forecasting
  - Short-term forecasts for finished products

Middleware

- Links the transactional and analytic IT
- Relevant data from transactional database has to be extracted to decision database
  - Analytic IT needs 10 to 20% data of the total volume

Distribution-scheduling system

- Manages transportation movements
  - Choose from transportation options
  - Choose from transportation vehicles
  - …
- Scheduling of distribution centers

Inventory management systems

- Keeps track of inventories and replenishment policies
  - Raw materials
  - Work-in-process and finished goods
  - Machines
- Centralized inventory management is an ideal that most companies have not realized
  - Inconsistency

Production-scheduling systems

- Located et each plant
- Operational decisions
  - Sequencing of orders on a machine
  - Timing of major and minor changeovers
  - Management of work-in-process inventories
- Minimize avoidable short-term costs while satisfying customer requirements

Tactical supply chain network

- Used each month to determine an integrated plan over the next 6 to 12 month
  - Supply
  - Manufacturing
  - Distribution
  - Inventory
- Aggregation
  - Similar materials into product families
  - Markets into market zones

Strategic supply chain network

- Used to analyze resource acquisition and other strategic decisions

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

2.5.2. FREQUENCY OF ANALYSIS, PLANNING TIMES AND RUN TIMES

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- Frequency of analysis
  - The number of times each year, month, week, or day that managers and planners use the system
- Planning time
  - How long it takes to complete analysis of planning problems with the system each time it is used
- Run time
  - Batch time required for each run of the system

Times

	Planning horizon	Model structure	Objective function	Frequency of analysis	Planning time	Run time
Strategic planning	1-5 years	Yearly snapshots	Net revenue	Once a year	2 weeks- 2 months	5-60 minutes
Tactical planning	6-12 moths	Months	Total cost	Once a month	2-4 days	20-120 minutes
Production-schedule	7-30 days	7-90 days	Production cost	Once a day	30 minutes	5-20 minutes
Inventory management	30 days	-	Inventory cost	Continuous	Instantaneous	Instantaneous
Distribution-schedule	7-30 days	7-90 days	Distribution cost	Once a day	30 minutes	5-20 minutes
Middleware	-	Range of descriptive model	-	Continuous	2-4 days	1-60 minutes
Forecasting and order-entry	-	-	-	Continuous	-	Instantaneous
MRP	7-30 days	7-30 days	-	Once a week	1-3 hours	60 minutes
DRP	7-30 days	7-30 days	-	Once a week	1-3 hours	60 minutes
ERP	-	-	-	Continuous	-	-

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

Modeling the supply chain

II. Models

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Content

- Linear programming
- Mixed integer programming
- Optimization methodology for operational planning problems

3. LINEAR PROGRAMMING

Introduction

- Linear programming (LP) models and methods for optimizing them played a central role in all types of supply chain applications

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

3.1. EXAMPLES

MAGYARORSZÁG MEGÚJUL

ÚJ SZÉCHENYI TERV

3.1.1. RESOURCE ALLOCATION MODEL

Introduction

- In a supply chain network, many activities compete for resources
  - Machine capacities
  - Inventory
- The available quantities of some resources may be insufficient to accommodate all the demands

Example 3.1

- A weekly resource allocation problem for the production manager of Ajax Computer Company
- Ajax sells three types of computers
  - Desktop computer (Alpha) – $350
  - Laptop (Beta) – $470
  - Workstation (Gamma) – $610
- We assume that all production during the week can and will be sold immediately

Example 3.1

- Two test equipment
  - A-line (Alpha and Beta) – 120 hours/week, 1 hour/PC
  - C-line (Gamma) – 48 hours/week, 1 hour/PC
- Human resource
  - Alfa – 10 labor-hours/PC
  - Beta – 15 labor-hours/PC
  - Gamma – 20 labor-hours/PC

Decision variables

- M_A : number of Alphas to be assembled during the week
- M_B : number of Betas to be assembled during the week
- M_C : number of Gammas to be assembled during the week

LP model

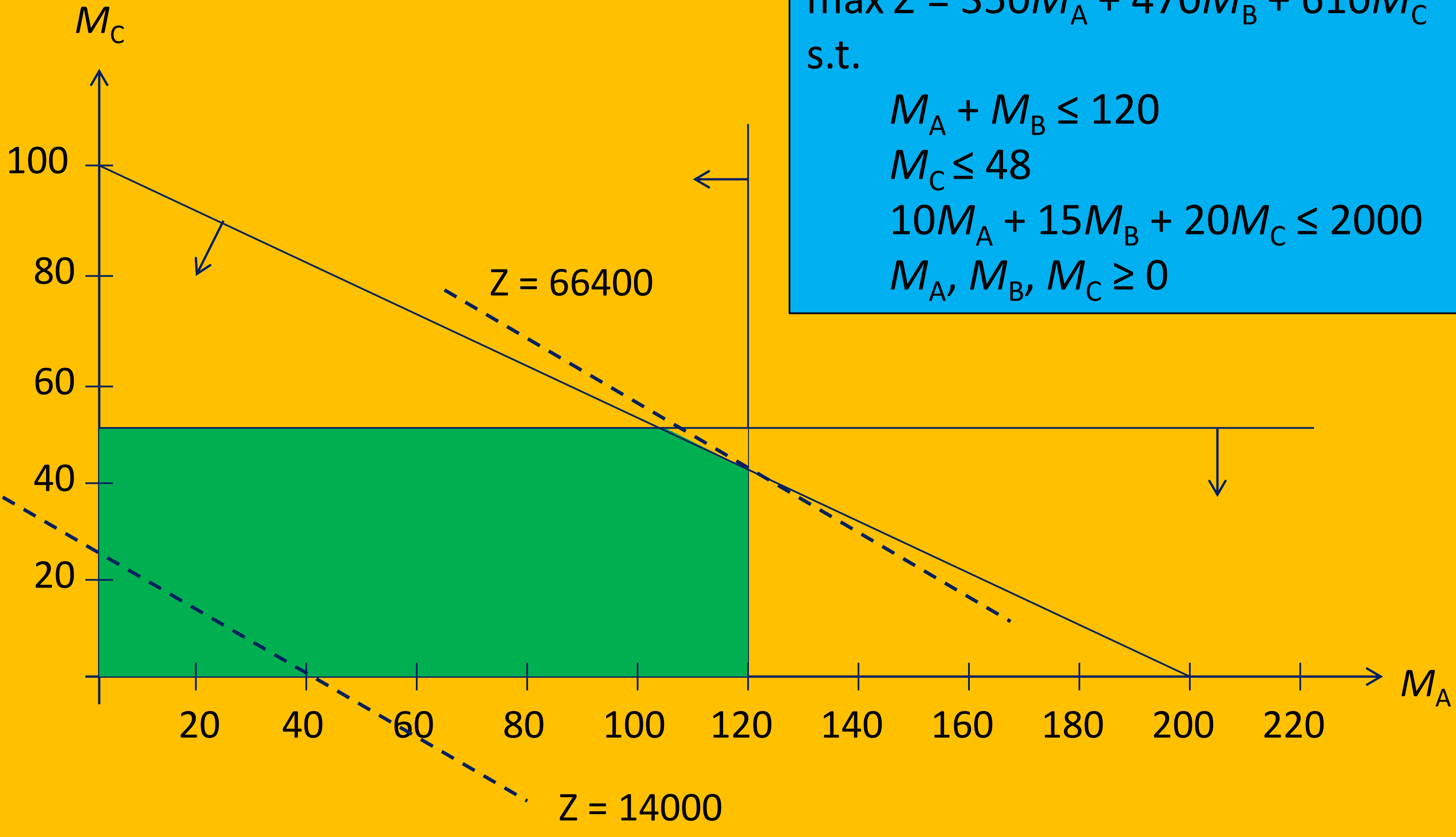
- Objective function
  - Max $Z = 350M_A + 470M_B + 610M_C$
- Constraints
  - $M_A + M_B \leq 120$
  - $M_C \leq 48$
  - $10M_A + 15M_B + 20M_C \leq 2000$

Optimal solution

- $Z^* = \$66{,}400$
- $M_A^* = 120$
- $M_B^* = 0$
- $M_C^* = 40$

- Labor availability are satisfied as equation
- A-line test capacity constraint are satisfied as equation
- C-line test capacity constraints is satisfied as a strict inequality
  - It has free capacity

Search space ($M_B = 0$)



$$\max Z = 350M_A + 470M_B + 610M_C$$
s.t.
$$M_A + M_B \leq 120$$
$$M_C \leq 48$$
$$10M_A + 15M_B + 20M_C \leq 2000$$
$$M_A, M_B, M_C \geq 0$$

Solution executed by ZIMPL

- ZIMPL is a language to translate the mathematical model of a problem into a linear or (mixed-) integer mathematical program expressed in .lp or .mps file format which can be read and solved by a LP or MILP solver
- ZIMPL is a command line program written in plain C and released under GNU LGLP

Solution executed by ZIMPL

- It has been tested to compile under Linux/Intel, Solaris, Tru64, HPUX, IRIX, AIX and MacOS-X
- It has even been successfully compiled for Windows using MinGW and GCC as a cross compiler and also directly using VisualStudio 2010

Other possibilities for ZIMPL

- The model can be more readable and more flexible using sets and parameters

Sets in ZIMPL

- Sets consist of (a finite number of tuples)
- A tuple is an ordered vector of fixed dimension where each component is either a number or a string
  - The tuple are in examples are one dimensional
- Each tuple is unique in a set
- The type of the nth component for all tuples of a set must be the same

Sets in ZIMPL

- Sets can be defined with the set statement
  - It consist of the
    - Keyword `set`
    - The name of the set
    - An assignment operator `:=`
    - A valid set expression
- For example
  - `set A := {1, 2, 8};`
  - `set computers := {"Alpha", "Beta", "Gamma"};`

Parameters in ZIMPL

- Parameters are the way to define constants in ZIMPL
- They can be defined with or without an index set
  - Without index, the parameter is just a single value or string
  - For indexed parameters, there is one value for each member of the set
    - It is possible to declare a default value

Parameters in ZIMPL

- Declaration of parameters
  - The keyword `param`
  - The name of the parameter
  - Optionally the name of the index set in square brackets
  - Assignment operator `:=`
  - The list of pairs
    - The first element of pairs is a tuple form the index set
    - The second element is the value of the parameter for this index

Parameters in ZIMPL

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

- For example
  - **set** computers := {"Alpha", "Beta", "Gamma"};
  - **set** resources := {"A-line", "C-line", "manpower"};
  - **param** netvalue[computers] := <"Alpha"> 350, <"Beta"> 470, <"Gamma"> 610;
  - **param** availability[resources] := <"A-line"> 120, <"C-line"> 48, <"manpower"> 2000;

Parameter tables in ZIMPL

- It is possible to initialize multi-dimensional indexed parameters from tables
  - Especially useful for two-dimensional parameters
- The data is put in a table structure with `|` signs on each margins
- A headline with column indexes has to be added
- One index for each row is needed

Parameter tables in ZIMPL

- For example
  - **set** computers := {"Alpha", "Beta", "Gamma"};
  - **set** resources := {"A-line", "C-line", "manpower"};
  - **param** usage[computers*resources] :=

```
           |"A-line", "C-line", "manpower"|
|"Alpha"|    1,         0,          10      |
|"Beta" |    1,         0,          15      |
|"Gamma"|    0,         1,          20      |;
```

Variables in ZIMPL

- Like parameters, variables can be indexed
- A variable has to be one out of three possible types
  - `real` (continuous), `binary`, `integer`
    - The default type is `real`
- Variables may have lower and upper bounds
  - Defaults are zero as lower and infinity as upper bound

Variables in ZIMPL

- For example
  - `var Produced_Alpha >=0;`
  - `set computers := {"Alpha", "Beta", "Gamma"};`
  - `var Produced[computers] >=0;`
  - `var Produced2[computers] integer;`

`sum` expression in ZIMPL

- Sums are stated in forms
  - **`sum`** `index` **`do`** `term`
  - **`sum`** `index : term`
- The general form of `index` is
  - `tuple` **`in`** `set` **`with`** `boolean-expression`
  - The number of components in the tuple and the members of the `set` must match
  - The `with` part is optional
  - The `set` can be any expression giving a set

`sum` expression in ZIMPL

- For example
  - **set** computers := {"Alpha", "Beta", "Gamma"};
  - **param** netvalue[computers] := <"Alpha"> 350, <"Beta"> 470, <"Gamma"> 610;
  - **sum** <c> **in** computers : netvalue[c];

`forall` statement in ZIMPL

- The general forms are
  - **`forall`** `index` **`do`** `term`
- The general form of `index` equals that of `sum-expression`

forall statement in ZIMPL

- For example

```
forall <r> in resources do
sum <c> in computers : usage[c,r] *
Produced[c] <= availability[r];
```

Objective in ZIMPL

- There must be at most one objective statement in a model
  - The objective can be either `minimize` or `maximize`
  - A name
  - A colon
  - A linear term expressing the objective function

Objective in ZIMPL

- For example
  - **maximize** Income_in_Dollars: 350 * Produced_Alpha + 470 * Produced_Beta + 610 * Produced_Gamma;
  - **maximize** Income_in_Dollars: **sum** <c> **in** computers : netvalue[c] * Produced[c];

Constraints in ZIMPL

- The general format for a constraint is
  - `**subto** name: term sense term`
  - The `name` can be any string starting with a letter
  - `term` is define as in the objective
  - `sense` is one of `<=, >=, ==`
- Many constraints can be generated with one statement by the use of the `forall` instruction

Constraints in ZIMPL

- For example
  - **subto** Max_Capacity_of_Production_Line1: Produced_Alpha + Produced_Beta <= 120;
  - **subto** Resource_constraints: **forall** <r> **in** resources **do** **sum** <c> **in** computers : usage[c,r] * Produced[c] <= availability[r];

Example 3.1 by ZIMPL

```
# Example 3.1
# A weekly resource allocation problem for the production
   manager of Ajax Computer Company

var Produced_Alpha >=0;
var Produced_Beta >=0;
var Produced_Gamma >=0;

maximize Income_in_Dollars:
350 * Produced_Alpha + 470 * Produced_Beta + 610 *
   Produced_Gamma;
```

Example 3.1 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta <= 120;

subto Max_Capacity_of_Production_Line2:
Produced_Gamma <= 48;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000;
```

Example 3.1 by ZIMPL results

```
Optimal - objective value                 -66400
      0 Produced_Alpha                      120
      1 Produced_Beta                         0
      2 Produced_Gamma                       40
```

Example 3.1 by ZIMPL ("set" model)

```
# Example 3.1
# A weekly resource allocation problem for the production
  manager of Ajax Computer Company

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};

#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta"> 470,
   <"Gamma"> 610;
param availability[resources] := <"A-line"> 120, <"C-line">
   48, <"manpower"> 2000;
param usage[computers*resources] :=
           |"A-line", "C-line", "manpower"|
   |"Alpha"|   1,         0,          10     |
   |"Beta" |   1,         0,          15     |
   |"Gamma"|   0,         1,          20     |;
```

Example 3.1 by ZIMPL ("set" model) cont.

```
# variables
var Produced[computers] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c];

#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * Produced[c] <=
   availability[r];
```

Example 3.1 by ZIMPL ("set" model) cont.

- The solution of this model is the same as the previous but it is more easy to modify
- Not needed to understand the model itself to change for example the net value of a computer
- Easy to add a new product without modifying the model

Example 3.2

- Assume that Alphas can also be tested on the C-line test equipment
  - 1.5 hours/PC
- New decision variables
  - M_{A1} : number of Alphas to be assembled during the week and tested on A-line
  - M_{A2} : number of Alphas to be assembled during the week and tested on C-line

LP model

max $Z = 350M_A + 470M_B + 610M_C$

max $Z = 350M_A + 470M_B + 610M_C$
s.t.
$M_A + M_B \leq 120$
$M_C \leq 48$
$10M_A + 15M_B + 20M_C \leq 2000$
$M_A, M_B, M_C \geq 0$

s.t.

$M_{A1} + M_B \leq 120$

$1.5M_{A2} + M_C \leq 48$

$10M_{A1} + 10M_{A2} + 15M_B + 20M_C \leq 2000$

$M_A - M_{A1} - M_{A2} = 0$

$M_A, M_{A1}, M_{A2}, M_B, M_C \geq 0$

Optimal solution

- $Z^* = \$66{,}760$
- $M_A^* = 128$
- $M_{A1}^* = 120$
- $M_{A2}^* = 8$
- $M_B^* = 0$
- $M_C^* = 36$

Example 3.2 by ZIMPL

```
# Example 3.2
# Like Example 3.1 but Alphas can also be tested on the C-
   line

var Produced_Alpha_Line1 >=0;
var Produced_Alpha_Line2 >=0;
var Produced_Beta >=0;
var Produced_Gamma >=0;

maximize Income_in_Dollars:
350 * (Produced_Alpha_Line1 + Produced_Alpha_Line2) + 470 *
   Produced_Beta + 610 * Produced_Gamma;
```

Example 3.2 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha_Line1 + Produced_Beta <= 120;

subto Max_Capacity_of_Production_Line2:
 1.5 * Produced_Alpha_Line2 + Produced_Gamma <= 48;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha_Line1 + 10 * Produced_Alpha_Line2 + 15 *
   Produced_Beta + 20 * Produced_Gamma <= 2000;
```

Example 3.2 by ZIMPL results

```
Optimal - objective value              -66760
      0 Produced_Alpha_Line1             120
      1 Produced_Alpha_Line2               8
      2 Produced_Beta                      0
      3 Produced_Gamma                    36
```

3.1.2. INFEASIBLE AND UNBOUNDED MODELS

Introduction

- The model is infeasible if there are no feasible solutions
  - For example, if a constraints requiring at least 50 Gammas for a week when the C-line capacity is 48
- The model is feasible but the objective function can be unbounded

Example 3.3

- Suppose that Ajax
  - Can outsource A-line testing equipment at $40/hour in unlimited quantities
  - Can hire additional labor at $30/hour in unlimited quantities

LP model

- New decision variables
  - E_A : quantity of outsourced A-line test hours
  - E_L : quantity of rented labor hours
- Modified constraints
  - $M_A + M_B \leq 120 + E_A$
  - $10M_A + 15M_B + 20M_C \leq 2000 + E_L$

Solution

- Because the net profit for each Alpha is $350 but would only cost $340 using extra resources, Ajax can achieve an unbounded net revenue
- This situation of "buying low and selling high" to make an unbounded profit is called arbitrage by finance theorists

- Marketing and sales department could not sell an unlimited number of Alphas
- Production managers would be limited the extra A-line capacity and labor

Example 3.3 by ZIMPL

```
# Example 3.3
# Like Example 3.1 but can outsource A-line testing
   equipment
# and can hire additional labor

var Produced_Alpha >=0;
var Produced_Beta >=0;
var Produced_Gamma >=0;
var Bought_Capacity_of_Production_Line1;
var Bought_Capacity_of_Man_Power;

maximize Income_in_Dollars:
350 * Produced_Alpha + 470 * Produced_Beta + 610 *
   Produced_Gamma - 40 * Bought_Capacity_of_Production_Line1
   - 30 * Bought_Capacity_of_Man_Power;
```

Example 3.3 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta <= 120 +
   Bought_Capacity_of_Production_Line1;

subto Max_Capacity_of_Production_Line2:
 Produced_Gamma <= 48;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000 + Bought_Capacity_of_Man_Power;
```

Example 3.3 by ZIMPL results

```
Unbounded - objective value                 0
   0 Produced_Alpha                            0
   1 Produced_Beta                             0
   2 Produced_Gamma                            0
   3 Bought_Capacity_of_Production_Line1      0
   4 Bought_Capacity_of_Man_Power             0
```

Modifying the "set" model

- New parameter for the cost of outsourcing
- New variables for the value of outsourcing
  - Because of using sets, the upper bound of outsourcing C-line has to set zero or the cost of it set to a big number
- Modification of the cost function
- Modification of constraints

Example 3.3 by ZIMPL ("set" model)

```
# Example 3.3
# Like Example 3.1 but can outsource A-line testing
   equipment and can hire additional labor

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};

#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta"> 470,
   <"Gamma"> 610;
param availability[resources] := <"A-line"> 120, <"C-line">
   48, <"manpower"> 2000;
param usage[computers*resources] :=
             |"A-line", "C-line", "manpower"|
   |"Alpha"|   1,         0,           10     |
   |"Beta" |   1,         0,           15     |
   |"Gamma"|   0,         1,           20     |;
param outsourcingCost[resources] := <"A-line"> 40,
   <"manpower"> 30 default 999999;
```

Example 3.3 by ZIMPL ("set" model) cont.

```
# variables
var Produced[computers] >=0;
var Outsourcing[resources] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c]
   - sum <r> in resources : outsourcingCost[r] *
   Outsourcing[r];

#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * Produced[c] <=
   availability[r] + Outsourcing[r];
```

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

3.1.3. MULTIPERIOD RESOURCE ALLOCATION MODEL

MAGYARORSZÁG MEGÚJUL

ÚJ SZÉCHENYI TERV

Introduction

- SC planning is a dynamic process because decisions made in this period are linked to decisions that will be made in later periods
- For example, Inventories of raw materials, intermediate products and finished goods play a central role in optimization

Example 3.4

- High level decisions
  - The organization can sell 120 units of Alphas next week (see example 3.1) → Beta must be included in Ajax's production line
  - Sales forecast in 4 weeks time horizon (R_{pt}, A_{pt})

	Week 1	Week 2	Week 3	Week 4
Alpha	[20, 60]	[20, 80]	[20, 120]	[20, 140]
Beta	[20, 40]	[20, 40]	[20, 40]	[20, 40]
Gamma	[20, 50]	[20, 40]	[20, 30]	[20, 70]

Decisions

- In each week for each product
  - How much to produce?
  - How much to sell?
  - How much to store in inventory?
- Decision variables
  - M_{pt} : assembly and testing of product p in week t
  - S_{pt} : sales of product p in week t
  - I_{pt} : inventory of product p at the end of the week t (or inventory at the beginning of week $t + 1$)

Sales

- The sales of each product is more than the minimal and less than the maximal value
  - $R_{pt} \leq S_{pt} \leq A_{pt}$

Inventory data

- Carrying costs
  - Alpha : \$9/week
  - Beta : \$10/week
  - Gamma : \$18/week
- Initial inventory
  - Alpha : 22 (I_{A0})
  - Beta : 42 (I_{B0})
  - Gamma : 36 (I_{C0})

Inventory

- The following balance equations are satisfy for each week
  - Ending inventory in week *t* = ending inventory in week *t* – 1 + production in week *t* – sales in week *t*
  - $I_{pt} = I_{p,t\text{-}1} + M_{pt} - S_{pt}$

Model

$$\max Z = 350(S_{A1} + S_{A2} + S_{A3} + S_{A4}) + 470(S_{B1} + S_{B2} + S_{B3} + S_{B4}) + 610(S_{C1} + S_{C2} + S_{C3} + S_{C4}) - 9(I_{A1} + I_{A2} + I_{A3} + I_{A4}) - 10(I_{B1} + I_{B2} + I_{B3} + I_{B4}) - 18(I_{C1} + I_{C2} + I_{C3} + I_{C4})$$

s.t.

$$M_{At} + M_{Bt} \le 120 \qquad (t = 1, 2, 3, 4)$$

$$M_{Ct} \le 48 \qquad (t = 1, 2, 3, 4)$$

$$10M_{At} + 15M_{Bt} + 20M_{Ct} \le 2000 \qquad (t = 1, 2, 3, 4)$$

$$I_{pt} = I_{p,t\text{-}1} + M_{pt} - S_{pt} \quad (t = 1, 2, 3, 4;\ p = A, B, C)$$

$$M_{At}, M_{Bt}, M_{Ct} \ge 0 \qquad (t = 1, 2, 3, 4)$$

$$R_{pt} \le S_{pt} \le A_{pt} \qquad (t = 1, 2, 3, 4;\ p = A, B, C)$$

Example 3.4 by ZIMPL

```
# Example 3.4
# Like Example 3.1 but the organization can sell 120 units
   of Alphas next week
# and Beta has to be included in Ajax's production line

var Produced_A_W1 integer >=0;
var Produced_A_W2 integer >=0;
var Produced_A_W3 integer >=0;
var Produced_A_W4 integer >=0;
var Produced_B_W1 integer >=1;
var Produced_B_W2 integer >=1;
var Produced_B_W3 integer >=1;
var Produced_B_W4 integer >=1;
var Produced_C_W1 integer >=0;
var Produced_C_W2 integer >=0;
var Produced_C_W3 integer >=0;
var Produced_C_W4 integer >=0;
```

Example 3.4 by ZIMPL cont.

```
var Sold_A_W1 integer >=20 <=60;
var Sold_A_W2 integer >=20 <=80;
var Sold_A_W3 integer >=20 <=120;
var Sold_A_W4 integer >=20 <=140;
var Sold_B_W1 integer >=20 <=40;
var Sold_B_W2 integer >=20 <=40;
var Sold_B_W3 integer >=20 <=40;
var Sold_B_W4 integer >=20 <=40;
var Sold_C_W1 integer >=20 <=50;
var Sold_C_W2 integer >=20 <=40;
var Sold_C_W3 integer >=20 <=30;
var Sold_C_W4 integer >=20 <=70;
```

Example 3.4 by ZIMPL cont.

```
var Stock_A_W1 integer >=0;
var Stock_A_W2 integer >=0;
var Stock_A_W3 integer >=0;
var Stock_A_W4 integer >=0;
var Stock_B_W1 integer >=0;
var Stock_B_W2 integer >=0;
var Stock_B_W3 integer >=0;
var Stock_B_W4 integer >=0;
var Stock_C_W1 integer >=0;
var Stock_C_W2 integer >=0;
var Stock_C_W3 integer >=0;
var Stock_C_W4 integer >=0;
```

Example 3.4 by ZIMPL cont.

```
maximize Income_in_Dollars:
350 * (Sold_A_W1 + Sold_A_W2 + Sold_A_W3 + Sold_A_W4) + 470
   * (Sold_B_W1 + Sold_B_W2 + Sold_B_W3 + Sold_B_W4) + 610 *
   (Sold_C_W1 + Sold_C_W2 + Sold_C_W3 + Sold_C_W4) - 9 *
   (Stock_A_W1 + Stock_A_W2 + Stock_A_W3 + Stock_A_W4) - 10
   * (Stock_B_W1 + Stock_B_W2 + Stock_B_W3 + Stock_B_W4) -
   18 * (Stock_C_W1 + Stock_C_W2 + Stock_C_W3 + Stock_C_W4);

subto Max_Capacity_of_Production_Line1_W1:
Produced_A_W1 + Produced_B_W1 <= 120;
subto Max_Capacity_of_Production_Line1_W2:
Produced_A_W2 + Produced_B_W2 <= 120;
subto Max_Capacity_of_Production_Line1_W3:
Produced_A_W3 + Produced_B_W3 <= 120;
subto Max_Capacity_of_Production_Line1_W4:
Produced_A_W4 + Produced_B_W4 <= 120;
subto Max_Capacity_of_Production_Line2_W1:
 Produced_C_W1 <= 48;
```

Example 3.4 by ZIMPL cont.

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

```
subto Max_Capacity_of_Production_Line2_W2:
 Produced_C_W2 <= 48;
subto Max_Capacity_of_Production_Line2_W3:
 Produced_C_W3 <= 48;
subto Max_Capacity_of_Production_Line2_W4:
 Produced_C_W4 <= 48;
subto Max_Capacity_of_Man_Power_W1:
10 * Produced_A_W1 + 15 * Produced_B_W1 + 20 * Produced_C_W1
   <= 2000;
subto Max_Capacity_of_Man_Power_W2:
10 * Produced_A_W2 + 15 * Produced_B_W2 + 20 * Produced_C_W2
   <= 2000;
subto Max_Capacity_of_Man_Power_W3:
10 * Produced_A_W3 + 15 * Produced_B_W3 + 20 * Produced_C_W3
   <= 2000;
subto Max_Capacity_of_Man_Power_W4:
10 * Produced_A_W4 + 15 * Produced_B_W4 + 20 * Produced_C_W4
   <= 2000;
```

Example 3.4 by ZIMPL cont.

```
subto A_in_Depot_W1:
Stock_A_W1 == 22 + Produced_A_W1 - Sold_A_W1;
subto A_in_Depot_W2:
Stock_A_W2 == Stock_A_W1 + Produced_A_W2 - Sold_A_W2;
subto A_in_Depot_W3:
Stock_A_W3 == Stock_A_W2 + Produced_A_W3 - Sold_A_W3;
subto A_in_Depot_W4:
Stock_A_W4 == Stock_A_W3 + Produced_A_W4 - Sold_A_W4;
subto B_in_Depot_W1:
Stock_B_W1 == 42 + Produced_B_W1 - Sold_B_W1;
subto B_in_Depot_W2:
Stock_B_W2 == Stock_B_W1 + Produced_B_W2 - Sold_B_W2;
subto B_in_Depot_W3:
Stock_B_W3 == Stock_B_W2 + Produced_B_W3 - Sold_B_W3;
subto B_in_Depot_W4:
Stock_B_W4 == Stock_B_W3 + Produced_B_W4 - Sold_B_W4;
```

Example 3.4 by ZIMPL cont.

```
subto C_in_Depot_W1:
Stock_C_W1 == 36 + Produced_C_W1 - Sold_C_W1;
subto C_in_Depot_W2:
Stock_C_W2 == Stock_C_W1 + Produced_C_W2 - Sold_C_W2;
subto C_in_Depot_W3:
Stock_C_W3 == Stock_C_W2 + Produced_C_W3 - Sold_C_W3;
subto C_in_Depot_W4:
Stock_C_W4 == Stock_C_W3 + Produced_C_W4 - Sold_C_W4;
```

Example 3.4 by ZIMPL results

```
Optimal - objective value          -309241
     0  Sold_A_W1                       60
     1  Sold_A_W2                       80
     2  Sold_A_W3                      120
     3  Sold_A_W4                      140
     4  Sold_B_W1                       40
     5  Sold_B_W2                       40
     6  Sold_B_W3                       40
     7  Sold_B_W4                       22
     8  Sold_C_W1                       50
     9  Sold_C_W2                       40
    10  Sold_C_W3                       30
    11  Sold_C_W4                       52
    12  Stock_A_W1                       2
    13  Stock_A_W2                      26
    14  Stock_A_W3                      23
    15  Stock_A_W4                       0
    16  Stock_B_W1                      82
```

Example 3.4 by ZIMPL results cont.

```
17 Stock_B_W2                  58
18 Stock_B_W3                  20
19 Stock_B_W4                   0
20 Stock_C_W1                   6
21 Stock_C_W2                   2
22 Stock_C_W3                  12
23 Stock_C_W4                   0
24 Produced_B_W1               80
25 Produced_A_W1               40
26 Produced_B_W2               16
27 Produced_A_W2              104
28 Produced_B_W3                2
29 Produced_A_W3              117
30 Produced_B_W4                2
31 Produced_A_W4              117
32 Produced_C_W1               20
33 Produced_C_W2               36
34 Produced_C_W3               40
35 Produced_C_W4               40
```

Rolling horizon

- The danger is the sales will not match expectations and cause inventories to build up
  - The forecast is too optimistic
- The danger can be reduced by employing the model on a rolling horizon base
  - At the beginning of each week we reoptimize the model with new forecast based on the current sales

3.1.4. NETWORK MODELS

Introduction

- LP models with a mathematical structure corresponding to networks (graphs) are called network models
- Can be optimized very efficiently by tailored algorithms
- It displays a supply chain model as a network which is very useful for communication purpose

Example 3.5

- Transportation data of Alpha computers
  - Plant at Chicago (100 units of Alpha)
  - Warehouse in St. Louis (45 units of Alpha)
  - 8 markets

j = From\to	1 CHI	2 STL	3 DET	4 CIN	5 LOU	6 IND	7 MIL	8 MIN
Plant	14.00	24.00	21.00	20.00	21.50	19.00	17.00	30.00
Warehouse	24.00	15.00	28.00	20.00	18.50	19.50	24.00	28.00
Demand	22	14	18	17	15	13	15	20

$/Alpha

Transportation

- A truck transports Alphas between two facility
  - The first facility is the plant or the warehouse
  - The second facility is a market
  - There is no need to return to the facility
- Objective is to minimize the total transportation cost of shipping Alphas

Feasibility

- There exists feasible solution because the total supply (145) is more than the total demand (134)
- If the supply chain is complex, it is more difficult to verify

Decision variables

- $X_{\mathrm{P}j}$: flow of Alphas from the plant to market j
- $X_{\mathrm{W}j}$: flow of Alphas from the warehouse to market j

Model

$$\min Z = 14X_{P1} + 24X_{P2} + 21X_{P3} + 20X_{P4} + 21.5X_{P5} + 19X_{P6} + 17X_{P7} + 30X_{P8} + 24X_{W1} + 15X_{W2} + 28X_{W3} + 20X_{W4} + 18.5X_{W5} + 19.5X_{W6} + 24X_{W7} + 28X_{W8}$$

s.t.

$$X_{P1} + X_{P2} + X_{P3} + X_{P4} + X_{P5} + X_{P6} + X_{P7} + X_{P8} \leq 100$$
$$X_{W1} + X_{W2} + X_{W3} + X_{W4} + X_{W5} + X_{W6} + X_{W7} + X_{W8} \leq 45$$
$$X_{P1} + X_{W1} = 22$$
$$X_{P2} + X_{W2} = 14$$
$$X_{P3} + X_{W3} = 18$$
$$X_{P4} + X_{W4} = 17$$
$$X_{P5} + X_{W5} = 15$$
$$X_{P6} + X_{W6} = 13$$
$$X_{P7} + X_{W7} = 15$$
$$X_{P8} + X_{W8} = 20$$

Optimal solution

- Minimal cost $2583.50

From/to	CHI	STL	DET	CIN	LOU	IND	MIL	MIN
Plant	22		18	17		13	15	4
Warehouse		14			15			16

Example 3.5 by ZIMPL

```
# Example 3.5 - Transportation problem

var A_Plant_Chicago integer >=0;
var A_Plant_Seatle integer >=0;
var A_Plant_Detroit integer >=0;
var A_Plant_Cincinnati integer >=0;
var A_Plant_Louisiana integer >=0;
var A_Plant_Indianapolis integer >=0;
var A_Plant_Milwaukee integer >=0;
var A_Plant_Minneapolis integer >=0;

var A_Warehouse_Chicago integer >=0;
var A_Warehouse_Seatle integer >=0;
var A_Warehouse_Detroit integer >=0;
var A_Warehouse_Cincinnati integer >=0;
var A_Warehouse_Louisiana integer >=0;
var A_Warehouse_Indianapolis integer >=0;
var A_Warehouse_Milwaukee integer >=0;
var A_Warehouse_Minneapolis integer >=0;
```

Example 3.5 by ZIMPL cont.

```
minimize Transfer_cost:
14 * A_Plant_Chicago + 24 * A_Plant_Seatle + 21 *
   A_Plant_Detroit + 20 * A_Plant_Cincinnati + 21.50 *
   A_Plant_Louisiana + 19 * A_Plant_Indianapolis + 17 *
   A_Plant_Milwaukee + 30 * A_Plant_Minneapolis + 24 *
   A_Warehouse_Chicago + 15 * A_Warehouse_Seatle + 28 *
   A_Warehouse_Detroit + 20 * A_Warehouse_Cincinnati + 18.50
   * A_Warehouse_Louisiana + 19.50 *
   A_Warehouse_Indianapolis + 24 * A_Warehouse_Milwaukee +
   28 * A_Warehouse_Minneapolis;
```

Example 3.5 by ZIMPL cont.

```
subto Max_capacity_of_Plant:
A_Plant_Chicago + A_Plant_Seatle + A_Plant_Detroit +
   A_Plant_Cincinnati + A_Plant_Louisiana +
   A_Plant_Indianapolis + A_Plant_Milwaukee +
   A_Plant_Minneapolis <= 100;

subto Max_capacity_of_Warehouse:
A_Warehouse_Chicago + A_Warehouse_Seatle +
   A_Warehouse_Detroit + A_Warehouse_Cincinnati +
   A_Warehouse_Louisiana + A_Warehouse_Indianapolis +
   A_Warehouse_Milwaukee + A_Warehouse_Minneapolis <= 45;
```

Example 3.5 by ZIMPL cont.

```
subto Need_in_Chicago:
A_Plant_Chicago + A_Warehouse_Chicago == 22;
subto Need_in_Seatle:
A_Plant_Seatle + A_Warehouse_Seatle == 14;
subto Need_in_Detroit:
A_Plant_Detroit + A_Warehouse_Detroit == 18;
subto Need_in_Cincinnati:
A_Plant_Cincinnati + A_Warehouse_Cincinnati == 17;
subto Need_in_Louisiana:
A_Plant_Louisiana + A_Warehouse_Louisiana == 15;
subto Need_in_Indianapolis:
A_Plant_Indianapolis + A_Warehouse_Indianapolis == 13;
subto Need_in_Milwaukee:
A_Plant_Milwaukee + A_Warehouse_Milwaukee == 15;
subto Need_in_Minneapolis:
A_Plant_Minneapolis + A_Warehouse_Minneapolis == 20;
```

Example 3.5 by ZIMPL results

```
Optimal - objective value              2583.5
      0 A_Plant_Chicago                  22
      1 A_Plant_Seatle                    0
      2 A_Plant_Detroit                  18
      3 A_Plant_Cincinnati               17
      4 A_Plant_Louisiana                 0
      5 A_Plant_Indianapolis             13
      6 A_Plant_Milwaukee                15
      7 A_Plant_Minneapolis               4
      8 A_Warehouse_Chicago               0
      9 A_Warehouse_Seatle               14
     10 A_Warehouse_Detroit               0
     11 A_Warehouse_Cincinnati            0
     12 A_Warehouse_Louisiana            15
     13 A_Warehouse_Indianapolis          0
     14 A_Warehouse_Milwaukee             0
     15 A_Warehouse_Minneapolis          16
```

Example 3.5 by ZIMPL ("set" model)

```
# Example 3.5 - Transportation problem

#sets
set Plants := {"CHI"};
set Warehouses := {"StL"};
set Markets := {"CHI", "STL", "DET", "CIN", "LOU", "IND", "MIL",
   "MIN"};

#parameters
param CostBetweenPlantsAndMarkets[Plants*Markets] :=
         |"CHI", "STL", "DET", "CIN", "LOU", "IND", "MIL", "MIN"|
   |"CHI"| 14,    24,     21,    20,    21.5,   19,     17,    30   |;
param CostBetweenWhousesAndMarkets[Warehouses*Markets] :=
         |"CHI", "STL", "DET", "CIN", "LOU", "IND", "MIL", "MIN"|
   |"StL"| 24,     15,    28,    20,    18.5,  19.5,    24,    28   |;
param Demands[Markets] := <"CHI"> 22, <"STL"> 14, <"DET"> 18,
   <"CIN"> 17, <"LOU"> 15, <"IND"> 13, <"MIL"> 15, <"MIN"> 20;
param InventoryinPlants[Plants] := <"CHI"> 100;
param InventoryinWhouses[Warehouses] := <"StL"> 45;
```

Example 3.5 by ZIMPL ("set" model) cont.

```
# variables
var FlowFromPlants[Plants*Markets] >=0;
var FlowFromWhouses[Warehouses*Markets] >=0;

#objective
minimize Transfer_Cost:
sum <p,m> in Plants cross Markets :
   CostBetweenPlantsAndMarkets[p,m] * FlowFromPlants[p,m]
   + sum <w,m> in Warehouses cross Markets :
   CostBetweenWhousesAndMarkets[w,m] * FlowFromWhouses[w,m];
```

Example 3.5 by ZIMPL ("set" model) cont.

```
#constraints
subto Plants_constraints:
forall <p> in Plants do
   sum <m> in Markets : FlowFromPlants[p,m] <=
   InventoryinPlants[p];

subto Warehouses_constraints:
forall <w> in Warehouses do
   sum <m> in Markets : FlowFromWhouses[w,m] <=
   InventoryinWhouses[w];

subto Markets_constraints:
forall <m> in Markets do
   sum <p> in Plants : FlowFromPlants[p,m]
   + sum <w> in Warehouses : FlowFromWhouses[w,m] ==
   Demands[m];
```

Example 3.6

- The distribution manager whishes to investigate the potential of locating a cross-docking facility
  - Can receive products from the plant and the warehouse and dispatch them to three markets
- Handling charge of $2 for each Alpha
- Maximal capacity of 30 Alphas per week

From/to	Cross-docking facility
Plant	11
Warehouse	10

From/to	CIN	LOU	IND
Cross-docking facility	6	5	5

$/Alpha

New decision variables

- X_{PC} : flow of Alphas from the plant to the cross-docking facility
- X_{WC} : flow of Alphas from the warehouse to the cross-docking facility
- X : throughput at the cross-docking facility
- $Y_{\mathrm{C}j}$: flow from the cross-docking facility to market j (j = 4, 5, 6)

Model

$$\min Z = 14X_{P1} + 24X_{P2} + 21X_{P3} + 20X_{P4} + 21.5X_{P5} + 19X_{P6} + 17X_{P7} + 30X_{P8} + 24X_{W1} + 15X_{W2} + 28X_{W3} + 20X_{W4} + 18.5X_{W5} + 19.5X_{W6} + 24X_{W7} + 28X_{W8} + 11X_{PC} + 10X_{WC} + 2X + 6Y_{C4} + 5Y_{C5} + 5Y_{C6}$$

s.t.

$$X_{P1} + X_{P2} + X_{P3} + X_{P4} + X_{P5} + X_{P6} + X_{P7} + X_{P8} + X_{PC} \leq 100$$
$$X_{W1} + X_{W2} + X_{W3} + X_{W4} + X_{W5} + X_{W6} + X_{W7} + X_{W8} + X_{WC} \leq 45$$
$$X_{P1} + X_{W1} = 22$$
$$X_{P2} + X_{W2} = 14$$
$$X_{P3} + X_{W3} = 18$$
$$X_{P4} + X_{W4} + \mathrm{Y}_{C4} = 17$$
$$X_{P5} + X_{W5} + \mathrm{Y}_{C5} = 15$$
$$X_{P6} + X_{W6} + \mathrm{Y}_{C6} = 13$$
$$X_{P7} + X_{W7} = 15$$
$$X_{P8} + X_{W8} = 20$$
$$X - X_{PC} - X_{WC} = 0$$
$$X - Y_{C4} - Y_{C5} - Y_{C6} = 0$$
$$X \leq 30$$

Optimal solution

- Minimal cost $2542
  - $41.50 savings

From/to	CHI	STL	DET	CIN	LOU	IND	MIL	MIN	Cross-docking
Plant	22		18	2		13	15		19
Warehouse		14						20	11
Cross-docking				15	15	0			

Example 3.6 by ZIMPL

```
# Example 3.6
# Transportation problem with transloading at the cross-
   docking facility

var A_Plant_Chicago integer >=0;
var A_Plant_Seatle integer >=0;
var A_Plant_Detroit integer >=0;
var A_Plant_Cincinnati integer >=0;
var A_Plant_Louisiana integer >=0;
var A_Plant_Indianapolis integer >=0;
var A_Plant_Milwaukee integer >=0;
var A_Plant_Minneapolis integer >=0;
var A_Plant_TS integer >=0;
```

Example 3.6 by ZIMPL cont.

```
var A_Warehouse_Chicago integer >=0;
var A_Warehouse_Seatle integer >=0;
var A_Warehouse_Detroit integer >=0;
var A_Warehouse_Cincinnati integer >=0;
var A_Warehouse_Louisiana integer >=0;
var A_Warehouse_Indianapolis integer >=0;
var A_Warehouse_Milwaukee integer >=0;
var A_Warehouse_Minneapolis integer >=0;
var A_Warehouse_TS integer >=0;

var Transloading integer <=30;

var A_TS_Cincinnati integer >=0;
var A_TS_Louisiana integer >=0;
var A_TS_Indianapolis integer >=0;
```

Example 3.6 by ZIMPL cont.

```
minimize Transfer_cost:
14 * A_Plant_Chicago + 24 * A_Plant_Seatle + 21 *
   A_Plant_Detroit + 20 * A_Plant_Cincinnati + 21.50 *
   A_Plant_Louisiana + 19 * A_Plant_Indianapolis + 17 *
   A_Plant_Milwaukee + 30 * A_Plant_Minneapolis + 24 *
   A_Warehouse_Chicago + 15 * A_Warehouse_Seatle + 28 *
   A_Warehouse_Detroit + 20 * A_Warehouse_Cincinnati + 18.50
   * A_Warehouse_Louisiana + 19.50 *
   A_Warehouse_Indianapolis + 24 * A_Warehouse_Milwaukee +
   28 * A_Warehouse_Minneapolis + 11 * A_Plant_TS + 10 *
   A_Warehouse_TS + 2 * Transloading + 6 * A_TS_Cincinnati +
   5 * A_TS_Louisiana + 5 * A_TS_Indianapolis;

subto Max_capacity_of_Plant:
A_Plant_Chicago + A_Plant_Seatle + A_Plant_Detroit +
   A_Plant_Cincinnati + A_Plant_Louisiana +
   A_Plant_Indianapolis + A_Plant_Milwaukee +
   A_Plant_Minneapolis + A_Plant_TS <= 100;
```

Example 3.6 by ZIMPL cont.

```
subto Max_capacity_of_Warehouse:
A_Warehouse_Chicago + A_Warehouse_Seatle +
   A_Warehouse_Detroit + A_Warehouse_Cincinnati +
   A_Warehouse_Louisiana + A_Warehouse_Indianapolis +
   A_Warehouse_Milwaukee + A_Warehouse_Minneapolis +
   A_Warehouse_TS <= 45;

subto Need_in_Chicago:
A_Plant_Chicago + A_Warehouse_Chicago == 22;
subto Need_in_Seatle:
A_Plant_Seatle + A_Warehouse_Seatle == 14;
subto Need_in_Detroit:
A_Plant_Detroit + A_Warehouse_Detroit == 18;
subto Need_in_Cincinnati:
A_Plant_Cincinnati + A_Warehouse_Cincinnati +
   A_TS_Cincinnati == 17;
```

Example 3.6 by ZIMPL cont.

```
subto Need_in_Louisiana:
A_Plant_Louisiana + A_Warehouse_Louisiana + A_TS_Louisiana
   == 15;
subto Need_in_Indianapolis:
A_Plant_Indianapolis + A_Warehouse_Indianapolis +
   A_TS_Indianapolis == 13;
subto Need_in_Milwaukee:
A_Plant_Milwaukee + A_Warehouse_Milwaukee == 15;
subto Need_in_Minneapolis:
A_Plant_Minneapolis + A_Warehouse_Minneapolis == 20;

subto Transfer_Station:
Transloading - A_Plant_TS - A_Warehouse_TS == 0;
subto Transfer_Station_2:
Transloading - A_TS_Cincinnati - A_TS_Louisiana -
   A_TS_Indianapolis == 0;
```

Example 3.6 by ZIMPL results

```
Optimal - objective value                2542
      0 A_Plant_Chicago                    22
      1 A_Plant_Seatle                      0
      2 A_Plant_Detroit                    18
      3 A_Plant_Cincinnati                  2
      4 A_Plant_Louisiana                   0
      5 A_Plant_Indianapolis               13
      6 A_Plant_Milwaukee                  15
      7 A_Plant_Minneapolis                 0
      8 A_Plant_TS                         19
      9 A_Warehouse_Chicago                 0
     10 A_Warehouse_Seatle                 14
     11 A_Warehouse_Detroit                 0
     12 A_Warehouse_Cincinnati              0
     13 A_Warehouse_Louisiana               0
```

Example 3.6 by ZIMPL results cont.

```
14 A_Warehouse_Indianapolis       0
15 A_Warehouse_Milwaukee          0
16 A_Warehouse_Minneapolis       20
17 A_Warehouse_TS                11
18 Transloading                  30
19 A_TS_Cincinnati               15
20 A_TS_Louisiana                15
21 A_TS_Indianapolis              0
```

3.2. PROPERTIES OF LINEAR PROGRAMMING MODELS

Introduction

- LP models of significant size are usually easy to optimize
- LP models can be soled by general-purpose optimizers
- The five fundamental properties of an LP model
  - Linearity
  - Separability and additivity
  - Indivisibility and continuity
  - Single-objective function
  - Data known with certain

3.2.1. LINEARITY

Introduction

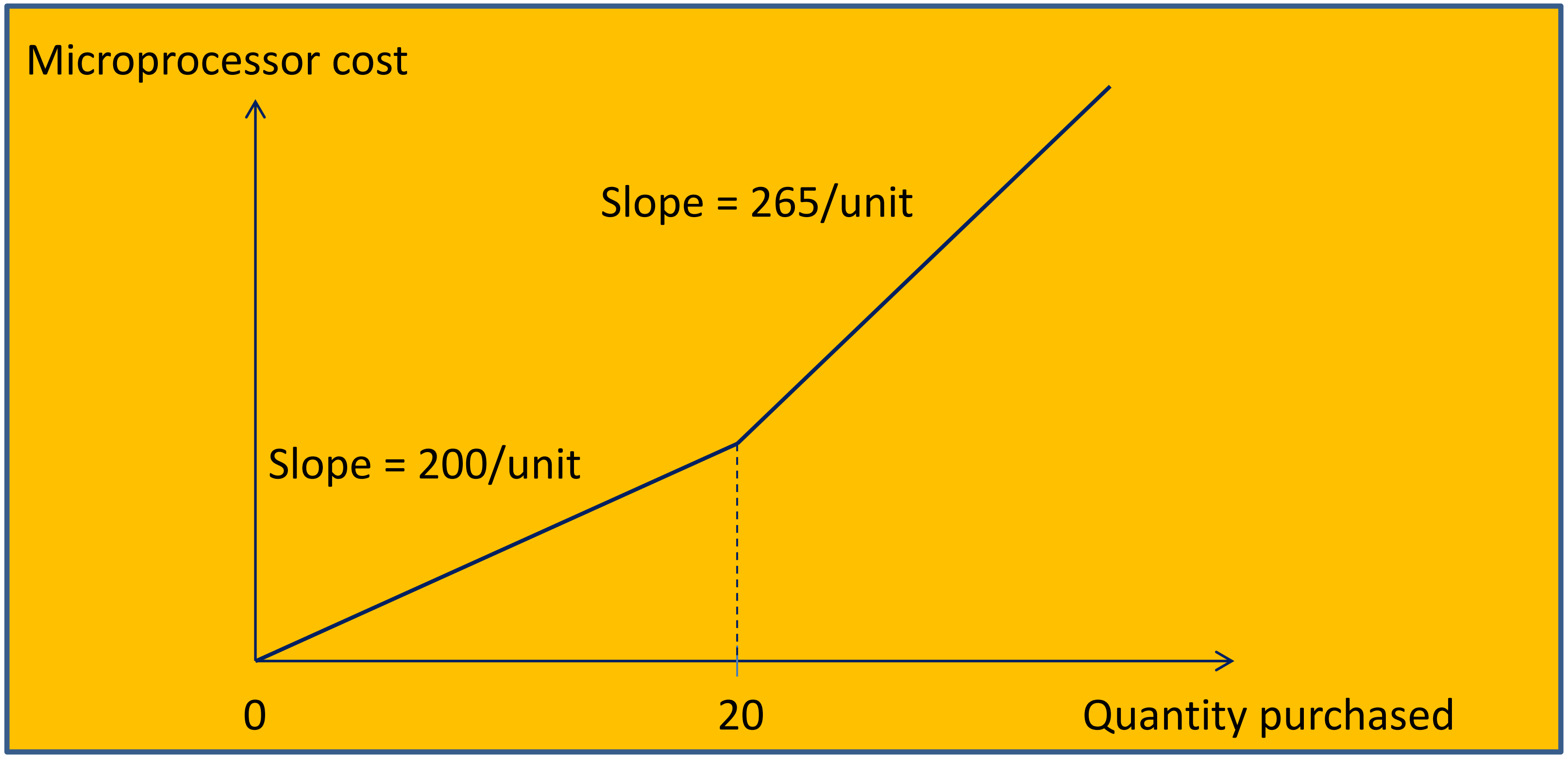
- In our all examples all revenues, costs and resource utilization occurs as a linear function of the associated decision variables
- It is easy to imagine situations where revenues or costs are not linear

Example 3.7

- Ajax acquires microprocessors for the Gamma computers from the Riverfront Corporation (RC)
  - $200/unit
  - Limited capacity : 20 units/week
- Another vendor: Glendale Company (GC)
  - $265/unit

Diseconomy of scale

- The unit cost increases at higher volume



Decision variables

- This type of model can be handle by LP model
- Decision variables
  - V_1 : weekly microprocessor purchase from RC
  - V_2 : weekly microprocessor purchase from GC

Changes in the model

- Objective function
  - Max $Z = 350M_A + 470M_B + 810M_C - 200V_1 - 265V_2$
- Constraints
  - $M_C = V_1 + V_2$
  - $0 \leq V_1 \leq 20$
  - $0 \leq V_2$
- Because RC is cheaper than GC, the objective function force the decision variable V_2 to zero till V_1 reach its upper limit

Example 3.7 by ZIMPL

```
# Example 3.7
# Imported processors from two different suppliers

var Produced_Alpha integer >=0;
var Produced_Beta integer >=0;
var Produced_Gamma integer >=0;
var RC_Proc integer <=20;
var GC_Proc integer >=0;

maximize Income_in_Dollars:
350 * Produced_Alpha + 470 * Produced_Beta + (610 + 200) *
   Produced_Gamma - 200 * RC_Proc - 265 * GC_Proc;
```

Example 3.7 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta <= 120;

subto Max_Capacity_of_Production_Line2:
Produced_Gamma <= 48;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000;

subto Proc_for_Gamma:
Produced_Gamma == RC_Proc + GC_Proc;
```

Example 3.7 by ZIMPL results

```
Optimal - objective value              -65100
      0 Produced_Alpha                    120
      1 Produced_Beta                       0
      2 Produced_Gamma                     40
      3 RC_Proc                            20
      4 GC_Proc                            20
```

Example 3.7 by ZIMPL ("set" model)

- Modification of "set" model of example 3.1
  - New set for vendors
  - Capacity and cost parameters for vendors
  - New variables for purchased processors
  - New constraints for vendors
  - New constraint to buy as many processor as many computer will be produced
  - Add vendors cost to cost function

Example 3.7 by ZIMPL ("set" model)

```
# Example 3.7
# Imported processors from two different suppliers

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};
set vendors := {"RC", "GC"};
```

Example 3.7 by ZIMPL ("set" model) cont.

```
#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta"> 470,
   <"Gamma"> 810;
param availability[resources] := <"A-line"> 120, <"C-line">
   48, <"manpower"> 2000;
param usage[computers*resources] :=
           |"A-line", "C-line", "manpower"|
   |"Alpha"|   1,        0,         10     |
   |"Beta" |   1,        0,         15     |
   |"Gamma"|   0,        1,         20     |;
param CostOfProcessor[vendors] := <"RC"> 200, <"GC">
   265;
param Capacity[vendors] := <"RC"> 20, <"GC"> 99999;
param UsedProcessors[computers] := <"Alpha"> 0,
   <"Beta"> 0, <"Gamma"> 1;
```

Example 3.7 by ZIMPL ("set" model) cont.

```
# variables
var Produced[computers] >=0;
var Purchased[vendors] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c]
   - sum <v> in vendors : CostOfProcessor[v] *
   Purchased[v];
```

Example 3.7 by ZIMPL ("set" model) cont.

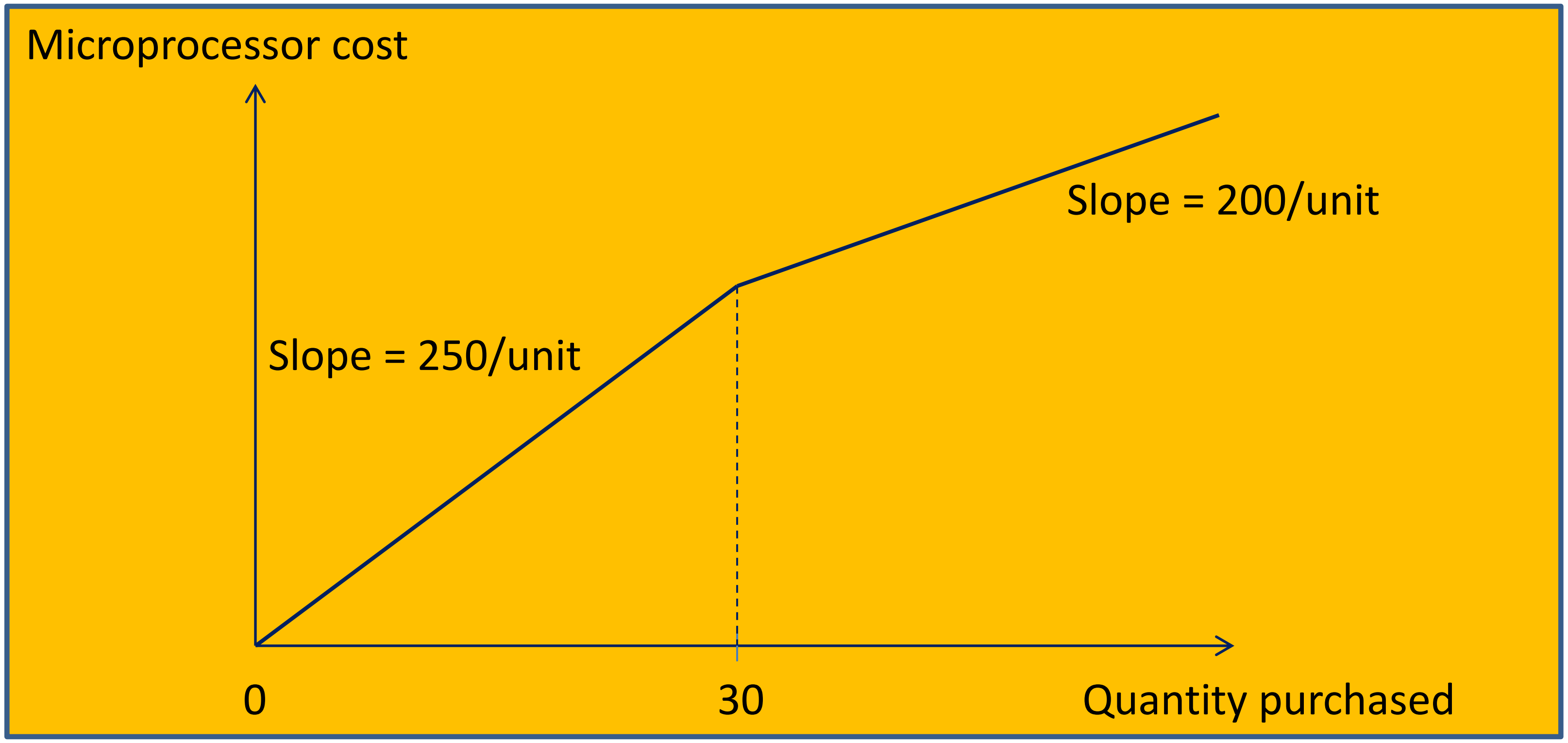
```
#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * Produced[c] <=
   availability[r];

subto Vendors_capcacity_constraints:
forall <v> in vendors do
   Purchased[v] <= Capacity[v];

subto Proc_for_computers:
sum <v> in vendors : Purchased[v] ==
   sum <c> in computers : Produced[c] *
   UsedProcessors[c];
```

Example 3.8

- Suppose we have only one vendor RC
- RC increased its capacity and changed its prices
  - $250/unit up to 30 per week
  - $200/unit otherwise

Economy of scale

- The unit cost decreases at higher volume



Changes in the model

- Try to change the model like previously
- Objective function
  - Max $Z = 350M_A + 470M_B + 810M_C - 250W_1 - 200W_2$
- Constraints
  - $M_C = W_1 + W_2$
  - $0 \leq W_1 \leq 30$
  - $0 \leq W_2$
- This model does not yield the desire result
- The modified LP model will choose first to increase W_2 from zero instead of W_1
- Mixed integer programming need

Nonlinearity in constraints

- Similarly the diseconomy of scale cases can be handle with LP model, the economy of case cannot

Example 3.8 by ZIMPL

```
# Example 3.8
# Imported processors from only one vendor, increased
  capacity and changed its prices

var Produced_Alpha integer >=0;
var Produced_Beta integer >=0;
var Produced_Gamma integer >=0;
var RC_Proc1 integer <=20;
var RC_Proc2 integer >=0;

maximize Income_in_Dollars:
350 * Produced_Alpha + 470 * Produced_Beta + (610 + 200) *
  Produced_Gamma - 250 * RC_Proc1 - 200 * RC_Proc2;
```

Example 3.8 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta <= 120;

subto Max_Capacity_of_Production_Line2:
 Produced_Gamma <= 48;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000;

subto Proc_for_Gamma:
Produced_Gamma == RC_Proc1 + RC_Proc2;
```

Example 3.8 by ZIMPL results

```
Optimal - objective value              -66400
      0 Produced_Alpha                   120
      1 Produced_Beta                      0
      2 Produced_Gamma                    40
      3 RC_Proc1                           0
      4 RC_Proc2                          40
```

Example 3.8 by ZIMPL ("set" model)

- Modification of "set" model of example 3.7
  - Only one vendor exists
  - No capacity parameter and constraint for vendor
  - Cost and limit parameters for vendor
  - New variables for purchased processors below and above the limit
  - Modified `Proc_for_computers` constraint
  - New constraint for economy limit
  - Modified vendors cost to cost function

Example 3.8 by ZIMPL ("set" model)

```
# Example 3.8
# Imported processors from only one vendor, increased
  capacity and changed its prices

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};
set vendors := {"RC"};
```

Example 3.8 by ZIMPL ("set" model) cont.

```
#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta">
   470, <"Gamma"> 810;
param availability[resources] := <"A-line"> 120, <"C-
   line"> 48, <"manpower"> 2000;
param usage[computers*resources] :=
            |"A-line", "C-line", "manpower"|
   |"Alpha"|   1,         0,          10      |
   |"Beta" |   1,         0,          15      |
   |"Gamma"|   0,         1,          20      |;
param CostOfProcessorBelowLimit[vendors] := <"RC">
   250;
param CostOfProcessorAboveLimit[vendors] := <"RC">
   200;
param EconomyLimit[vendors] := <"RC"> 30;
param UsedProcessors[computers] := <"Gamma"> 1 default
   0;
```

Example 3.8 by ZIMPL ("set" model) cont.

```
# variables
var Produced[computers] >=0;
var PurchasedBelowLimit[vendors] >=0;
var PurchasedAboveLimit[vendors] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c]
  - sum <v> in vendors : CostOfProcessorBelowLimit[v]
  * PurchasedBelowLimit[v]
  - sum <v> in vendors : CostOfProcessorAboveLimit[v]
  * PurchasedAboveLimit[v];
```

Example 3.8 by ZIMPL ("set" model) cont.

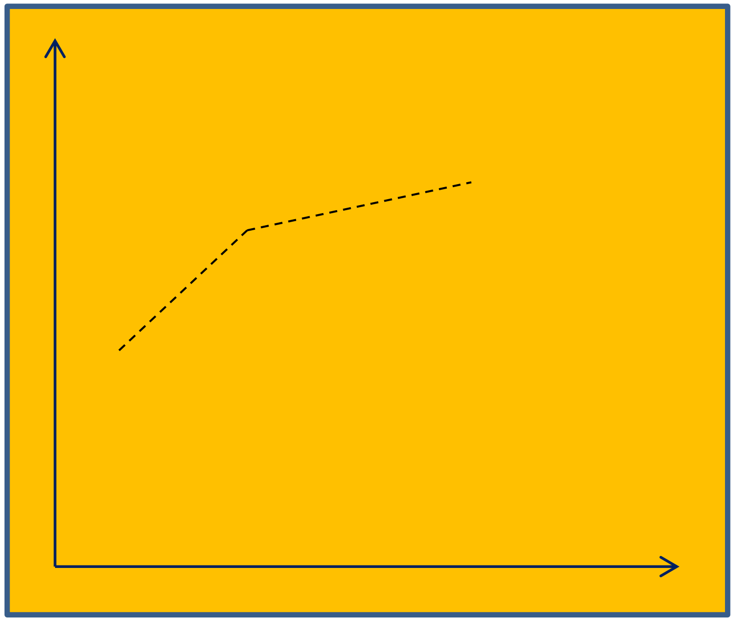
```
#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * Produced[c] <=
   availability[r];

subto Vendors_EconomyLimit_constraints:
forall <v> in vendors do
   PurchasedBelowLimit[v] <= EconomyLimit[v];

subto Proc_for_computers:
sum <v> in vendors : (PurchasedBelowLimit[v] +
   PurchasedAboveLimit[v]) ==
   sum <p> in computers : Produced[p] *
   UsedProcessors[p];
```

Example 3.9

- Another type of nonlinearity
- The quantity of Alphas that can be sold depends on its price
  - $Q_A = 240 - 0.4P_A$
    - Q_A : quantity of Alphas
    - P_A : price of Alpha
- Objective
  - Revenue = $Q_A P_A = 600Q_A - 2.5Q_A^2$
- Quadratic function

Model

- Piecewise linear approximation



3.2.2. SEPARABILITY AND ADDITIVITY

Separability

- The example is separable because the net profit of Alpha does not depend on
  - Its resource utilization
  - Net profit of Beta and Gamma
  - Resource utilization of Beta and Gamma
- Additivity : the separable effects can be accumulated simply by adding them up

Example 3.10

- The sales of all three products are sensitive with cross-price effects
- If the price of Alphas rises, the sales of Betas and Gammas will increase
- Non-separable problem

3.2.3. INDIVISIBILITY AND CONTINUITY

Indivisibility

- Decision variables can take real values
- In most cases, it is necessary to explicitly constrain decision variables to equal integer
  – For example, the number of Alphas assembled in a week can take only integer value

Continuity

- Jumps is not allowed in revenues, costs, utilization of resource, …
- For example, suppose space for inventory in Ajax’s facility is limited to 60 units, and for $500 per week more space can be rented
  – The cost does not depend on the size of overshoot
  – Fixed cost

3.2.4. SINGLE-OBJECTIVE FUNCTION

Single-objective function

- There is only one objective function
- In the pervious example it was the maximization of net revenue
- In an optimal solution of a multiperiodic model the ending inventories are zeros
  - Short-term profit
- Company long-term stability and profitability (second objective) does not allow to reduce inventories to zero

3.2.5. DATA KNOWN WITH CERTAINTY

Introduction

- In the examples data was known
  - Costs
  - Prices
  - Capacities
  - …

Uncertainty

- Sensitivity analysis
- Multiple scenarios
- Stochastic linear programming models

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

3.3. INTERPRETING AN OPTIMAL LINEAR PROGRAMMING SOLUTION

Economic interpretation provides

- Useful information for valuing resources and attributing costs to requirements
  - Why optimal solution is optimal?
- The underpinning for sensitivity and parametric analysis of model data
- The algorithmic bases for applying linear programming for large and complex models

3.3.1. SHADOW PRICES

Introduction

- A fundamental ingredient on the economic analysis is the shadow prices
  - Associated with each constraints
- It is defined as the change in the value of the objective function if the right-hand side of the constraint is increased (or decreased) by 1 unit

Example 3.1

- Recasting the model into canonical form transforming all constraints into equations by introducing a new (slack) variable for each
  - $Z - 350M_A - 470M_B - 610M_C = 0$
  - $M_A + M_B + S_1 = 120$
  - $M_C + S_2 = 48$
  - $10M_A + 15M_B + 20M_C + S_3 = 2000$

Feasible solution

- Feasible solution can be found by setting decision variables to zero than the value of the slack variables are equal to the right-hand side
  - $Z = 0$
  - $S_1 = 120$
  - $S_2 = 48$
  - $S_3 = 2000$

Simplex method

- The simplex method proceeds by a series of manipulation of the initial canonical form
- Transform the mathematical formulation to identify feasible solutions and test them for optimality

Optimal solution

- The optimal canonical form is the following
  - $Z - 32.5M_B + 45S_1 + 30.5S_3 = 66.4$
  - $M_A + M_B + S_1 = 120$
  - $- 0.25M_B + 0.5S_1 + S_2 - 0.05S_3 = 8$
  - $0.25M_B + M_C - 0.5S_1 + 0.05S_3 = 40$

Values of variables

- Rewrite the form
  - $Z = 66.0 - 32.5M_B - 45S_1 - 30.5S_3$
  - $M_A = 120 - M_B - S_1$
  - $S_2 = 8 + 0.25M_B - 0.5S_1 + 0.05S_3$
  - $M_C = 40 - 0.25M_B - 0.5S_1 - 0.05S_3$
- Where Z, M_A, S_2 and M_C are the (dependent) basic variables, M_B, S_1 and S_3 are the (independent) nonbasic variables

Solution

- Any value for nonbasic variables yields into feasible solution
- If the nonbasic variables are zeros the objective value is maximal, i.e. optimal

Slack variables

- In the optimal solution S_1 and S_3 are zero indicating that the capacity of A-line and the labor hours are fully used
  - If we increased the capacity of A-line by 1 it would increase the objective function by $45
    - $Z = 66.0 - 32.5M_B - 45S_1 - 30.5S_3$
- The value of S_2 is 8, so we have an excess of C-line testing hours
  - Addition of an hour would not cause the objective function to increase

Type of constraints

- In case of maximization
  - The shadow prices of the constraints whose slack variables are in the basis are 0
  - ≤ constraints
    - The shadow price is nonnegative
  - ≥ constraints
    - The shadow price is nonpositive
  - = constraints
    - Unconstrained in sign

3.3.2. REDUCED COST COEFFICIENTS

Introduction

- From economic interpretation we can see that the objective function would decrease by $32.5 for every Beta computer
  - $Z = 66.0 - 32.5M_B - 45S_1 - 30.5S_3$
- This coefficient called the reduced cost

Calculating reduced cost

- The reduced cost of M_B can be derived by using the shadow prices subtract a charge for the resource consumed by the manufacture of a single Beta from the net profit it would bring in
- The same calculation can be made for all variables

Decision variables

- M_A : 350 – 45*1 – 0*0 – 30.5*10 = 0
  - Efficient production
- M_B : 470 – 45*1 – 0*0 – 30.5*15 = – 32.5
  - Increasing M_B by 1 causes decreasing the objective by 32.5
- M_C : 610 – 45*0 – 0*1 – 30.5*20 = 0
  - Efficient production

min $350M_A + 470M_B + 610M_C = 0$	
$M_A + M_B + S_1 = 120$	45
$M_C + S_2 = 48$	0
$10M_A + 15M_B + 20M_C + S_3 = 2000$	30.5

Slack variables

- S_1 : 0 – 45*1 – 0*0 – 30.5*0 = – 45
  - Increasing S_1 by 1 causes increasing the objective by 45
- S_2 : 0 – 45*0 – 0*1 – 30.5*0 = 0
  - Increasing S_2 has no effect on the objective
- S_3 : 0 – 45*0 – 0*0 – 30.5*1 = – 30.5
  - Increasing S_3 by 1 causes increasing the objective by 30.5

min $350M_A + 470M_B + 610M_C = 0$	
$M_A + M_B + S_1 = 120$	45
$M_C + S_2 = 48$	0
$10M_A + 15M_B + 20M_C + S_3 = 2000$	30.5

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

3.3.3. PARAMETRIC AND SENSITIVITY ANALYSIS

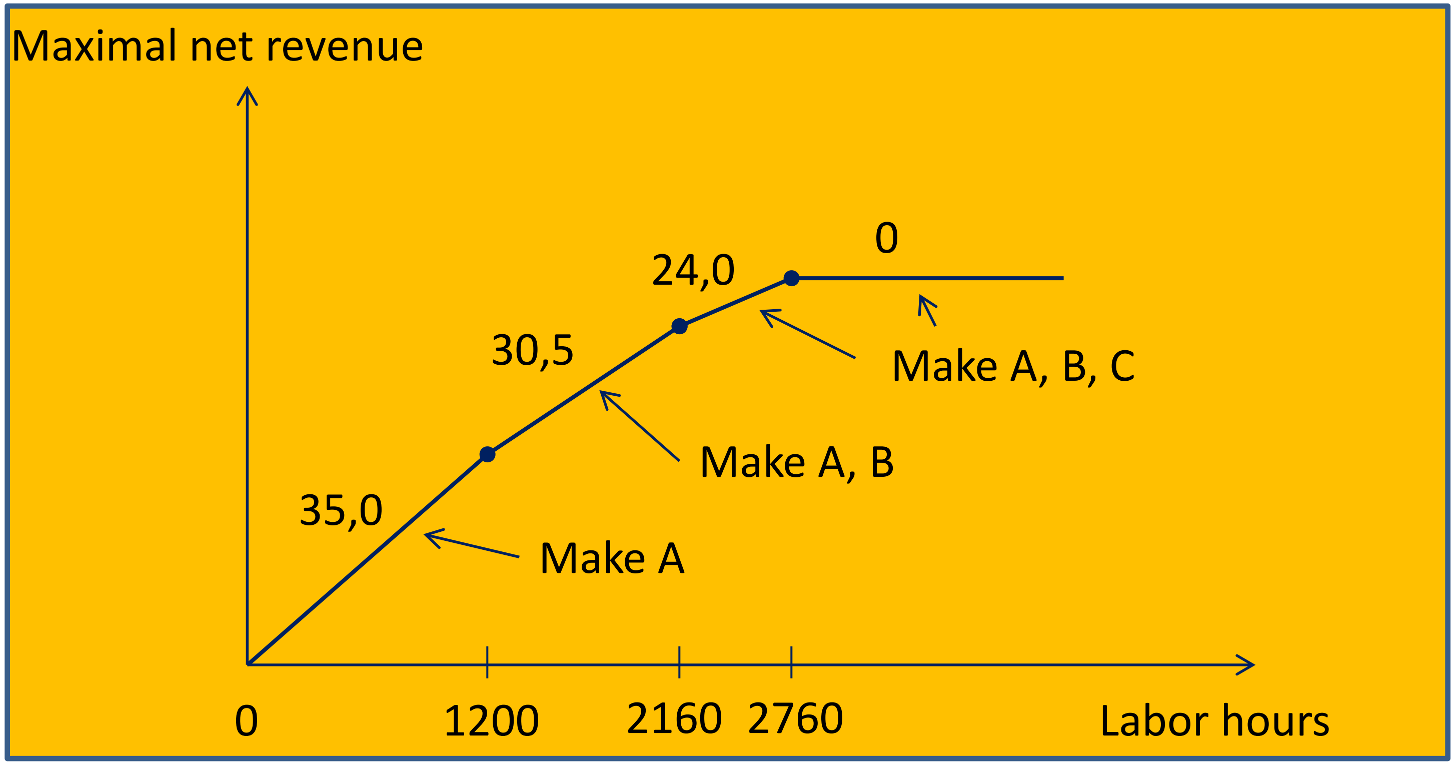
MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- The management would like to increase the production
- The reduced cost of the third constraint, 30.5, means increasing labor hours would increase the objective value
  - It is valid only at the current levels of resource availability
- The increasing of labor hours cannot be arbitrary big
- Decreasing labor hours has also effect on objective

Changing labor hours

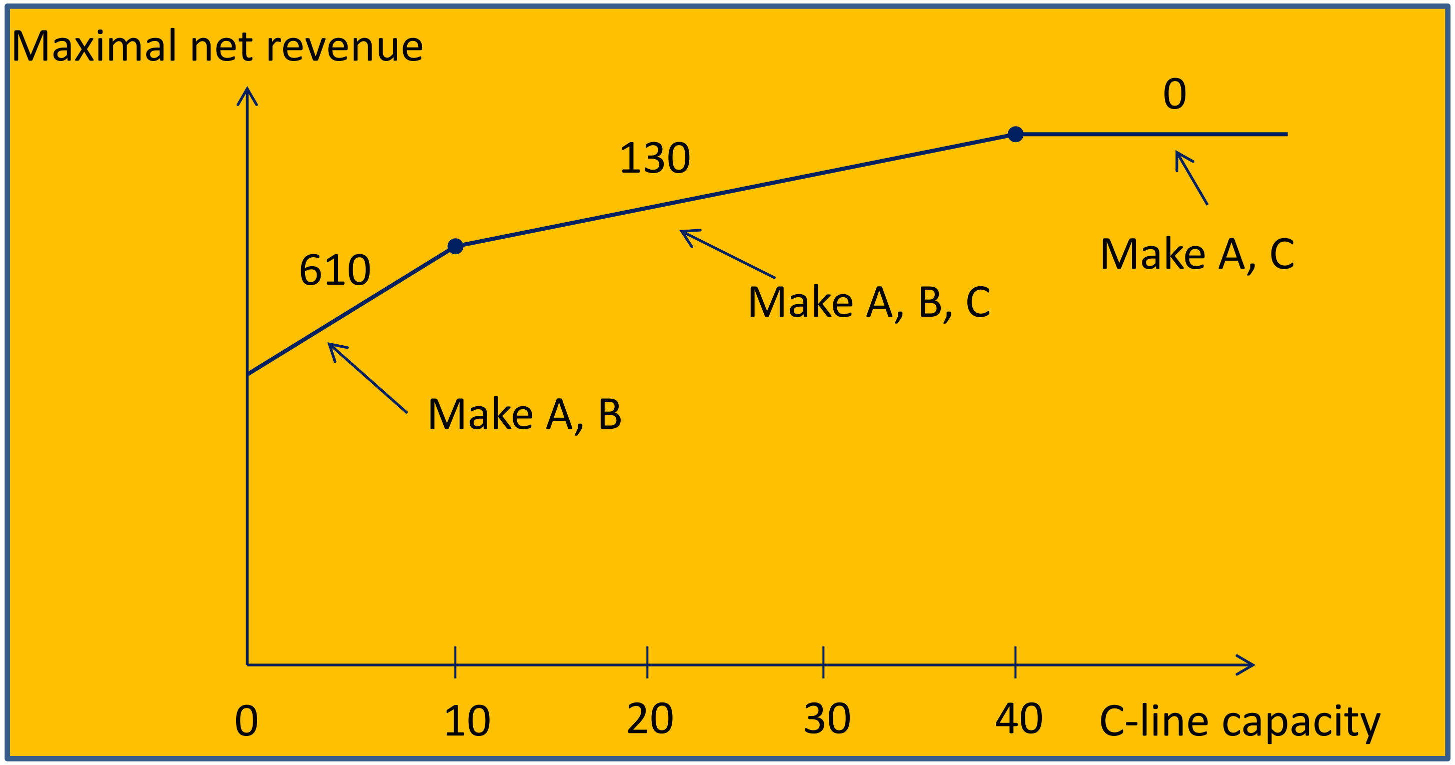
- Maximal net revenue versus available labor-hours



Changing C-line capacity

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

- Maximal net revenue versus C-line capacity



3.4. MULTIPLE-OBJECTIVE OPTIMIZATION

Introduction

- Net revenue may not be the only objective that the manager seeks to optimize
  - Customer satisfaction
  - Minimal risk
  - …

Effective frontier

- An optimal solution must be on the effective frontier
  - No achievable other solution exists that is at least as good with respect to all the objectives and strictly better on one of them
- Methods
  - Goal programming
  - Weighted objective function optimization

Goal programming

- The model has on objective function, the other objectives become constraints
  - We need bound for the objectives

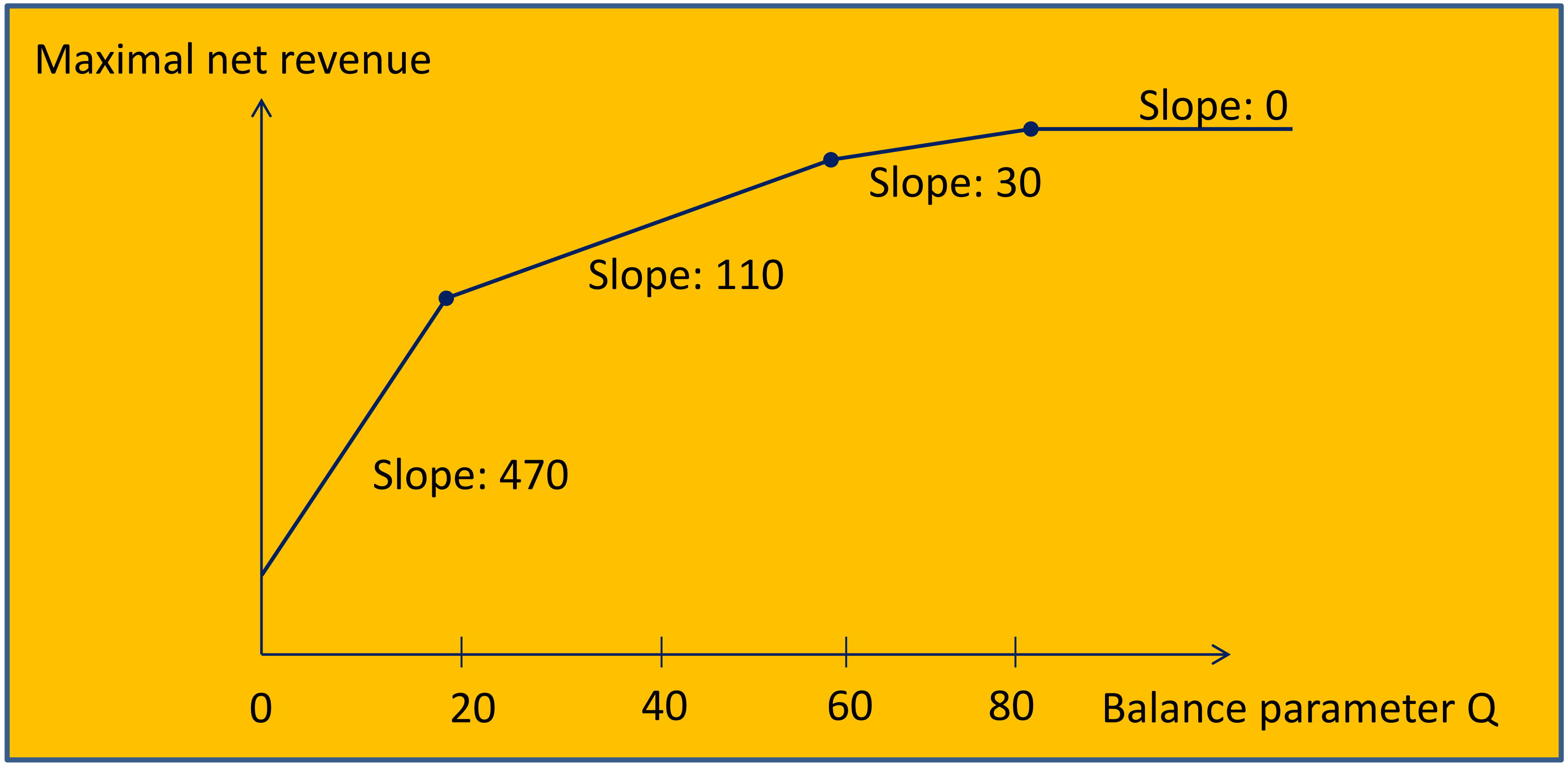
Example 3.11

- The management objects that they entails much heavier sales of desktop computers (Alphas) than workstations (Gammas)
- The marketing managers proposes the following constraints
  - The number of Alpha and Beat computers sold can be no more tan the number of Gammas sold plus *Q*
- This can be expressed as the goal constraint
  - $M_A + M_B - M_C \leq Q$

Solution

- If $Q = 20$
  - $M_A{}^* = 0$
  - $M_B{}^* = 68$
  - $M_C{}^* = 48$
  - $Z^* = 61{,}240\$$

Efficient frontier



Weighted objectives

- The objectives has weights by its importance
- The objective function is the sum of weighted objectives

Example 3.11 by ZIMPL

```
# Example 3.11
# Like Example 3.1 but the number of Alpha and Beta sold
   cannot be more than the number of Gamma sold plus Q
   (Q=20)

var Produced_Alpha integer >=0;
var Produced_Beta integer >=0;
var Produced_Gamma integer >=0;

maximize Income_in_Dollars:
350 * Produced_Alpha + 470 * Produced_Beta + 610 *
   Produced_Gamma;
```

Example 3.11 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta <= 120;

subto Max_Capacity_of_Production_Line2:
 Produced_Gamma <= 48;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000;

subto Goal_Constraint:
Produced_Alpha + Produced_Beta - Produced_Gamma <= 20;
```

Example 3.11 by ZIMPL results

```
Optimal - objective value           -61240
      0  Produced_Alpha                  0
      1  Produced_Beta                  68
      2  Produced_Gamma                 48
```

Example 3.11 by ZIMPL ("set" model)

- Modification of "set" model of example 3.1
  - New parameters for Goal and Q
  - New goal constraint

Example 3.11 by ZIMPL ("set" model) cont.

```
# Example 3.11
# Like Example 3.1 but the number of Alpha and Beta sold
   cannot be more than the number of Gamma sold plus Q
   (Q=20)

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};
```

Example 3.11 by ZIMPL ("set" model) cont.

```
#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta"> 470,
   <"Gamma"> 610;
param availability[resources] := <"A-line"> 120, <"C-line">
   48, <"manpower"> 2000;
param usage[computers*resources] :=
           |"A-line", "C-line", "manpower"|
   |"Alpha"|    1,        0,          10      |
   |"Beta" |    1,        0,          15      |
   |"Gamma"|    0,        1,          20      |;
param Goal[computers] := <"Alpha"> 1, <"Beta"> 1, <"Gamma">
   -1;
param Q := 20;
```

Example 3.11 by ZIMPL ("set" model) cont.

```
# variables
var Produced[computers] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c];

#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * Produced[c] <=
   availability[r];

subto Goal_constraint:
sum <p> in computers : Goal[p] * Produced[p] <= Q;
```

Example 3.12

- There is no exact number of the difference of sold computers
  - There is reward multiplier R for producing Gammas
  - The same value is penalty of producing Alphas and Betas
- Objective function
  - $Z(R) = 350M_A + 470M_C + 610M_C - RM_A - RM_B + RM_C = (350 - R)M_A + (470 - R)M_B + (610 + R)M_C$

Solution

- If $R = 50$
  - $M_A{}^* = 104$
  - $M_B{}^* = 0$
  - $M_C{}^* = 48$
  - $Z^*(50) = 62{,}880\$$

Example 3.12 by ZIMPL

```
# Example 3.12
# Like Example 3.11 but there's no exact number of the
  difference of sold computers
#R - Reward multiplier for producing Gammas and penalty of
  producing Alphas and Betas (R=50)

var Produced_Alpha integer >=0;
var Produced_Beta integer >=0;
var Produced_Gamma integer >=0;

maximize Income_in_Dollars:
350 * Produced_Alpha + 470 * Produced_Beta + 610 *
  Produced_Gamma - 50 * Produced_Alpha - 50 * Produced_Beta
  + 50 * Produced_Gamma;
```

Example 3.12 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta <= 120;

subto Max_Capacity_of_Production_Line2:
 Produced_Gamma <= 48;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000;
```

Example 3.12 by ZIMPL results

```
Optimal - objective value          -62880
      0 Produced_Alpha               104
      1 Produced_Beta                  0
      2 Produced_Gamma                48
```

Example 3.12 by ZIMPL ("set" model)

- Modification of "set" model of example 3.11
  - R parameter instead of Q parameter
  - Goal goes into the cost function
    - No goal constraint

Example 3.12 by ZIMPL ("set" model) cont.

```
# Example 3.12
# Like Example 3.11 but there's no exact number of the
  difference of sold computers
#R - Reward multiplier for producing Gammas and penalty of
  producing Alphas and Betas (R=50)

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};
```

Example 3.12 by ZIMPL ("set" model) cont.

```
#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta"> 470,
   <"Gamma"> 610;
param availability[resources] := <"A-line"> 120, <"C-line">
   48, <"manpower"> 2000;
param usage[computers*resources] :=
              |"A-line", "C-line", "manpower"|
   |"Alpha"|    1,         0,          10     |
   |"Beta" |    1,         0,          15     |
   |"Gamma"|    0,         1,          20     |;
param Goal[computers] := <"Alpha"> 1, <"Beta"> 1, <"Gamma">
   -1;
param R := 50;
```

Example 3.12 by ZIMPL ("set" model) cont.

```
# variables
var Produced[computers] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c]
   - sum <p> in computers : Goal[p] * R * Produced[p];

#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * Produced[c] <=
   availability[r];
```

3.5. STOCHASTIC PROGRAMMING

Introduction

- The models have to reflect changes in external and internal data dealing with uncertainties
  - Can be handled by optimizing multiple scenarios one by one
- The idea in stochastic programming is to simultaneously consider multiple scenarios of an uncertain future, each with an associated probability of occurrence

Example 3.13

- Extension of example 3.4
- 2-week planning horizon
  - First week is known
  - Three scenarios for the second week

Scenario	Alpha	Beta	Gamma	Probability
First week (0)	32	24	28	1.00

Scenario	Alpha	Beta	Gamma	Probability
Low (1)	40	30	30	0.25
Medium (2)	80	60	40	0.50
High (3)	120	100	100	0.25

Variables

- New decision variables
  - S_{pti} : sales of product *p* in week *t* in scenario *i*
  - M_{pti} : assembly and testing of product *p* in week *t* in scenario *i*
  - I_{pti} : inventory of product *p* at the end of the week *t* in scenario *i*
- New parameters
  - Sales forecasts [R_{pti}, A_{pti}]

Model

$$\max Z = 350S_{A10} + 470S_{B10} + 610S_{C10} - 9I_{A10} - 10I_{B10} - 18I_{C10} + 0.25(350S_{A21} + 470S_{B21} + 610S_{C21} - 9I_{A21} - 10I_{B21} - 18I_{C21}) + 0.5(350S_{A22} + 470S_{B22} + 610S_{C22} - 9I_{A22} - 10I_{B22} - 18I_{C22}) + 0.25(350S_{A23} + 470S_{B23} + 610S_{C23} - 9I_{A23} - 10I_{B23} - 18I_{C23})$$

s.t.

$M_{Ati} + M_{Bti} \leq 120$ $(t = 1, 2, 3, 4; i = 0, 1, 2, 3)$

$M_{Cti} \leq 48$ $(t = 1, 2, 3, 4; i = 0, 1, 2, 3)$

$10M_{Ati} + 15M_{Bti} + 20M_{Cti} \leq 2000$ $(t = 1, 2, 3, 4; i = 0, 1, 2, 3)$

$I_{pti} = I_{p,t-1,i} + M_{pti} - S_{pti}$ $(t = 1, 2, 3, 4; p = A, B, C; i = 0, 1, 2, 3)$

$M_{Ati}, M_{Bti}, M_{Cti} \geq 0$ $(t = 1, 2, 3, 4; i = 0, 1, 2, 3)$

$R_{pti} \leq S_{pti} \leq A_{pti}$ $(t = 1, 2, 3, 4; p = A, B, C; i = 0, 1, 2, 3)$

3.6 EXERCISES

Exercise 3.1

- Recognizing that A-line test capacity is tightly constrained according to the modeling analysis in Example 3.1
  - The production manager has solicited an offer from Jones Tech to perform out-sourced testing of Alphas
    - Specifically, Jones Tech will test up to 50 alphas a week at charge of $20/unit tested
  - The production manager estimates that there will be an additional overhead cost of $7.50/unit to outsource testing

Exercise 3.1

- Extend the model of Example 3.1 to incorporate and evaluate the outsourcing option

Exercise 3.1 by ZIMPL

```
# Exercise 3.1

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};

#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta">
   470, <"Gamma"> 610;
param availability[resources] := <"A-line"> 120, <"C-
   line"> 48, <"manpower"> 2000;
```

Exercise 3.1 by ZIMPL cont.

```
param usage[computers*resources] :=
          |"A-line", "C-line", "manpower"|
  |"Alpha"|   1,         0,          10      |
  |"Beta" |   1,         0,          15      |
  |"Gamma"|   0,         1,          20      |;
param OutsourceCost[computers*resources] :=
          |"A-line", "C-line", "manpower"|
  |"Alpha"| 27.5,        0,          0       |
  |"Beta" |   0,         0,          0       |
  |"Gamma"|   0,         0,          0       |;
param OutsourceCapacity[computers*resources] :=
          |"A-line", "C-line", "manpower"|
  |"Alpha"|  50,         0,          0       |
  |"Beta" |   0,         0,          0       |
  |"Gamma"|   0,         0,          0       |;
```

Exercise 3.1 by ZIMPL cont.

```
# variables
var Produced[computers] >=0;
var Outsource[computers*resources] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c]
   - sum <c,r> in computers cross resources :
   OutsourceCost[c,r] * Outsource[c,r];
```

Exercise 3.1 by ZIMPL cont.

```
#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : (usage[c,r] * Produced[c] -
   Outsource[c,r]) <= availability[r];

subto Outsource_constraints:
forall <c,r> in computers cross resources do
   Outsource[c,r] <= OutsourceCapacity[c,r];
```

Exercise 3.2

- After considering the strategy suggested by the 1-week Ajax Model given in Example 3.1, the marketing manager at Ajax revised his thinking on sales of Alphas, Betas and Gammas each week
- He now believes price discounts on Alphas and Gammas is selected sales locations are needed to reach the level of sales each week indicated by the optimal strategy
- High-volume sales of Beta would also require price discounts

Exercise 3.2

- Specifically, the selling process and therefore net profits need to be lowered according to the following table

	Sales range	Net profit ($)
Alphas	[0,60] [60,150]	350 315
Betas	[0, 40] [40, 70]	470 455
Gammas	[0, 20] [20, 40]	610 560

Exercise 3.2

- Note that, for simplicity, we have specified continuous ranges for sales of the three products recognizing that they are sold in integer amounts
  - A noninteger optimal solution to the LP model would be round down to be implemented
- Extend the model of Example 3.1 to include this price discounts

Exercise 3.2 by ZIMPL

```
# Exercise 3.2

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};
```

Exercise 3.2 by ZIMPL cont.

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

```
#parameters
param availability[resources] := <"A-line"> 120, <"C-
   line"> 48, <"manpower"> 2000;
param LowRangePrice[computers] := <"Alpha"> 350,
   <"Beta"> 470, <"Gamma"> 610;
param HighRangePrice[computers] := <"Alpha"> 315,
   <"Beta"> 455, <"Gamma"> 560;
param LowRangeSales[computers] := <"Alpha"> 60,
   <"Beta"> 40, <"Gamma"> 20;
param HighRangeSales[computers] := <"Alpha"> 90,
   <"Beta"> 30, <"Gamma"> 20;
param usage[computers*resources] :=
           |"A-line", "C-line", "manpower"|
   |"Alpha"|   1,         0,          10     |
   |"Beta" |   1,         0,          15     |
   |"Gamma"|   0,         1,          20     |;
```

Exercise 3.2 by ZIMPL cont.

```
# variables
var LowProduced[computers] >=0;
var HighProduced[computers] >=0;

#objective
maximize Income_in_Dollars:
sum <c> in computers : (LowRangePrice[c] *
   LowProduced[c] + HighRangePrice[c] *
   HighProduced[c]);
```

Exercise 3.2 by ZIMPL cont.

```
#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * (LowProduced[c]
   + HighProduced[c]) <= availability[r];

subto LowRange_constraints:
forall <c> in computers do
   LowProduced[c] <= LowRangeSales[c];

subto HighRange_constraints:
forall <c> in computers do
   HighProduced[c] <= HighRangeSales[c];
```

Exercise 3.3

- Ajax’s marketing representative to the Electrics Outlet ha interested the Outlet’s vice president of marketing in acquiring customized version of Alphas, Betas, and Gammas for sale in their stores
  - Specifically, the Outlet may be willing to enter into a long-term arrangement to acquire 10 Alphas, 10 Betas, and 10 Gammas every week

Exercise 3.3

- To negotiate the price for such an arrangement, the marketing representative has asked Ajax's production manager to determine the break-out price
  - He will then attempt to seek a healthy markup from the Outlet

Exercise 3.3

- After reviewing details of the product customizations, Ajax's production manager has determined the following data that describes their resource utilization

	Customized Alpha	Customized Beta	Customized Gamma
A-line test hours/unit	1.2	1.1	
C-line test hours/unit			1.25
Assembly hours/unit	11	16.5	22

Exercise 3.3

- Use these data to calculate the break-even price per week for the proposed arrangement by expanding the model of Example 3.1 to include assembly and testing of the customized products

Exercise 3.3 by ZIMPL

```
# Exercise 3.3

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};
```

Exercise 3.3 by ZIMPL cont.

```
#parameters
param availability[resources] := <"A-line"> 120, <"C-
   line"> 48, <"manpower"> 2000;
param price[computers] := <"Alpha"> 350, <"Beta"> 470,
   <"Gamma"> 610;
param packageSize[computers] := <"Alpha"> 10, <"Beta">
   10, <"Gamma"> 10;
param usage[computers*resources] :=
             |"A-line", "C-line", "manpower"|
   |"Alpha"|   1,         0,          10     |
   |"Beta" |   1,         0,          15     |
   |"Gamma"|   0,         1,          20     |;
param packageUsage[computers*resources] :=
             |"A-line", "C-line", "manpower"|
   |"Alpha"|   1.2,       0,          11     |
   |"Beta" |   1.1,       0,          16.5   |
   |"Gamma"|   0,         1.25,       22      |;
```

Exercise 3.3 by ZIMPL cont.

```
# variables
var Produced[computers] >=0;
var PackagePrice >=0;

#objective
minimize Package_Price:
PackagePrice;
```

Exercise 3.3 by ZIMPL cont.

```
#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : (usage[c,r] * Produced[c] +
   packageUsage[c,r] * packageSize[c]) <=
   availability[r];

subto No_Lost_constraint:
sum <c> in computers : price[c] * Produced[c] +
   PackagePrice >= 66400; #66400 is the optimum value
   of example 3.1
```

Exercise 3.4

- Specific Motors manufactures three different car models Model X, Model Y, and Model Z, with respective net revenues of $1000, $3000 and $6000
  – Each Model X requires 40 labor-hours and 1 ton of steel to produce
  – Each Model Y requires 65 labor-hours and 1.5 tons of steels
  – Each Model Z requires 110 labor-hours and 2 tons of steel

Exercise 3.4

- This month, Specific Motors has available
  - 16,000 labor-hours of labor
  - 600 tons of steel
  - Ample supply of all other relevant resources

Exercise 3.4

a) Formulate a linear programming model to find a plan that maximizes the monthly profits for Specific Motors

b) Suppose an engineer develops the design for a model Q yielding $4000 profit and requiring 120 labor-hours and 1.5 tons steel

Exercise 3.4a by ZIMPL

```
# Exercise 3.4a

#sets
set products := {"X", "Y", "Z"};
set resources := { "manpower", "steel"};

#parameters
param availability[resources] := <"manpower"> 16000,
   <"steel"> 600;
param price[products] := <"X"> 1000, <"Y"> 3000, <"Z">
   6000;
param usage[products*resources] :=
         |"manpower", "steel"|
   |"X"|     40,         1   |
   |"Y"|     65,         1.5 |
   |"Z"|    110,         2   |;
```

Exercise 3.4a by ZIMPL cont.

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

```
# variables
var Produced[products] >=0;

#objective
maximize Income_in_Dollars :
sum <p> in products : price[p] * Produced[p];

#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <p> in products : usage[p,r] * Produced[p] <=
   availability[r];
```

Exercise 3.4b by ZIMPL

```
# Exercise 3.4b

#sets
set products := {"X", "Y", "Z", "Q"};
set resources := { "manpower", "steel"};

#parameters
param availability[resources] := <"manpower"> 16000,
   <"steel"> 600;
param price[products] := <"X"> 1000, <"Y"> 3000, <"Z">
   6000, <"Q"> 4000;
param usage[products*resources] :=
         |"manpower", "steel"|
   |"X"|     40,          1   |
   |"Y"|     65,          1.5 |
   |"Z"|    110,          2   |
   |"Q"|    120,          1.5 |;
```

Exercise 3.4b by ZIMPL cont.

```
# variables
var Produced[products] >=0;

#objective
maximize Income_in_Dollars :
sum <p> in products : price[p] * Produced[p];

#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <p> in products : usage[p,r] * Produced[p] <=
   availability[r];
```

Exercise 3.5

- A grain cooperative with warehouse in Lincoln, Des Moines, and Pierre must meet market demands in Denver, Kansas City, Minneapolis, and St. Louis. The following table summarizes data relevant to this week’s distribution problem
- The numbers in the first three rows and the first four columns are the transportation costs/unit from each warehouse to each market

Exercise 3.5

From/to	Denver	Kansas City	Minneapolis	St. Louis	Supply
Lincoln	6	4	8	9	750
Des Moines	9	3	5	6	750
Pierre	9	11	6	14	500
Demand	400	700	400	500	

Exercise 3.5

a. Formulate an LP model to minimize the total transportation costs associated with the shipping product from the warehouse to meet market demand

b. Suppose the cooperative also has a warehouse in Omaha with a supply of 500 units and transportation costs to Denver, Kansas City, Minneapolis, and St. Louis of, respectively, 6, 5, 7, and 9
   - Extend the LP formulation of part (b) to include this option

Exercise 3.5a by ZIMPL

```
# Exercise 3.5a

#sets
set warehouses := {"Lincoln", "DesMoines", "Pierre"};
set markets := {"Denver", "Kansas", "Min.",
   "StLouis"};

#parameters
param demand[markets] := <"Denver"> 400, <"Kansas">
   700, <"Min."> 400, <"StLouis"> 500;
param supply[warehouses] := <"Lincoln"> 750,
   <"DesMoines"> 750, <"Pierre"> 500;
param cost[warehouses*markets] :=
              |"Denver", "Kansas", "Min.", "StLouis"|
  |"Lincoln"  |    6,        4,       8,        9    |
  |"DesMoines"|    9,        3,       5,        6    |
  |"Pierre"   |    9,       11,       6,       14    |;
```

Exercise 3.5a by ZIMPL cont.

```
# variables
var Transport[warehouses*markets] >=0;

#objective
minimize Transportation_Cost :
sum <w,m> in warehouses cross markets : cost[w,m] *
   Transport[w,m];

#constraints
subto Warehouses_constraints:
forall <w> in warehouses do
   sum <m> in markets : Transport[w,m] <= supply[w];

subto Markets_constraints:
forall <m> in markets do
   sum <w> in warehouses : Transport[w,m] >=
   demand[m];
```

Exercise 3.5b by ZIMPL

```
# Exercise 3.5b

#sets
set warehouses := {"Lincoln", "DesMoines", "Pierre",
   "Omaha"};
set markets := {"Denver", "Kansas", "Min.", "StLouis"};

#parameters
param demand[markets] := <"Denver"> 400, <"Kansas"> 700,
   <"Min."> 400, <"StLouis"> 500;
param supply[warehouses] := <"Lincoln"> 750, <"DesMoines">
   750, <"Pierre"> 500, <"Omaha"> 500;
param cost[warehouses*markets] :=
               |"Denver", "Kansas", "Min.", "StLouis"|
   |"Lincoln"  |    6,        4,      8,        9    |
   |"DesMoines"|    9,        3,      5,        6    |
   |"Pierre"   |    9,       11,      6,       14    |
   |"Omaha"    |    6,        5,      7,        9    |;
```

Exercise 3.5b by ZIMPL cont.

```
# variables
var Transport[warehouses*markets] >=0;

#objective
minimize Transportation_Cost :
sum <w,m> in warehouses cross markets : cost[w,m] *
   Transport[w,m];

#constraints
subto Warehouses_constraints:
forall <w> in warehouses do
   sum <m> in markets : Transport[w,m] <= supply[w];

subto Markets_constraints:
forall <m> in markets do
   sum <w> in warehouses : Transport[w,m] >=
   demand[m];
```

Exercise 3.6

- A classical application of linear programming is the blending problem
  - Minimize the cost of raw materials or maximize the revenues from the finished products subject to constraints on raw material availability, finished product-blending requirements, and finished-product demands

Exercise 3.6

- A pharmaceutical company faces a problem in manufacturing a product made from a natural raw material whose ingredients vary
- Specifically, the company's product planner has been provided with 10 drum of the raw material with the following characteristics

Exercise 3.6

Drum	Weight (kg)	Ingredient A (%)	Ingredient B (%)	Ingredient C (%)
1	80	60	20	20
2	78	59	18	23
3	76	57	19	24
4	81	56	20	24
5	82	54	21	25
6	80	53	20	27
7	77	51	19	30
8	70	50	20	30
9	81	49	19	32
10	79	48	20	32

Exercise 3.6

- The Food and Drug Administration has set the following requirements on product to be produced from these drums
  - Ingredient A percentage must lie within the range 54-56
  - Ingredient B percentage must lie within the range 19.5-20.5
  - Ingredient C percentage must lie within the range 24.2-25.8

Exercise 3.6

- The production planner wishes to maximize the total kilograms of raw material that can be blended together that meets the FDA requirements
- Formulate an LP model to solve this problem

Exercise 3.6 by ZIMPL

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

```
# Exercise 3.6

#sets
set drums := {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
set ingredients := {"A", "B", "C"};

#parameters
param weight[drums] := <1> 80, <2> 78, <3> 76, <4> 81,
   <5> 82, <6> 80, <7> 77, <8> 70, <9> 81, <10> 79;
param lowRange[ingredients] := <"A"> 54, <"B"> 19.5,
   <"C"> 24.2;
param upRange[ingredients] := <"A"> 56, <"B"> 20.5,
   <"C"> 25.8;
```

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Exercise 3.6 by ZIMPL cont.

```
param components[drums*ingredients] :=
         |"A", "B", "C"|
    |1 | 60,  20, 20 |
    |2 | 59,  18, 23 |
    |3 | 57,  19, 24 |
    |4 | 56,  20, 24 |
    |5 | 54,  21, 25 |
    |6 | 53,  20, 27 |
    |7 | 51,  19, 30 |
    |8 | 50,  20, 30 |
    |9 | 49,  19, 32 |
    |10| 48,  20, 32 |;
```

Exercise 3.6 by ZIMPL cont.

```
# variables
var Usage[drums] >=0 <=1;

#objective
maximize Production :
sum <d> in drums : Usage[d] * weight[d];
```

Exercise 3.6 by ZIMPL cont.

```
#constraints
subto Low_constraints:
forall <i> in ingredients do
   sum <d> in drums : Usage[d] * weight[d] *
   components[d,i] >= lowRange[i] * sum <d> in drums :
   Usage[d] * weight[d];

subto Up_constraints:
forall <i> in ingredients do
   sum <d> in drums : Usage[d] * weight[d] *
   components[d,i] <= upRange[i] * sum <d> in drums :
   Usage[d] * weight[d];
```

Exercise 3.7

- A classical application is the assignment problem
  - The company must assign *n* individuals to *n* tasks to minimize total cost or maximize total benefit

Exercise 3.7

- The company making home deliveries of groceries has selected 8 routes for today's delivery
  - These routes are scheduled to depart at different times during the day
- The company has 8 drivers who are also scheduled to arrive at different times during the day
  - Each driver has certain neighborhoods that he or she prefers for deliveries
    - Thus, for each route, the assignment of a specific driver will incur penalties

Exercise 3.7

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

- The following table describes the possibilities for individual assignments

Route	1	2	3	4	5	6	7	8
Sally	0	0	25	100	5	25	100	100
Jim	5	100	5	0	25	100	25	100
Bob	5	5	100	5	5	0	25	100
Carol	0	5	25	25	25	0	100	25
John	100	100	100	25	5	25	5	5
Bill	100	100	0	25	25	25	5	0
Frank	25	25	25	100	100	100	0	5
Harry	25	5	100	25	0	100	100	0

Exercise 3.7

- Formulate this assignment problem

Exercise 3.7 by ZIMPL

```
# Exercise 3.7


#sets
set routes := {1, 2, 3, 4, 5, 6, 7, 8};
set drivers := {"Sally", "Jim", "Bob", "Carol",
  "John", "Bill", "Frank", "Harry"};
#parameters
param penalty[drivers*routes] :=
          |   1,   2,   3,   4,   5,   6,   7,   8 |
  |"Sally"|   0,   0,  25, 100,   5,  25, 100, 100 |
  |"Jim"  |   5, 100,   5,   0,  25, 100,  25, 100 |
  |"Bob"  |   5,   5, 100,   5,   5,   0,  25, 100 |
  |"Carol"|   0,   5,  25,  25,  25,   0, 100,  25 |
  |"John" | 100, 100, 100,  25,   5,  25,   5,   5 |
  |"Bill" | 100, 100,   0,  25,  25,  25,   5,   0 |
  |"Frank"|  25,  25,  25, 100, 100, 100,   0,   5 |
  |"Harry"|  25,   5, 100,  25,   0, 100, 100,   0 |;
```

Exercise 3.7 by ZIMPL cont.

```
# variables
var Assignment[drivers*routes] >=0;

#objective
maximize Penalty :
sum <d,r> in drivers cross routes : Assignment[d,r] *
   penalty[d,r];

#constraints
subto Driver_constraints:
forall <d> in drivers do
   sum <r> in routes : Assignment[d,r] == 1;

subto Route_constraints:
forall <r> in routes do
   sum <d> in drivers : Assignment[d,r] == 1;
```

4. MIXED INTEGER PROGRAMMING

Introduction

- Mixed-integer linear programming (MILP) models are generalization of linear programming (LP) models in which some variables
  - Continuous variables
  - Integer variables
- Integer variables most frequently are binary (or 0-1) variables

Integer variables

- They are describe decisions
  - Operational problems
    - Sequencing decisions, …
  - Tactical problems
    - Fixed cost, …
  - Strategic problems
    - Investment options, …

Methods

- Branch-and-bound methods
- Heuristic methods

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

4.1. MIXED-INTEGER PROGRAMMING MODELING VIGNETTES

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- We illustrate several MILP model constructs for planning situations that arise in the management of Ajax’s supply chain

4.1.1. FIXED COST

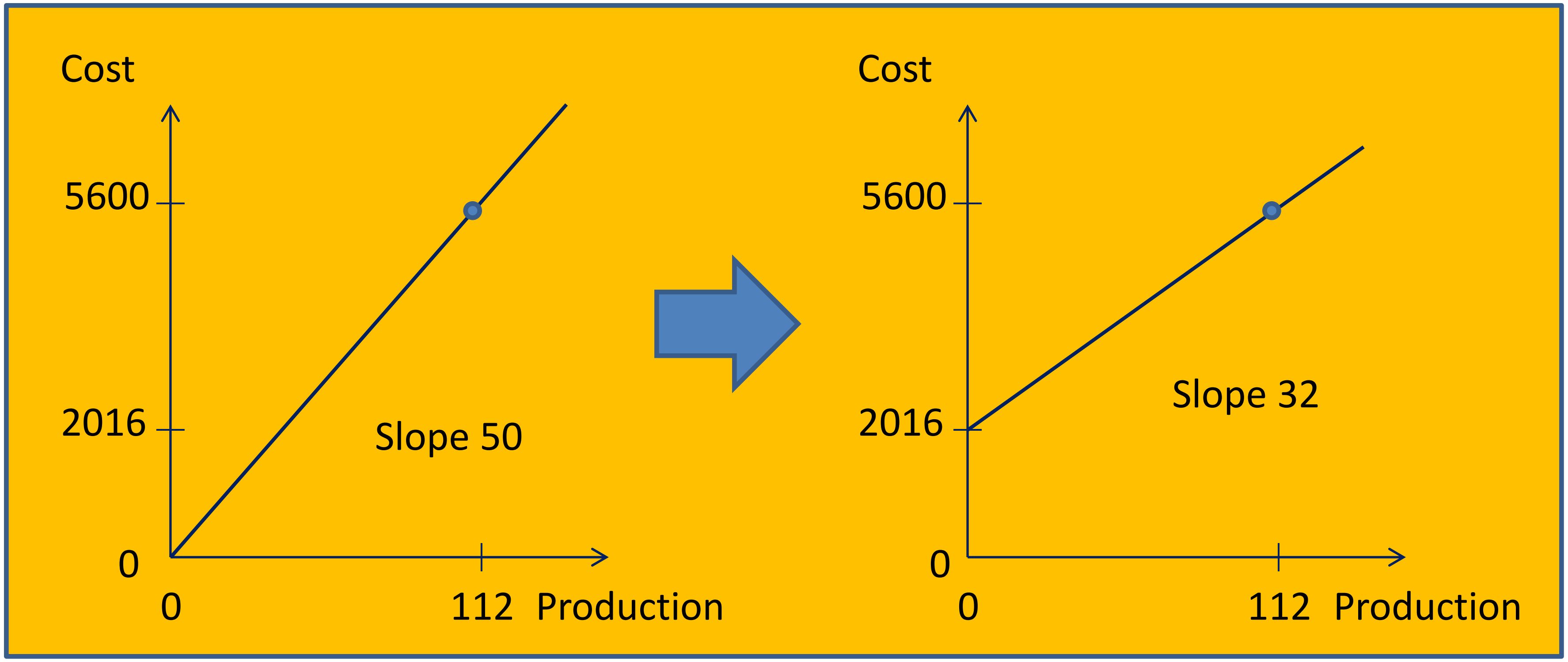
Introduction

- We will use costs for available resources instead of quantities
- The cost is not a linear function of the quantity of used resource

Cost function of A-line

- The historical weekly utilization is 112 hours (112 computers) of the A-line which costs $5600
  - The unit charge for A-line testing was $5600/112 = $50/hours
- The cost function involves a fixed cost of $2016 if the A-line is used at all during the week
  - For example, the labor cost of maintenance
  - It means the cost of testing a computer is $32

Cost of A-line



Cost of C-line

- Fixed cost: $1200
- Cost of testing a computer: $38.5

Fixed cost

- The term of fixed cost is misleading
- The fixed cost is not actually fixed because we might decide not to operate on of the two test lines

Example 4.1

- Integrate the new cost functions into example 3.1
- New decision variables
  - F_A : equals 1 if the A-line is used during the week; 0 otherwise
  - F_C : equals 1 if the C-line is used during the week; 0 otherwise

Model

- New constraints
  - $M_A + M_B - 120F_A \leq 0$
  - $M_C - 48F_C \leq 0$
- Objective function
  - Max $Z = 410M_A + 520M_B + 686M_C - 2016F_A - 1200F_C - 32M_A - 32M_B - 38.5M_C = 378M_A + 488M_B + 647.5M_C - 2016F_A - 1200F_C$

max $Z = 350M_A + 470M_B + 610M_C$
s.t.
$M_A + M_B \leq 120$
$M_C \leq 48$
$10M_A + 15M_B + 20M_C \leq 2000$

Solution

- $M_A^* = 120$
- $M_B^* = 0$
- $M_C^* = 40$
- $Z^* = 68{,}044\$$
- The solution is the same as in example 3.1

Example 4.1 by ZIMPL

```
# Example 4.1
# Like the example 3.1 with fixed costs

var Produced_Alpha >=0;
var Produced_Beta >=0;
var Produced_Gamma >=0;
var Used_Line_A binary;
var Used_Line_C binary;

maximize Income_in_Dollars:
410 * Produced_Alpha + 520 * Produced_Beta + 686 *
   Produced_Gamma - 2016 * Used_Line_A - 1200 * Used_Line_C
   - 32 * Produced_Alpha - 32 * Produced_Beta - 38.5 *
   Produced_Gamma;
```

Example 4.1 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta - 120 * Used_Line_A <= 0;

subto Max_Capacity_of_Production_Line2:
 Produced_Gamma - 48 * Used_Line_C <= 0;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000;
```

Example 4.1 by ZIMPL results

```
Optimal - objective value                -68044
      0 Produced_Alpha                     120
      1 Produced_Beta                        0
      2 Produced_Gamma                      40
      3 Used_Line_A                          1
      4 Used_Line_C                          1
```

Conditional minimum

- A constraint sometimes linked to a decision about whether or not to incur a fixed cost is the conditional minimum
- This states that a continuous variable must either equal zero or be above a certain threshold
- For example, Ajax's production manager might wish to impose the constraint that testing of Gammas in the week must either be 0 or at least 5
  - $M_C - 5F_C \geq 0$
    - $F_C = 1 \rightarrow M_C \geq 5$

4.1.2. ECONOMIES OF SCALE

Introduction

- In section 3.2 we stated that an MILP model is needed to correctly portray economies of scale in cost and resource utilization

Example 4.2 (example 3.8)

- We have one vendor RC
  - $250/processor up to 30 pieces
  - $200/processor for additional deliveries
  - The capacity of RC is 48 pieces/week

Decision variables

- W_1 : weekly microprocessor purchase from RC at \$250/unit
- W_2 : weekly microprocessor purchase from RC at \$200/unit
- D : equals 1 if economy of scale achieved; equals to 0 if economy of scale not achieved

Model

- New constraints
  - $0 \le W_1 \le 30$
  - $W_2 \ge 0$
  - $- W_1 + 30D \le 0$
    - $D = 0 \rightarrow W_1 \ge 0 \rightarrow 0 \le W_1 \le 30$
    - $D = 1 \rightarrow W_1 \ge 30 \rightarrow W_1 = 30$
  - $W_2 - 18D \le 0$
    - $D = 0 \rightarrow W_2 \le 0 \rightarrow W_2 = 0$
    - $D = 1 \rightarrow W_2 \le 18 \rightarrow W_1 + W_2 \le 48$

Example 4.2 by ZIMPL

```
# Example 4.2
# Imported processors from only one vendor, increased
  capacity and changed its prices

#sets
set computers := {"Alpha", "Beta", "Gamma"};
set resources := {"A-line", "C-line", "manpower"};
set vendors := {"RC"};
```

Example 4.2 by ZIMPL cont.

```
#parameters
param netvalue[computers] := <"Alpha"> 350, <"Beta">
   470, <"Gamma"> 810;
param availability[resources] := <"A-line"> 120, <"C-
   line"> 48, <"manpower"> 2000;
param usage[computers*resources] :=
           |"A-line", "C-line", "manpower"|
   |"Alpha"|   1,         0,          10     |
   |"Beta" |   1,         0,          15     |
   |"Gamma"|   0,         1,          20     |;
param CostOfProcessorBelowLimit[vendors] := <"RC">
   250;
param CostOfProcessorAboveLimit[vendors] := <"RC">
   200;
param EconomyLimit[vendors] := <"RC"> 30;
param UsedProcessors[computers] := <"Gamma"> 1 default
   0;
param Capacity[vendors] := <"RC"> 48;
```

Example 4.2 by ZIMPL cont.

```
# variables
var Produced[computers] >=0;
var PurchasedBelowLimit[vendors] >=0;
var PurchasedAboveLimit[vendors] >=0;
var D binary; #The economy limit has been reached

#objective
maximize Income_in_Dollars:
sum <c> in computers : netvalue[c] * Produced[c]
   - sum <v> in vendors : CostOfProcessorBelowLimit[v]
   * PurchasedBelowLimit[v]
   - sum <v> in vendors : CostOfProcessorAboveLimit[v]
   * PurchasedAboveLimit[v];
```

Example 4.2 by ZIMPL cont.

```
#constraints
subto Resource_constraints:
forall <r> in resources do
   sum <c> in computers : usage[c,r] * Produced[c] <=
   availability[r];

subto Vendors_EconomyLimit_constraints:
forall <v> in vendors do
   PurchasedBelowLimit[v] <= EconomyLimit[v];

subto Vendors_EconomyLimit_logical_constraints:
forall <v> in vendors do
   PurchasedBelowLimit[v] >= EconomyLimit[v] * D;
```

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

Example 4.2 by ZIMPL cont.

```
subto Proc_for_computers:
sum <v> in vendors : (PurchasedBelowLimit[v] +
   PurchasedAboveLimit[v]) ==
   sum <p> in computers : Produced[p] *
   UsedProcessors[p];

subto EconomyLimit_constraint:
forall <v> in vendors do
   PurchasedAboveLimit[v] <= (capacity[v] -
   EconomyLimit[v]) * D;
```

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

4.1.3. PRODUCTION CHANGEOVERS

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- Setup resource utilization due to the changeover that occurs on the A-line in any week when both Alphas and Betas are tested
  - The changeover takes 20 hours

Example 4.3

- Decision variables
  - T_A : equals 1 if Alphas are tested in a given week; 0 otherwise
  - T_B : equals 1 if Betas are tested in a given week; 0 otherwise
  - T_2 : equals 1 if Alphas and Betas are tested in a given week; 0 otherwise

Model

- New constraints
  - $T_2 \geq T_A + T_B - 1$
    - $T_2 = 1$, if $T_A = 1$ and $T_B = 1$
  - $M_A + M_B + 20T_2 \leq 120$
  - $M_A - 120T_A \leq 0$
    - $T_A = 0 \rightarrow M_A \leq 0 \rightarrow M_A = 0$
    - $T_A = 1 \rightarrow M_A \leq 120$
  - $M_B - 120T_B \leq 0$

$\max Z = 350M_A + 470M_B + 610M_C$
s.t.
$M_A + M_B \leq 120$
$M_C \leq 48$
$10M_A + 15M_B + 20M_C \leq 2000$

Example 4.3 by ZIMPL

```
# Example 4.3
# Setup resource utilization due to the changeover that
   occurs on the A-line in any week when both Alphas and
   Betas are tested

var Produced_Alpha >=0;
var Produced_Beta >=0;
var Produced_Gamma >=0;
var Tested_Alpha binary;
var Tested_Beta binary;
var Tested_Both binary;

maximize Income_in_Dollars:
350 * Produced_Alpha + 470 * Produced_Beta + 610 *
   Produced_Gamma;
```

Example 4.3 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta + 20 * Tested_Both <= 120;
subto Max_Capacity_of_Production_Line2:
 Produced_Gamma <= 48;
subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000;

subto Test2:
Tested_Both >= Tested_Alpha + Tested_Beta -1;
subto Alpha:
Produced_Alpha - 120 * Tested_Alpha <= 120;
subto Beta:
Produced_Beta - 120 * Tested_Beta <= 120;
```

Example 4.3 by ZIMPL results

```
Optimal - objective value              -66400
      0 Produced_Alpha                   120
      1 Produced_Beta                      0
      2 Produced_Gamma                    40
      3 Tested_Both                        0
      4 Tested_Beta                        0
      5 Tested_Alpha                       0
```

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

4.1.4. NON-NUMERICAL CONSTRAINTS

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- Binary variables can be used for non-numerical or logical constraints
- A frequently occurring constraint of this type is the multiple-choice constraint
  - Exactly one, or at most one, logical decision must be selected from among a set of possible logical decisions

Example 4.4 (example 3.5)

- The distribution manager at Ajax would like to investigate the consequences of limiting shipments to market 8 and all other markets to a single source

- Transportation data of Alpha computers
  - Plant at Chicago (100 units of Alpha)
  - Warehouse in St. Louis (45 units of Alpha)
  - 8 markets

Model

- Decision variables
  - D_{P8} : equals 1 if the plant serves market 8; 0 otherwise
  - D_{W8} : equals 1 if the warehouse serves market 8; 0 otherwise

Model

- Change ($X_{P8} \to 20D_{P8}$; $X_{W8} \to 20D_{W8}$)
  - $20D_{P8} + 20D_{W8} = 20 \to D_{P8} + D_{W8} = 1$
  - $X_{P1} + X_{P2} + X_{P3} + X_{P4} + X_{P5} + X_{P6} + X_{P7} + 20D_{P8} \leq 100$
  - $X_{W1} + X_{W2} + X_{W3} + X_{W4} + X_{W5} + X_{W6} + X_{W7} + 20D_{W8} \leq 45$
  - $\min Z = 14X_{P1} + \cdots + 17X_{P7} + 600D_{P8} + 24X_{W1} + \cdots + 24X_{W7} + 560D_{W8}$

Example 4.4 by ZIMPL

```
# Example 4.4
# Like Example 3.5 with limiting shipments to market
  Minnepolis and all other markets to a single source

var A_Plant_Chicago integer >=0;
var A_Plant_Seatle integer >=0;
var A_Plant_Detroit integer >=0;
var A_Plant_Cincinnati integer >=0;
var A_Plant_Louisiana integer >=0;
var A_Plant_Indianapolis integer >=0;
var A_Plant_Milwaukee integer >=0;
```

Example 4.4 by ZIMPL cont.

```
var A_Warehouse_Chicago integer >=0;
var A_Warehouse_Seatle integer >=0;
var A_Warehouse_Detroit integer >=0;
var A_Warehouse_Cincinnati integer >=0;
var A_Warehouse_Louisiana integer >=0;
var A_Warehouse_Indianapolis integer >=0;
var A_Warehouse_Milwaukee integer >=0;

var D_Plant_Minneapolis binary;
var D_Warehouse_Minneapolis binary;
```

Example 4.4 by ZIMPL cont.

```
minimize Transfer_cost:
14 * A_Plant_Chicago + 24 * A_Plant_Seatle + 21 *
   A_Plant_Detroit + 20 * A_Plant_Cincinnati + 21.50 *
   A_Plant_Louisiana + 19 * A_Plant_Indianapolis + 17 *
   A_Plant_Milwaukee + 30 * 20 * D_Plant_Minneapolis +
24 * A_Warehouse_Chicago + 15 * A_Warehouse_Seatle + 28 *
   A_Warehouse_Detroit + 20 * A_Warehouse_Cincinnati + 18.50
   * A_Warehouse_Louisiana + 19.50 *
   A_Warehouse_Indianapolis + 24 * A_Warehouse_Milwaukee +
   28 * 20 * D_Warehouse_Minneapolis;

subto Max_capacity_of_Plant:
A_Plant_Chicago + A_Plant_Seatle + A_Plant_Detroit +
   A_Plant_Cincinnati + A_Plant_Louisiana +
   A_Plant_Indianapolis + A_Plant_Milwaukee +
20 * D_Plant_Minneapolis <= 100;
```

Example 4.4 by ZIMPL cont.

```
subto Max_capacity_of_Warehouse:
A_Warehouse_Chicago + A_Warehouse_Seatle +
   A_Warehouse_Detroit + A_Warehouse_Cincinnati +
   A_Warehouse_Louisiana + A_Warehouse_Indianapolis +
   A_Warehouse_Milwaukee +
20 * D_Warehouse_Minneapolis <= 45;

subto Need_in_Chicago:
A_Plant_Chicago + A_Warehouse_Chicago == 22;
subto Need_in_Seatle:
A_Plant_Seatle + A_Warehouse_Seatle == 14;
subto Need_in_Detroit:
A_Plant_Detroit + A_Warehouse_Detroit == 18;
subto Need_in_Cincinnati:
A_Plant_Cincinnati + A_Warehouse_Cincinnati == 17;
```

Example 4.4 by ZIMPL cont.

```
subto Need_in_Louisiana:
A_Plant_Louisiana + A_Warehouse_Louisiana == 15;
subto Need_in_Indianapolis:
A_Plant_Indianapolis + A_Warehouse_Indianapolis == 13;
subto Need_in_Milwaukee:
A_Plant_Milwaukee + A_Warehouse_Milwaukee == 15;
subto Need_in_Minneapolis:
D_Plant_Minneapolis + D_Warehouse_Minneapolis == 1;
```

Example 4.4 by ZIMPL results

```
Optimal - objective value              2587.5
      0 A_Plant_Chicago                  22
      1 A_Plant_Seatle                    0
      2 A_Plant_Detroit                  18
      3 A_Plant_Cincinnati               17
      4 A_Plant_Louisiana                 4
      5 A_Plant_Indianapolis             13
      6 A_Plant_Milwaukee                15
      7 A_Warehouse_Chicago               0
      8 A_Warehouse_Seatle               14
      9 A_Warehouse_Detroit               0
     10 A_Warehouse_Cincinnati            0
     11 A_Warehouse_Louisiana            11
     12 A_Warehouse_Indianapolis          0
     13 A_Warehouse_Milwaukee             0
     14 D_Plant_Minneapolis               0
     15 D_Warehouse_Minneapolis           1
```

Example 4.5

- The management policy requires
  - No more than 3 markets can be served by the warehouse
    - $D_{W1} + D_{W2} + D_{W3} + D_{W4} + D_{W5} + D_{W6} + D_{W7} + D_{W8} \leq 3$
  - Markets 4 and 6 must both be served either by the warehouse or the plant
    - $D_{W4} - D_{W6} = 0$
    - $D_{P4} + D_{W4} = 1$
    - $D_{P6} + D_{W6} = 1$

Example 4.5 by ZIMPL

```
# Example 4.5
# Like Example 4.4 but no more than 3 markets can be served
   by the warehouse and markets of Cincinnati and
   Indianapolis must both be served either by the warehouse
   or the plant

var A_Plant_Chicago integer >=0;
var A_Plant_Seatle integer >=0;
var A_Plant_Detroit integer >=0;
var A_Plant_Cincinnati integer >=0;
var A_Plant_Louisiana integer >=0;
var A_Plant_Indianapolis integer >=0;
var A_Plant_Milwaukee integer >=0;
var A_Plant_Minneapolis integer >=0;
```

Example 4.5 by ZIMPL cont.

```
var A_Warehouse_Chicago integer >=0;
var A_Warehouse_Seatle integer >=0;
var A_Warehouse_Detroit integer >=0;
var A_Warehouse_Cincinnati integer >=0;
var A_Warehouse_Louisiana integer >=0;
var A_Warehouse_Indianapolis integer >=0;
var A_Warehouse_Milwaukee integer >=0;
var A_Warehouse_Minneapolis integer >=0;

var D_Plant_Cincinnati binary;
var D_Plant_Indianapolis binary;
```

Example 4.5 by ZIMPL cont.

```
var D_Warehouse_Chicago binary;
var D_Warehouse_Seatle binary;
var D_Warehouse_Detroit binary;
var D_Warehouse_Cincinnati binary;
var D_Warehouse_Louisiana binary;
var D_Warehouse_Indianapolis binary;
var D_Warehouse_Milwaukee binary;
var D_Warehouse_Minneapolis binary;
```

Example 4.5 by ZIMPL cont.

```
minimize Transfer_cost:
14 * A_Plant_Chicago + 24 * A_Plant_Seatle + 21 *
   A_Plant_Detroit + 20 * A_Plant_Cincinnati + 21.50 *
   A_Plant_Louisiana + 19 * A_Plant_Indianapolis + 17 *
   A_Plant_Milwaukee + 30 * A_Plant_Minneapolis +
24 * A_Warehouse_Chicago + 15 * A_Warehouse_Seatle + 28 *
   A_Warehouse_Detroit + 20 * A_Warehouse_Cincinnati + 18.50
   * A_Warehouse_Louisiana + 19.50 *
   A_Warehouse_Indianapolis + 24 * A_Warehouse_Milwaukee +
   28 * A_Warehouse_Minneapolis;


subto Max_capacity_of_Plant:
A_Plant_Chicago + A_Plant_Seatle + A_Plant_Detroit +
   A_Plant_Cincinnati + A_Plant_Louisiana +
   A_Plant_Indianapolis + A_Plant_Milwaukee +
   A_Plant_Minneapolis <= 100;
```

Example 4.5 by ZIMPL cont.

```
subto Max_capacity_of_Warehouse:
A_Warehouse_Chicago + A_Warehouse_Seatle +
   A_Warehouse_Detroit + A_Warehouse_Cincinnati +
   A_Warehouse_Louisiana + A_Warehouse_Indianapolis +
   A_Warehouse_Milwaukee + A_Warehouse_Minneapolis <= 45;

subto Need_in_Chicago:
A_Plant_Chicago + A_Warehouse_Chicago == 22;
subto Need_in_Seatle:
A_Plant_Seatle + A_Warehouse_Seatle == 14;
subto Need_in_Detroit:
A_Plant_Detroit + A_Warehouse_Detroit == 18;
subto Need_in_Cincinnati:
A_Plant_Cincinnati + A_Warehouse_Cincinnati == 17;
subto Need_in_Louisiana:
A_Plant_Louisiana + A_Warehouse_Louisiana == 15;
```

Example 4.5 by ZIMPL cont.

```
subto Need_in_Indianapolis:
A_Plant_Indianapolis + A_Warehouse_Indianapolis == 13;
subto Need_in_Milwaukee:
A_Plant_Milwaukee + A_Warehouse_Milwaukee == 15;
subto Need_in_Minneapolis:
A_Plant_Minneapolis + A_Warehouse_Minneapolis == 20;

subto Chicago:
A_Warehouse_Chicago <= 22 * D_Warehouse_Chicago;
subto Seatle:
A_Warehouse_Seatle <= 14 * D_Warehouse_Seatle;
subto Detroit:
A_Warehouse_Detroit <= 18 * D_Warehouse_Detroit;
subto Cincinnati:
A_Warehouse_Cincinnati <= 17 * D_Warehouse_Cincinnati;
```

Example 4.5 by ZIMPL cont.

```
subto Louisiana:
A_Warehouse_Louisiana <= 15 * D_Warehouse_Louisiana;
subto Indianapolis:
A_Warehouse_Indianapolis <= 13 * D_Warehouse_Indianapolis;
subto Milwaukee:
A_Warehouse_Milwaukee <= 15 * D_Warehouse_Milwaukee;
subto Minneapolis:
A_Warehouse_Minneapolis <= 20 * D_Warehouse_Minneapolis;

subto Plant_Cin:
A_Plant_Cincinnati <= 17 * D_Plant_Cincinnati;
subto Plant_Ind:
A_Plant_Indianapolis <= 13 * D_Plant_Indianapolis;
```

Example 4.5 by ZIMPL cont.

```
subto Warehouse:
D_Warehouse_Chicago + D_Warehouse_Seatle +
   D_Warehouse_Detroit + D_Warehouse_Cincinnati +
   D_Warehouse_Louisiana + D_Warehouse_Indianapolis +
   D_Warehouse_Milwaukee +
   D_Warehouse_Minneapolis <= 3;

subto Cin_Ind:
D_Warehouse_Cincinnati - D_Warehouse_Indianapolis == 0;

subto Cin:
D_Plant_Cincinnati + D_Warehouse_Cincinnati == 1;
subto Ind:
D_Plant_Indianapolis + D_Warehouse_Indianapolis == 1;
```

Example 4.5 by ZIMPL results

```
Optimal - objective value           2583.5
      0 A_Plant_Chicago                 22
      1 A_Plant_Seatle                   0
      2 A_Plant_Detroit                 18
      3 A_Plant_Cincin@3                17
      4 A_Plant_Louisi@4                 0
      5 A_Plant_Indian@5                13
      6 A_Plant_Milwau@6                15
      7 A_Plant_Minnea@7                 4
      8 A_Warehouse_Ch@8                 0
      9 A_Warehouse_Se@9                14
     10 A_Warehouse_De@a                 0
     11 A_Warehouse_Ci@b                 0
     12 A_Warehouse_Lo@c                15
     13 A_Warehouse_In@d                 0
     14 A_Warehouse_Mi@e                 0
```

Example 4.5 by ZIMPL results cont.

```
15 A_Warehouse_Mi@f              16
16 D_Warehouse_C@12               0
17 D_Warehouse_S@13               1
18 D_Warehouse_D@14               0
19 D_Warehouse_C@15               0
20 D_Warehouse_L@16               1
21 D_Warehouse_I@17               0
22 D_Warehouse_M@18               0
23 D_Warehouse_M@19               1
24 D_Plant_Cinci@10               1
25 D_Plant_India@11               1
```

4.2. DISTRIBUTION CENTER LOCATION MODELS

Introduction

- In a standard version of this problem, a distribution company must design the network of DCs from which to ship its products to its markets to meet forecasted demand
- The objective is to minimize the sum of warehousing and transportation costs

Example 4.6

- The Electronica Corporation is a wholesale distributor of consumer electronic products
  - 20 markets
  - 8 potential DC locations (rentable)
    - No DC, small DC, large DC
    - Fix and linear costs

Parameters

- D_{ij} : distance between DC *i* and market *j*
- F_{iS} : fixed cost of small DC at location *i*
- F_{iL} : fixed cost of large DC at location *i*
- V_{iS} : linear cost of small DC at location *i*
- V_{iL} : linear cost of large DC at location *i*
- C_{iS} : capacity of small DC at location *i*
- C_{iL} : capacity of large DC at location *I*
- R_j : demand in market *j*
- T : cost per truckload-mile

Decision variables

- $X_{ij\mathrm{S}}$: product flow from small DC *i* to market *j*
- $X_{ij\mathrm{L}}$: product flow from large DC *i* to market *j*
- S_i : equals 1 if a small DC is selected at location *i*; 0 otherwise
- L_i : equals 1 if a large DC is selected at location *i*; 0 otherwise

Model

$$\min Z = T(D_{11}(X_{11S} + X_{11L}) + \cdots + D_{8,20}(X_{8,20,S} + X_{8,20,L})) + S_1F_{1S} + L_1F_{1L} + \cdots + S_8F_{8S} + L_8F_{8L} + V_{1S}(X_{11S} + \cdots + X_{1,20,S}) + \cdots + V_{8L}(X_{81L} + \cdots + X_{8,20,L})$$

s.t.

$$S_i + L_i \le 1 \qquad (i = 1, 2, \ldots, 8)$$

$$X_{i1S} + \cdots + X_{i,20,S} - S_iC_{iS} \le 0 \quad (i = 1, 2, \ldots, 8)$$

$$X_{i1L} + \cdots + X_{i,20,L} - L_iC_{iL} \le 0 \quad (i = 1, 2, \ldots, 8)$$

$$X_{1jS} + \cdots + X_{8jL} = R_j \qquad (j = 1, 2, \ldots, 20)$$

What-if scenarios

- If the model has been constructed and optimized the managers can check different scenarios
  - For example, rent at most 4 DC
    - $S_1 + L_1 + \cdots + S_8 + L_8 \leq 4$

Generalization

- The pervious DC location model is simple, although nontrivial
- It can be generalized in a number of ways
  – More products
  – Multilevel DC network
  – Different cost per truckload-mile for each link in the network

4.3. SUPPLY CHAIN NETWORK OPTIMIZATION MODELS

Introduction

- We have used different models for
  - Transportation
  - Resource utilization
- We need an integrated model

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

4.3.1. STRATEGIC PLANNING AT AJAX

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

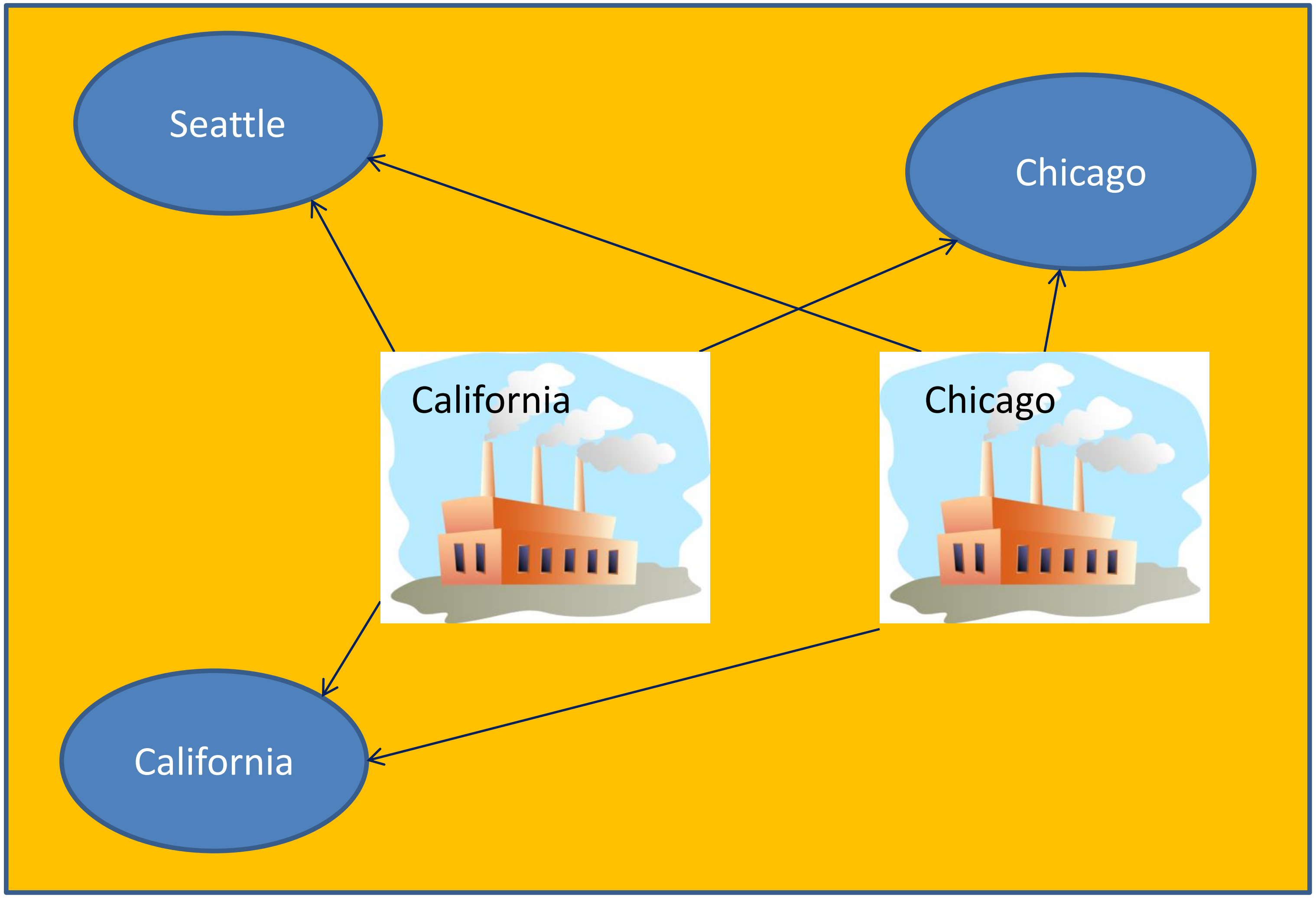
- We construct an MILP model to evaluate strategic planning options over the next 3 years
- A 3-years forecast is needed

Strategic possibilities

- New assembly plant in a new location (Sunnyvale, California)
  - In what year
- Major expansion of existing assembly plant
  - In what year
- Investment in development of a new product (Delta)
  - Where should it be assembled
- What quantity of each product should be assembled at each plant in each time period
  - Which plant should serve which market

4.3.1.1. CONSTRUCTING AN INTEGRATED SUPPLY CHAIN MODEL

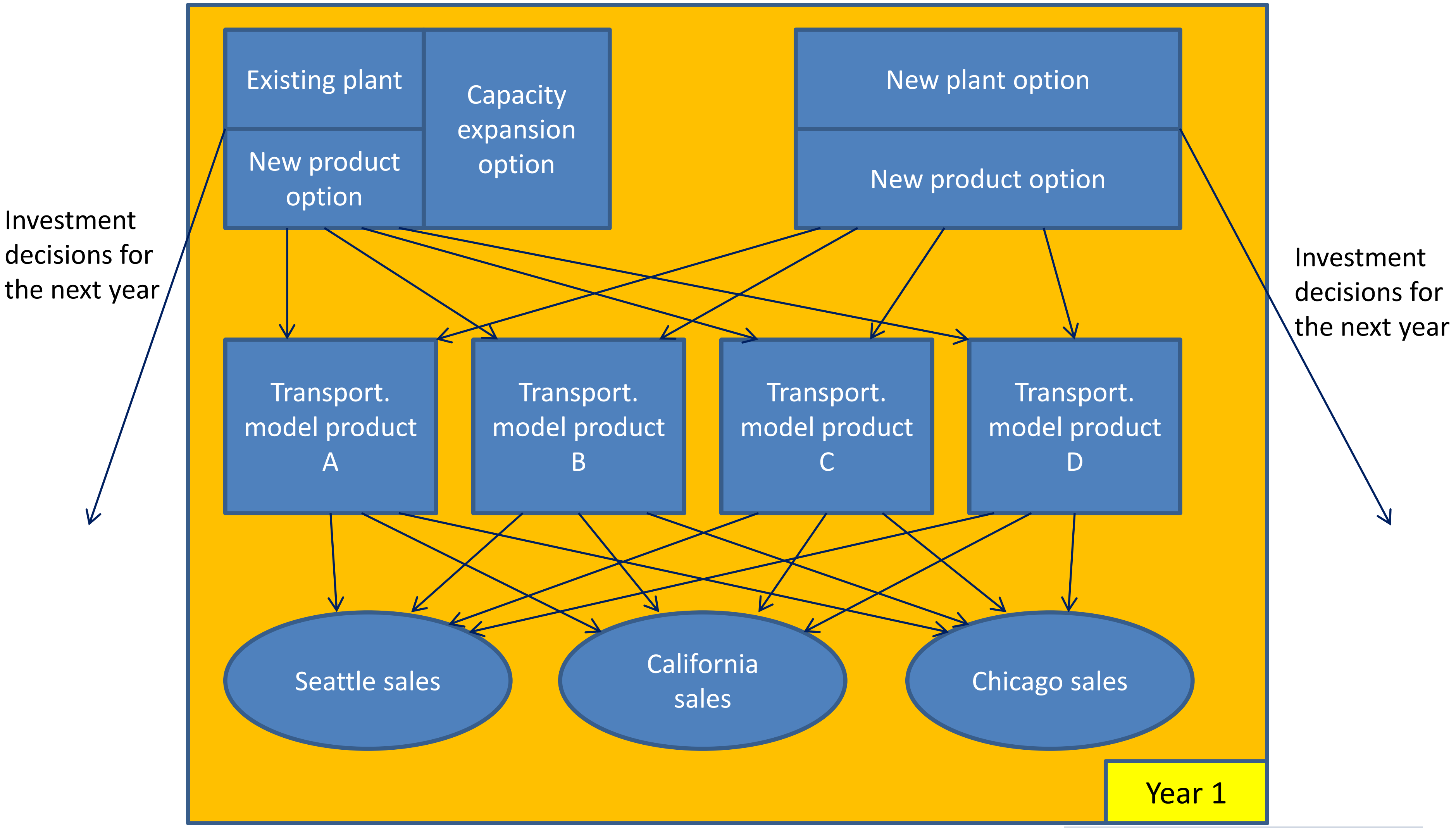
Strategic SCN



Submodels

- Production model (existing plant, expand capacity, new product)
- Production model (new plant, new product)
- Transportation model
- Sales model (maximal limits of markets)
- Multiple-choice constraints
- Net revenue functions for each year

Modeling objects


Existing plant
New product option
Capacity expansion option
New plant option
New product option
Investment decisions for the next year
Investment decisions for the next year
Transport. model product A
Transport. model product B
Transport. model product C
Transport. model product D
Seattle sales
California sales
Chicago sales
Year 1

Decision variables of production

- M_{pXt} : number of product p assembled at existing plant in year t (t = 1, 2, 3; p = A, B, C, D)
- X_t : equals 1 if existing plant is expanded in year t; 0 otherwise (t = 1, 2, 3)
- D_{Xt} : equals 1 if new product developed and assembly at existing plant in year t; 0 otherwise (t = 1, 2, 3)
- D_{Nt} : equals 1 if new product developed and assembly at new plant in year t; 0 otherwise (t = 1, 2, 3)

Constraints

- The existing plant can be extended at most 1 of the 3 years
  - $X_1 + X_2 + X_3 \leq 1$
- The new product can be developed for assembly at the existing plant, the new plant, or neither
  - $D_{X1} + D_{N1} + D_{X2} + D_{N2} + D_{X3} + D_{N3} \leq 1$
- Resource capacity constraints (example for labor hours)
  - $10M_{AX1} + 15M_{BX1} + 20M_{CX1} + 22M_{DX1} - 33{,}000X_1 \leq 100{,}000$
- Production of Delta (example for year 1, forecast: 1100 unit)
  - $M_{DX1} - 1100 * D_{X1} \leq 0$

Decision variables of new plant

- $M_{p\mathrm{N}t}$: number of product p assembled at the new plant in year t (t = 1, 2, 3; p = A, B, C, D)
- N_t : equals 1 if the new plants is opened in year t; 0 otherwise (t = 1, 2, 3)

Constraints

- The new plant can be opened at most 1 of the 3 years
  - $N_1 + N_2 + N_3 \leq 1$

Objective function

- Max $Z = Z_1 + 0.9Z_2 + 0.81Z_3$
  - Ajax discounts cash flow at 10% per year
- Z_t = gross revenue from sales – production costs – transportation costs – investment cost in expanding the existing plant – investment cost in constructing the new plant – investment cost in developing the new product (t = 1, 2, 3)

4.3.1.2. BASE CASE AND SCENARIO ANALYSES

Introduction

- The timing and sizing of the investments allow maximal sales is called base case
- Senior management will seek to refine their intuition by making additional runs with the model
  - The additional runs will be based on scenarios analyses or what-if questions

Base case

Option	Decision
Build new plant	Build immediately for use in year 1
Expand existing plant	Expand for use in year 2
Develop new product	Reject

Scenarios

1. Delay construction of the new plant to year 2 or 3
2. Maximal potential sales are 20% higher than forecast
3. Maximal potential sales are 20% lower than forecast
4. Force acceptance of the new product
5. New Delta design
   – More expensive product

Results

Description	Base case	1	2	3	4	5
Expansion of existing plant	Year 2	Year 1	Year 2	Rejected	Year 2	Year 2
Constructing new plant	Year 1	Year 2	Year 1	Year 1	Year 1	Year 1
New product development	Rejected	Rejected	Rejected	Rejected	At new plant	At new plant
Net revenues	12,734,301	12,006,545	13,100,828	11,619,773	12,196,501	14,375,469

1. Delay construction of the new plant to year 2 or 3
2. Maximal potential sales are 20% higher than forecast
3. Maximal potential sales are 20% lower than forecast
4. Force acceptance of the new product
5. New Delta design

4.4. OPTIMIZATION SOFTWARE

Introduction

- The first LP packages appeared during the late 1950s
- The first MILP packages appeared during the early 1970s
- Their capabilities have evolved with advanced information technology as well as optimization algorithms
- Optimizers are packages containing numerical algorithms to produce an optimal or near optimal solution

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

4.4.1. OPTIMIZERS

LP algorithms

- Simplex method
  - Efficient in reoptimization
    - Scenarios
    - Branch-and-bound method
  - Different implementations and efficiency
- Interior point method
  - Less sensitivity for numerical errors

MILP algorithms

- Solving MILP models is less predictable
  - Modeling expert
  - Demonstrably good solution
  - Tailored to exploit the structure
  - Built-in commercial B&B

Modeling expert

- The structure of model
  - Binary variables for major decisions
  - Binary variables should not be created for relatively very small fixed cost
    - Fixed costs that represents less than .001 of total facility cost
  - Small economies of scale should not be explicitly modeled

Demonstrably good solution

- Easier to find a demonstrably good solution than an optimal one
- Branch-and-bound method determines bounds on how far the cost value if the best-known solution is from optimality
  - A solution with a small positive tolerant (for example, 0.5%) may actually be optimal, but the search has not yet proven it so

Tailored to exploit the structure

- Use of branching priorities based on the relative importance
  - Plant location decisions
- Branching refers to fixing these variables

Built-in commercial B&B

- Commercial MILP code vary significantly in the efficiency and flexibility of their B&B methods
- To solve a MILP model takes an order of magnitude more time than an LP with the same size

Limit on model size

- Many commercial packages allow an unlimited number of variables and constraints
  - This feature can be dangerous
- A supply chain model may contain very large but well-behaved submodels
  - For example: SC transportation problem with 20 DCs, 500 markets and 10 products
    - Number of constraints ≈ 50,000
    - Number of variables ≈ 250,000
    - Solvable because of its simple structure
      - Modeling expert needed

Availability of callable library

- There exists routines for commercial packages which are callable from other programs
- Specialized branch-and-bound algorithms
- Fully self-developed algorithms may not work efficiently
  - Professional libraries can help

4.5. EXERCISES

Exercise 4.1

- Regular-time assembly labor at Ajax Computer is a sunk cost providing up to a maximum of 2000 hours/week
- Overtime costs $25/hour up to a maximum of 400 hours/week
- There is also a fixed cost of $250/week if any overtime is incurred
- Extend Example 4.1 to incorporate the overtime option for the 1-week model

Exercise 4.1 by ZIMPL

```
# Exercise 4.1

var Produced_Alpha >=0;
var Produced_Beta >=0;
var Produced_Gamma >=0;
var Used_Line_A binary;
var Used_Line_C binary;
var Overtime >=0;
var IsOvertime binary;

maximize Income_in_Dollars:
410 * Produced_Alpha + 520 * Produced_Beta + 686 *
   Produced_Gamma - 2016 * Used_Line_A - 1200 * Used_Line_C
   - 32 * Produced_Alpha - 32 * Produced_Beta - 38.5 *
   Produced_Gamma - 250 * IsOvertime - 25 * Overtime;
```

Exercise 4.1 by ZIMPL cont.

```
subto Max_Capacity_of_Production_Line1:
Produced_Alpha + Produced_Beta - 120 * Used_Line_A <= 0;

subto Max_Capacity_of_Production_Line2:
 Produced_Gamma - 48 * Used_Line_C <= 0;

subto Max_Capacity_of_Man_Power:
10 * Produced_Alpha + 15 * Produced_Beta + 20 *
   Produced_Gamma <= 2000 + Overtime;

subto Overtime_constraint:
Overtime <= 400 * IsOvertime;
```

Exercise 4.2

- Sigma Computer Company must purchase 600 customized hard drives for its laptop computer for next year
- As shown below, the company received quotes from three vendors for this drives

Company	Fixed cost ($)	Unit cost 1 ($)	Break point	Unit cost 2 ($)	Max available
Acme	15,000	240	200	200	500
Best	0	325	150	225	700
Champion	10,000	265	250	190	800

Exercise 4.2

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

- For example, Champion will require a fixed cost of $10,000 to tool up to make the drivers
  - It will charge $265/unit for the first 250 units and $190/unit for additional units up to a maximum (supplied at both prices) of 800 units

Company	Fixed cost ($)	Unit cost 1 ($)	Break point	Unit cost 2 ($)	Max available
Acme	15,000	240	200	200	500
Best	0	325	150	225	700
Champion	10,000	265	250	190	800

Exercise 4.2

a. Formulate Sigma’s vendor selection problem as a MILP model and optimize it

b. Reformulate and reoptimize the model of part (a) under the additional constraint that no vendor is allowed to supply more than 75% of the total number of units required

Exercise 4.2a by ZIMPL

```
# Exercise 4.2a


#sets
set vendors := {"Acme", "Best", "Champion"};
#parameters
param Demand := 600;
param FixedCost[vendors] := <"Acme"> 15000, <"Best">
   0, <"Champion"> 10000;
param UnitCost1[vendors] := <"Acme"> 240, <"Best">
   325, <"Champion"> 265;
param BreakPoint[vendors] := <"Acme"> 200, <"Best">
   150, <"Champion"> 250;
param UnitCost2[vendors] := <"Acme"> 200, <"Best">
   225, <"Champion"> 190;
param Capacity[vendors] := <"Acme"> 500, <"Best"> 700,
   <"Champion">  800;
```

Exercise 4.2a by ZIMPL cont.

```
#variables
var PurchasedBelow[vendors] >=0;
var PurchasedAbove[vendors] >=0;
var BreakReached[vendors] binary;
var IsPurchased[vendors] binary;

#cost function
minimize Overal_cost:
sum <v> in vendors : IsPurchased[v] * FixedCost[v] +
   sum <v> in vendors : PurchasedBelow[v] *
   UnitCost1[v] +
   sum <v> in vendors : PurchasedAbove[v] *
   UnitCost2[v];
```

Exercise 4.2a by ZIMPL cont.

```
#constraints
subto Purchase_logical_constraint:
forall <v> in vendors do
   PurchasedBelow[v] + PurchasedAbove[v] <=
   Capacity[v] * IsPurchased[v];

subto PurchaseBellow_capacity_constraint1:
forall <v> in vendors do
   PurchasedBelow[v] <= BreakPoint[v];

subto PurchaseBellow_capacity_constraint2:
forall <v> in vendors do
   PurchasedBelow[v] >= BreakPoint[v] *
   BreakReached[v];
```

Exercise 4.2a by ZIMPL cont.

```
subto PurchaseAbove_capacity_constraint1:
forall <v> in vendors do
   PurchasedAbove[v] <= (Capacity[v] - BreakPoint[v])
   * BreakReached[v];

subto Demand_Satisfying_constraint:
sum <v> in vendors : (PurchasedBelow[v] +
   PurchasedAbove[v]) == Demand;
```

Exercise 4.2b by ZIMPL

```
# Exercise 4.2b


#sets
set vendors := {"Acme", "Best", "Champion"};
#parameters
param Demand := 600;
param FixedCost[vendors] := <"Acme"> 15000, <"Best">
   0, <"Champion"> 10000;
param UnitCost1[vendors] := <"Acme"> 240, <"Best">
   325, <"Champion"> 265;
param BreakPoint[vendors] := <"Acme"> 200, <"Best">
   150, <"Champion"> 250;
param UnitCost2[vendors] := <"Acme"> 200, <"Best">
   225, <"Champion"> 190;
param Capacity[vendors] := <"Acme"> 500, <"Best"> 700,
   <"Champion">  800;
```

Exercise 4.2b by ZIMPL cont.

```
#variables
var PurchasedBelow[vendors] >=0;
var PurchasedAbove[vendors] >=0;
var BreakReached[vendors] binary;
var IsPurchased[vendors] binary;

#cost function
minimize Overal_cost:
sum <v> in vendors : IsPurchased[v] * FixedCost[v] +
   sum <v> in vendors : PurchasedBelow[v] *
   UnitCost1[v] +
   sum <v> in vendors : PurchasedAbove[v] *
   UnitCost2[v];
```

Exercise 4.2b by ZIMPL cont.

```
#constraints
subto Purchase_logical_constraint:
forall <v> in vendors do
   PurchasedBelow[v] + PurchasedAbove[v] <=
   Capacity[v] * IsPurchased[v];

subto PurchaseBellow_capacity_constraint1:
forall <v> in vendors do
   PurchasedBelow[v] <= BreakPoint[v];

subto PurchaseBellow_capacity_constraint2:
forall <v> in vendors do
   PurchasedBelow[v] >= BreakPoint[v] *
   BreakReached[v];
```

Exercise 4.2b by ZIMPL cont.

```
subto PurchaseAbove_capacity_constraint1:
forall <v> in vendors do
   PurchasedAbove[v] <= (Capacity[v] - BreakPoint[v])
   * BreakReached[v];

subto Demand_Satisfying_constraint:
sum <v> in vendors : (PurchasedBelow[v] +
   PurchasedAbove[v]) == Demand;

subto 0_75_constraint:
forall <v> in vendors do
   PurchasedBelow[v] + PurchasedAbove[v] <= Demand *
   0.75;
```

Exercise 4.3

- Home Grocery is a new company that makes same-day deliveries of groceries to people’s homes
- The company is launching its business in Metropolis, a large urban area
- The marketing department has identified eight neighborhoods in Metropolis where the company should concentrate its business

Exercise 4.3

- The logistic manager has identified six locations where the company may locate grocery depots
- The following table shows the average time (in minutes) required to travel from each of the six potential locations to the center of each of the eight neighborhoods
  - It also shows the target population (in thousands) for the company’s service in each neighborhood

Exercise 4.3

Neighborhoods	Depots						Population
	1	2	3	4	5	6	
1	15	17	27	5	25	22	12
2	10	12	24	4	22	20	8
3	5	6	17	9	21	17	11
4	7	6	8	15	13	10	14
5	14	12	6	23	6	8	22
6	18	17	10	28	9	5	18
7	11	10	5	21	10	9	16
8	24	22	22	33	6	16	20

Exercise 4.3

- The company wishes to locate two depots so that they maximize the population served within 12 minutes of average travel time
- Formulate and optimize the model of this problem

Exercise 4.3 by ZIMPL

```
# Exercise 4.3

#sets
set Neighborhoods := {1, 2, 3, 4, 5, 6, 7, 8};
set Locations := {1, 2, 3, 4, 5, 6};

#parameters
param Limit := 12;
param Population[Neighborhoods] := <1> 12, <2> 8, <3>
   11, <4> 14, <5> 22, <6> 18, <7> 16, <8> 20;
```

Exercise 4.3 by ZIMPL cont.

```
param AverageTime[Neighborhoods*Locations] :=
     |  1,  2,  3,  4,  5,  6|
   |1| 15, 17, 27,  5, 25, 22|
   |2| 10, 12, 24,  4, 22, 20|
   |3|  5,  6, 17,  9, 21, 17|
   |4|  7,  6,  8, 15, 13, 10|
   |5| 14, 12,  6, 23,  6,  8|
   |6| 18, 17, 10, 28,  9,  5|
   |7| 11, 10,  5, 21, 10,  9|
   |8| 24, 22, 22, 33,  6, 16|;
```

Exercise 4.3 by ZIMPL cont.

```
#variables
var Locate[Locations] binary;
var
   ReachableInLimitFromLocation[Neighborhoods*Location
   s] binary;
var ReachableInLimit[Neighborhoods] binary;

#cost function
maximize Reacable_population:
sum <n> in Neighborhoods : ReachableInLimit[n] *
   Population[n];
```

Exercise 4.3 by ZIMPL cont.

```
#constraints
subto Locate_2_depots:
sum <l> in Locations : Locate[l] == 2;

subto Reachable_in_Limit_from_location:
forall <n,l> in Neighborhoods cross Locations with
  AverageTime[n,l] <= Limit do
      ReachableInLimitFromLocation[n,l] == 1;

subto Not_reachable_in_Limit_from_location:
forall <n,l> in Neighborhoods cross Locations with
  AverageTime[n,l] > Limit do
  ReachableInLimitFromLocation[n,l] == 0;
```

Exercise 4.3 by ZIMPL cont.

```
subto Reachable_in_Limit:
forall <n> in Neighborhoods do
   ReachableInLimit[n] <= sum <l> in Locations :
   Locate[l] * ReachableInLimitFromLocation[n,l];

subto PurchaseAbove_capacity_constraint1:
forall <v> in vendors do
   PurchasedAbove[v] <= (Capacity[v] - BreakPoint[v])
   * BreakReached[v];

subto Demand_Satisfying_constraint:
sum <v> in vendors : (PurchasedBelow[v] +
   PurchasedAbove[v]) == Demand;
```

Exercise 4.4

- A mail-order company accepts telephone orders 7 days a week during the period 9 A.M. to 5 P.M. (local time)
- Based on analysis of historical data, the following number of people are needed each day in the call center to cover incoming orders

Sun	Mon	Tue	Wed	Thu	Fri	Sat
12	17	8	9	10	11	8

Exercise 4.4

- The company hires staff to work 5 consecutive days a week
- It pays these staff $100/day to work on Monday through Friday and $150/day to work on the weekends
- Develop and optimize an MILP model to minimize the cost of staffing the call center

Exercise 4.4 by ZIMPL

```
# Exercise 4.4


#sets
set Days := {0, 1, 2, 3, 4, 5, 6};


#parameters
param WorkingDays := 5;
param Needed[Days] := <0> 12, <1> 17, <2> 8, <3> 9,
   <4> 10, <5> 11, <6> 8;
param Payment[Days] := <0> 100, <1> 100, <2> 100, <3>
   100, <4> 100, <5> 150, <6> 150;
```

Exercise 4.4 by ZIMPL cont.

```
#variables
var Staff[Days] integer;
var HiredStaf[Days] integer;

#cost function
minimize Weekly_payment:
sum <d> in Days : Staff[d] * Payment[d];
```

Exercise 4.4 by ZIMPL cont.

```
#constraints
subto Staff_needed_for_each_day:
forall <d> in Days do
   Staff[d] >= Needed[d];

subto Staff_is_the_sum_of_the_previous_five_days:
forall <d> in Days do
   Staff[d] == sum <d2> in {0 to 2*card(Days)} with d2
   >= card(Days) + d - 5 and d2 <= d + card(Days) :
   HiredStaff[d2 mod card(Days)];
```

Exercise 4.5

- The distribution manager at Hudson's Pet Food Company wishes to analyze the company's plans for next quarter to ship its products from Hudson's plants to the DCs of retailers who sell the products to consumers
- Hudson uses third-party truckers to distribute its products
- The following table provides the distance that products must travel along with forecast of supply and demand

Exercise 4.5

From/To	Distances (miles)								Supply (1000 tons)
	DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8	
Plant 1	450	225	175	500	775	650	200	400	94
Plant 2	650	675	450	200	150	75	100	250	97
Plant 3	250	375	350	450	225	225	725	200	88
Demands (1000 tons)	45	36	38	24	29	19	31	44	

Exercise 4.5

- Further analysis of the distribution plans and third-party charges indicates the following rates will apply
  - A rate of $0.60/ton-mile for flows from plants to DCs in excess of 10,000 tons
  - A rate of $1.05/ton-mile for flows from plants to DCS below 10,000 tons
- The higher rate for the lower level of flow reflects the need to use more expensive LTL (less-than-truckload) shipments approximately 40% of the time
- The distribution manager is concerned that the tight supply quantities relative to forecasted demand will lead to an excessive number of LTL shipments

Exercise 4.5

- To address his concern, develop and solve an MILP model of Hudson’s distribution plan for next quarter
  - Hint: You need to define constraints employing binary variables for each plant/DC combination

Exercise 4.5 by ZIMPL

```
# Exercise 4.5


#sets
set Plants := {"Plant1", "Plant2", "Plant3"};
set DCs := {"DC1", "DC2", "DC3", "DC4", "DC5", "DC6", "DC7",
   "DC8"};


#parameters
param Demands[DCs] := <"DC1"> 45, <"DC2"> 36, <"DC3"> 38,
   <"DC4"> 24, <"DC5"> 29, <"DC6"> 19, <"DC7"> 31, <"DC8">
   44;
param Supply[Plants] := <"Plant1"> 94, <"Plant2"> 97,
   <"Plant3"> 88;
param BreakPoint := 10;
param RateBelowBreak := 1.05;
param RateAboveBreak := 0.6;
```

Exercise 4.5 by ZIMPL cont.

```
param Distance[Plants*DCs] :=
   |"DC1", "DC2", "DC3", "DC4", "DC5", "DC6", "DC7", "DC8"|
|"Plant1"| 450, 225, 175, 500, 775, 650, 200, 400 |
|"Plant2"| 650, 675, 450, 200, 150,  75, 100, 250 |
|"Plant3"| 250, 375, 350, 450, 225, 225, 725, 200 |;

#variables
var FlowBelow[Plants*DCs] >=0;
var FlowAbove[Plants*DCs] >=0;
var IsBelowBreak[Plants*DCs] binary;
```

Exercise 4.5 by ZIMPL cont.

```
#cost function
minimize Transportation_cost:
RateBelowBreak * sum <p,d> in Plants cross DCs :
   FlowBelow[p,d]
+ RateAboveBreak * sum <p,d> in Plants cross DCs :
   FlowAbove[p,d];

#constraints
subto Plants_constraint:
forall <p> in Plants do
    sum <d> in DCs : (FlowBelow[p,d] + FlowAbove[p,d]) <=
   Supply[p];

subto DCs_constraint:
forall <d> in DCs do
    sum <p> in Plants : (FlowBelow[p,d] + FlowAbove[p,d]) >=
   Demands[d];
```

Exercise 4.5 by ZIMPL cont.

```
subto FlowBelow_Logical_constraint:
forall <p,d> in Plants cross DCs do
   FlowBelow[p,d] <= BreakPoint * IsBelowBreak[p,d];

subto FlowAbove_logical_constraint1:
forall <p,d> in Plants cross DCs do
   (1 - IsBelowBreak[p,d]) * BreakPoint <= FlowAbove[p,d];

subto FlowAbove_logical_constraint2:
forall <p,d> in Plants cross DCs do
   FlowAbove[p,d] <= (1 - IsBelowBreak[p,d]) * sum <d2> in
   DCs : Demands[d2];
```

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

5. UNIFIED OPTIMIZATION METHODOLOGY FOR OPERATIONAL PLANNING PROBLEMS

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- Solving large-scale operational planning model is difficult
  - Detailed model → lots of variable
    - Sequencing decisions
    - Timing decisions

Example

- Paper mill
  - 30 days planning horizon
  - 4 product families
- MILP model
  - 12 binary variables for each hour → 8640 binary variables
  - In the optimal solution no more than 5 binary variables are equal 1
- Design a model that is less direct but much easier to optimize

Example

- Vehicle-routing problem
  - A company dispatches trucks on a daily basis from a central depot
  - The routing of vehicles
    - Starts from central depot
    - Stops at multiple customers
      - Each costumer is visited exactly once
    - Arrives back to depot
  - It is solvable using heuristics

Heuristic

- Heuristic are ad hoc search methods customized to a specific problem
- Based on rules gleaned by humans about the problem
- Heuristics can find good solution very quickly

Unified optimization methodology

- The presented unified optimization methodology combines mathematical programming decomposition methods with heuristics

5.1. HEURISTIC METHODS

Introduction

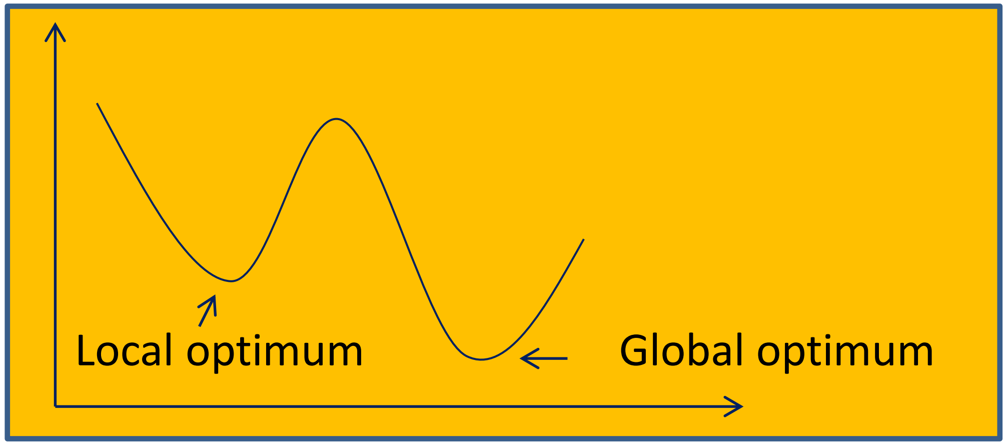
- Heuristic methods are myopic search methods that attempt to quickly find a good solution
- Problem-specific heuristics
  - They are not guaranteed to find an optimal or even a feasible solution
- General-purpose heuristics
  - Intelligently search the space of solution
  - Can be combined with problem-specific heuristics

Combinatorial optimization problems

- Selection of an optimal combination of objects from a discrete set of possible objects
- For example
  - Production
    - Manufacturing 15 products among 200 possible products on a machine
    - 10^{12} feasible schedule

Solution

- MILP solver
  - It may take lot of time
- Heuristic methods
  - Constraints are poorly handled
  - Can become trapped at a local optimum



5.1.1. LOCAL DELIVERY HEURISTICS

Introduction

- Local delivery problem of Chemtech distribution company
  - They have more than 50 depots located around the US
  - We investigate only one depot
- Problem-specific and general-purpose heuristics will be illustrated

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

5.1.1.1. DETAILS OF LOCAL DELIVERY PROBLEM

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Example 5.1

- 13 customers (denoted by 1, 2, …, 13)
- 1 depot (denoted by 0)
- Trucks may not travel distances greater than 60 miles between two customers

Distance matrix

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Customer demands

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

Trucks

- We have "unlimited" number of trucks
- The capacity of a track is 10 tons
- Each truck costs $100 per day in overhead expense to put on the road (insurance, …)
- The direct cost of a truck is $1 per mile (gasoline, …)

Feasible route

- Feasible route of a truck
  - Leaves the depot with 10 tons or less of product
  - Delivers to one or more customers
  - Does not visit consecutive customers who are more than 60 miles apart
  - Returns to the depot
- Feasible solution
  - Every customer is visited exactly once by some feasible route

Combinatorial property

- Selection of a combination of customers for a feasible route
- Selection of a combination of feasible routes in a feasible routing solution
  – Approximately 750 feasible routes exists

5.1.1.2. PROBLEM-SPECIFIC HEURISTICS

Traveling salesman problem

- Traveling salesperson wants to travel a minimal distance
  - Starts at city 1
  - Seeks a route that visit K – 1 other cities
    - Exactly once
  - Returns to city 1
- Well-known, difficult to optimize problem
- It can be described in at least three distinct ways as an MILP model
- There exists effective heuristics

Route of a truck

- Determine a feasible route for a truck in base of travelling salesmen problem
- Unserved customers
  - Constraints
    - Capacity of the truck
    - Distance → cost

Heuristic

- Feasible route
  - Send a truck from the depot to any unserved customer
  - Send the truck to the nearest unserved neighbor
    - Repeat until neighbor in 60 miles distance is exists and the capacity is not violated
- Repeat until exits unserved customer

Notes

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

- Quickly can be determined a good solution
- Heuristic allow to chose the first customer to visit
  - By repeating it with different rule for selecting the first customer in a route, the heuristic may be used to generate multiple feasible solutions
  - Best-know solution, incumbent solution (its cost is denoted by z_{INC})
- We have no information about the difference between z_{INC} and z_{MIN} (optimum)

Heuristic solution 1

- Choose the nearest unserved customer first

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Route	Customers	Cost
1	0	100

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

Heuristic solution 1

- Choose the nearest unserved customer first

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Route	Customers	Cost
1	0	100

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

Heuristic solution 1

- Choose the nearest unserved customer first

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Route	Customers	Cost
1	0-11	116

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

Heuristic solution 1

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

- Choose the nearest unserved customer first

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Route	Customers	Cost
1	0-11-10	140

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

Heuristic solution 1

- Choose the nearest unserved customer first

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Route	Customers	Cost
1	0-11-10-0	182

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

Heuristic solution 1

- Choose the nearest unserved customer first

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Route	Customers	Cost
1	0-11-10-0	182
2	0-5-6-9-0	174
3	0-12-7-0	169
4	0-2-1-0	197
5	0-4-3-0	214
6	0-8-0	170
7	0-13-0	200
	Total cost	1306

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

Heuristic solution 2

- Choose the furthest unserved customer first

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13
0	45	31	47	34	19	30	32	35	20	42	16	19	50
1		21	43	55	58	X	49	X	X	35	24	54	23
2			27	29	38	56	48	X	50	44	21	45	21
3				33	50	X	X	X	X	X	52	X	22
4					19	36	X	X	45	X	47	56	45
5						18	56	50	25	X	41	40	56
6							59	44	17	X	55	41	X
7								30	41	18	43	18	X
8									25	50	48	17	X
9										56	42	23	X
10											24	35	56
11												29	38
12													X

Route	Customers	Cost
8	0-13-3-5-0	241
9	0-1-2-6-0	252
10	0-10-7-0	192
6	0-8-0	170
11	0-4-9-0	199
12	0-12-0	138
13	0-11-0	132
	Total cost	1324

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13
Tons	3	5	5	4	2	2	6	8	3	3	7	4	3

5.1.1.3. GENERAL PURPOSE HEURISTIC

Introduction

- General-purpose heuristics starts from a (feasible of infeasible) solution and try to improve it
- Some methods are generalization of neighborhood search method
- Some general-purpose heuristics can theoretically extend their search until a global optimal solution is found

Genetic algorithm

- Randomized operations
  - Attempts to improve a pair of solutions (the parents) by a crossover
  - Mutation randomly modifies a given solution
    - Come out from local optimum
- The fittest solution survives
  - Fitness value

Example for crossover

- Take into account the two heuristic solutions
- Routes are ordered by descending value of their dollar cost per delivered ton
  - Fitness value
- Probability of accepting a route based on its fitness value x
  - $P(x) = Ke^{-\lambda x}$, $x \geq 15$
    - $K = 1.953$
    - $\lambda = 0.04463$
  - $P(15) = 1$, $P(20) = 0.8$

Routes

Order		Route	Cost	Tons	Fitness value	P(x)
1	Route 3	0-12-7-0	169	10	16.90	0.92
2	Route 1	0-11-10-0	182	10	18.20	0.87
3	Route 13	0-11-0	132	7	18.86	0.84
4	Route 6	0-8-0	170	8	21.25	0.76
5	Route 10	0-10-7-0	192	9	21.33	0.75
6	Route 5	0-4-3-0	214	9	23.78	0.68
7	Route 8	0-13-3-5-0	241	10	24.10	0.67
8	Route 4	0-2-1-0	197	8	24.63	0.65
9	Route 2	0-5-6-9-0	174	7	24.86	0.64
10	Route 9	0-1-2-6-0	252	10	25.20	0.63
11	Route 11	0-4-9-0	199	7	28.43	0.55
12	Route 12	0-12-0	138	4	34.50	0.42
13	Route 7	0-13-0	200	3	66.67	0.10

Crossover analysis

- A combined solution is created from the two feasible solution by selecting routes according to *P* and a random probability
- A route is selected for inclusion if
  - *P* is at least as great as the random probability
  - The route does not visit customer that has already covered by an earlier route selected

Routes

Order		Route	Probabi-lity	P(x)	Overlap	Include
1	Route 3	0-12-7-0	0.69	0.92		
2	Route 1	0-11-10-0	0.56	0.87		
3	Route 13	0-11-0	0.30	0.84		
4	Route 6	0-8-0	0.32	0.76		
5	Route 10	0-10-7-0	0.66	0.75		
6	Route 5	0-4-3-0	0.79	0.68		
7	Route 8	0-13-3-5-0	0.55	0.67		
8	Route 4	0-2-1-0	0.24	0.65		
9	Route 2	0-5-6-9-0	0.80	0.64		
10	Route 9	0-1-2-6-0	0.35	0.63		
11	Route 11	0-4-9-0	0.10	0.55		
12	Route12	0-12-0	0.98	0.42		
13	Route 7	0-13-0	0.92	0.10		

Routes

Order		Route	Probabi-lity	P(x)	Overlap	Include
1	Route 3	0-12-7-0	0.69	0.92		
2	Route 1	0-11-10-0	0.56	0.87		
3	Route 13	0-11-0	0.30	0.84		
4	Route 6	0-8-0	0.32	0.76		
5	Route 10	0-10-7-0	0.66	0.75		
6	Route 5	0-4-3-0	0.79	0.68	X	No
7	Route 8	0-13-3-5-0	0.55	0.67		
8	Route 4	0-2-1-0	0.24	0.65		
9	Route 2	0-5-6-9-0	0.80	0.64	X	No
10	Route 9	0-1-2-6-0	0.35	0.63		
11	Route 11	0-4-9-0	0.10	0.55		
12	Route12	0-12-0	0.98	0.42	X	No
13	Route 7	0-13-0	0.92	0.10	X	No

Routes

Order		Route	Probabi-lity	P(x)	Overlap	Include
1	Route 3	0-12-7-0	0.69	0.92	No	Yes
2	Route 1	0-11-10-0	0.56	0.87		
3	Route 13	0-11-0	0.30	0.84		
4	Route 6	0-8-0	0.32	0.76		
5	Route 10	0-10-7-0	0.66	0.75		
6	Route 5	0-4-3-0	0.79	0.68	X	No
7	Route 8	0-13-3-5-0	0.55	0.67		
8	Route 4	0-2-1-0	0.24	0.65		
9	Route 2	0-5-6-9-0	0.80	0.64	X	No
10	Route 9	0-1-2-6-0	0.35	0.63		
11	Route 11	0-4-9-0	0.10	0.55		
12	Route12	0-12-0	0.98	0.42	X	No
13	Route 7	0-13-0	0.92	0.10	X	No

Routes

Order		Route	Probabi-lity	P(x)	Overlap	Include
1	Route 3	0-12-7-0	0.69	0.92	No	Yes
2	Route 1	0-11-10-0	0.56	0.87	No	Yes
3	Route 13	0-11-0	0.30	0.84		
4	Route 6	0-8-0	0.32	0.76		
5	Route 10	0-10-7-0	0.66	0.75		
6	Route 5	0-4-3-0	0.79	0.68	X	No
7	Route 8	0-13-3-5-0	0.55	0.67		
8	Route 4	0-2-1-0	0.24	0.65		
9	Route 2	0-5-6-9-0	0.80	0.64	X	No
10	Route 9	0-1-2-6-0	0.35	0.63		
11	Route 11	0-4-9-0	0.10	0.55		
12	Route12	0-12-0	0.98	0.42	X	No
13	Route 7	0-13-0	0.92	0.10	X	No

Routes

Order		Route	Probabi-lity	P(x)	Overlap	Include
1	Route 3	0-12-7-0	0.69	0.92	No	Yes
2	Route 1	0-11-10-0	0.56	0.87	No	Yes
3	Route 13	0-11-0	0.30	0.84	Yes	No
4	Route 6	0-8-0	0.32	0.76		
5	Route 10	0-10-7-0	0.66	0.75		
6	Route 5	0-4-3-0	0.79	0.68	X	No
7	Route 8	0-13-3-5-0	0.55	0.67		
8	Route 4	0-2-1-0	0.24	0.65		
9	Route 2	0-5-6-9-0	0.80	0.64	X	No
10	Route 9	0-1-2-6-0	0.35	0.63		
11	Route 11	0-4-9-0	0.10	0.55		
12	Route12	0-12-0	0.98	0.42	X	No
13	Route 7	0-13-0	0.92	0.10	X	No

Routes

Order		Route	Probabi-lity	P(x)	Overlap	Include
1	Route 3	🟩 0-12-7-0	0.69	0.92	No	Yes
2	Route 1	🟩 0-11-10-0	0.56	0.87	No	Yes
3	Route 13	0-11-0	0.30	0.84	Yes	No
4	Route 6	🟩 0-8-0	0.32	0.76	No	Yes
5	Route 10	0-10-7-0	0.66	0.75	Yes	No
6	Route 5	0-4-3-0	0.79	0.68	X	No
7	Route 8	🟩 0-13-3-5-0	0.55	0.67	No	Yes
8	Route 4	🟩 0-2-1-0	0.24	0.65	No	Yes
9	Route 2	0-5-6-9-0	0.80	0.64	X	No
10	Route 9	0-1-2-6-0	0.35	0.63	Yes	No
11	Route 11	🟩 0-4-9-0	0.10	0.55	No	Yes
12	Route12	0-12-0	0.98	0.42	X	No
13	Route 7	0-13-0	0.92	0.10	X	No

Covering

- The resulted routes does not cover all customers
  - The orphan customer is customer 6
- An attempt to complete the solution is made
  - The orphan could be assigned to routes 4, 6 and 11
  - Most efficient assignment is to assign to route 11

New solution

- The values of the two previous solutions were 1306 and 1324
- The value of the new solution is 1166

Route	Customers	Cost
3	0-12-7-0	169
1	0-11-10-0	182
6	0-8-0	170
8	0-13-3-5-0	241
4	0-2-1-0	197
11*	0-4-6-9-0	207
	Total cost	1166

Notes

- One numerical example does not validate a theory
  - Because of random values it can lead to worse solution than parents’
- The computation is very simple → the crossover analysis could be applied thousands of times a minute

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

5.2. OVERVIEW OF THE UNIFIED OPTIMIZATION METHODOLOGY

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- Decomposition
  - Breaks a large-scale, monolithic MILP model down into several smaller submodels
  - Solves these submodels
  - Build the global plant from these solutions
- Heuristics

Motivation

- Quickly produces good solution
  - Heuristics
- Global decisions, constraints
  - Heuristics cannot handle

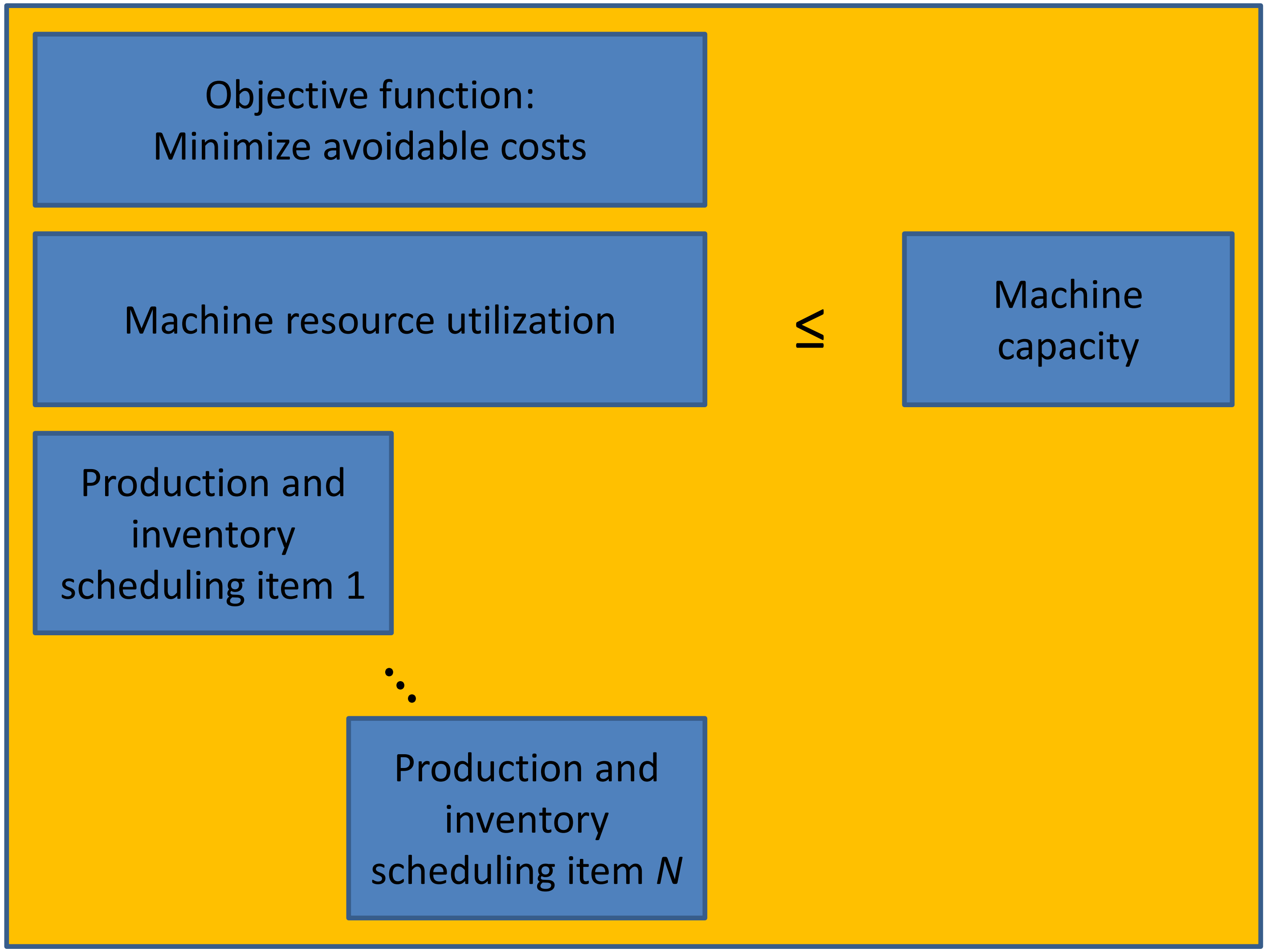
5.2.1. PRODUCTION SCHEDULING EXAMPLE OF DECOMPOSITION

Example

- Manufacturing *N* products
- Shared resources
- Multiple-week planning horizon
- An individual item is produced intermittently of a variable number (production lot size)

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

Discrete part of the MILP model


Objective function:
Minimize avoidable costs
Machine resource utilization
≤
Machine capacity
Production and inventory scheduling item 1
Production and inventory scheduling item *N*

Resources

- If an item is produced in a particular week, it uses machine resources
  - Setup time
  - Production rate
- Resources use across items produced in a week cannot exceed the available machine capacity

The size of the MILP model

- 13-week planning horizon
- 500 products
- 6500 binary setup variables
- 20,000 constraints

- Rolling-horizon is required

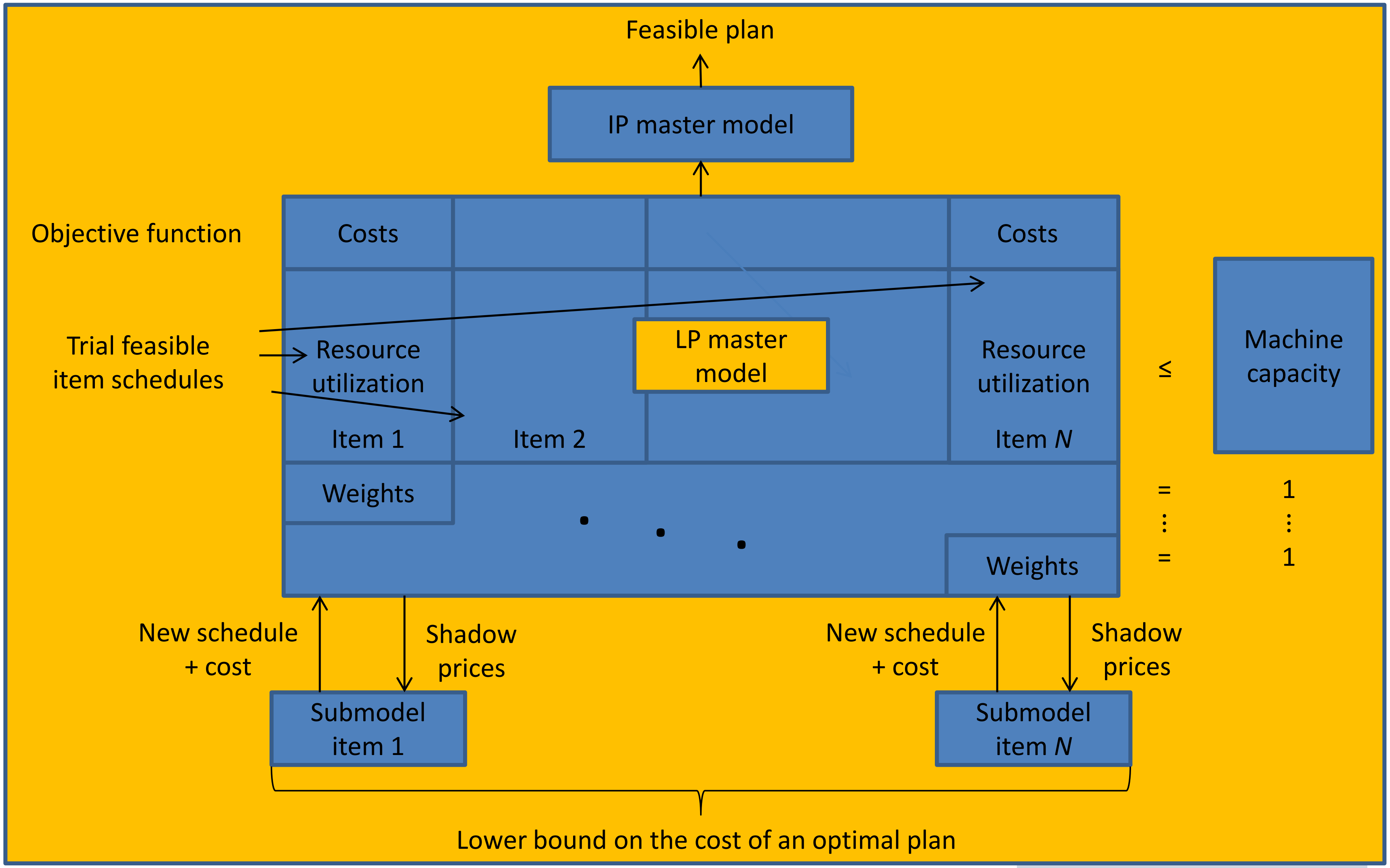
Decomposition idea

- Production/inventory scheduling submodels for individual items could be easily optimized
- Common resources connect the submodels

„Price out"

- Add cost for machine resources consumed
  - Converts constraints into objective function term
  - What the price to charge for a machine?

Decomposition



Feasible plan

- Feasible item schedule
  - Consists of weekly decisions regarding production and inventory such that demand for the item is met
  - The resource constraints is valid only for the single item
- Feasible plan
  - Collection of feasible item schedules
  - The sum of resource requirements does not exceed machine capacities

Trial feasible item schedules

- Might be constructed
  - Manually
  - Based on heuristics
  - Some combination of the two
- There are more schedule for an item
  - Decision variables for all item schedule (weights)
    - Fraction of that schedule that the model wishes to select
    - Sum of the weights over the schedules must equal 1 for each item

Master LP model

- May be solved by an LP solver
- The solution determines for an item
  - Pure plan
    - One feasible item schedule has been selected
  - Mixed plan
    - More than one schedule has been selected
    - Cannot be implemented
- The purpose is to
  - Select optimal combination of schedules
  - Produce optimal shadow prices on machine constraints and weight constraints

IP master model

- The schedule weights must take on values either of 0 or 1
  - Only pure plans for each item
- The purpose is to determine the best feasible plan from the know feasible item schedules
  - Not necessary the optimal

Shadow prices

- Assuming there is one machine capacity constraint for each week *t*
  - Let π_t denote the shadow price (obtained by optimizing the LP master model)
- Let θ_i denote the shadow price on the equation constraining the weights for item *i* to equal 1

Submodels

- Without shadow prices, the objective function of each submodel is the minimization of costs associated with the item
- The shadow prices are used to add a resource cost
  - π_t * machine-hours for machine-hours consumed in week *t* producing item *i*
  - We call it resource adjusted cost
- It can be demonstrated that θ_i equals the resource adjusted cost of the one or more feasible item schedules for item *i* in the LP master model that have minimal resource adjusted cost

Solving submodels

- Submodels can be solved easily my MILP solver

Test for a new feasible item schedule

- For each item *i*, solve an optimization model to compute the minimal resource adjusted cost
  - Δ_i = minimal resource adjusted cost – θ_i
- If $\Delta_i < 0$, than we add the new feasible schedule to the LP master model
  - More cost effective than any of the current ones
- If $\Delta_i = 0$ for each item we do not add the schedule to the LP master model

Lower bound

- Adding a new feasible item schedule to the master LP model yields a lower bound ($LB_\sqcap$) on the cost of optimal solution of the original MILP model
  - $LB_\sqcap$ = minimal objective function cost of master LP model $+ \Delta_1 + \Delta_2 + \cdots + \Delta_N$

IP master model

- When to optimize the IP master model
  - Ideally, once
    - It may not occur very often
  - Experimentation is needed for each implementation to determine the frequency

Heuristics

- Heuristics are needed to initialize the LP master model
  - Feasible schedules for each item are needed
- Heuristics may be developed to generate feasible schedules for the LP master model in addition

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

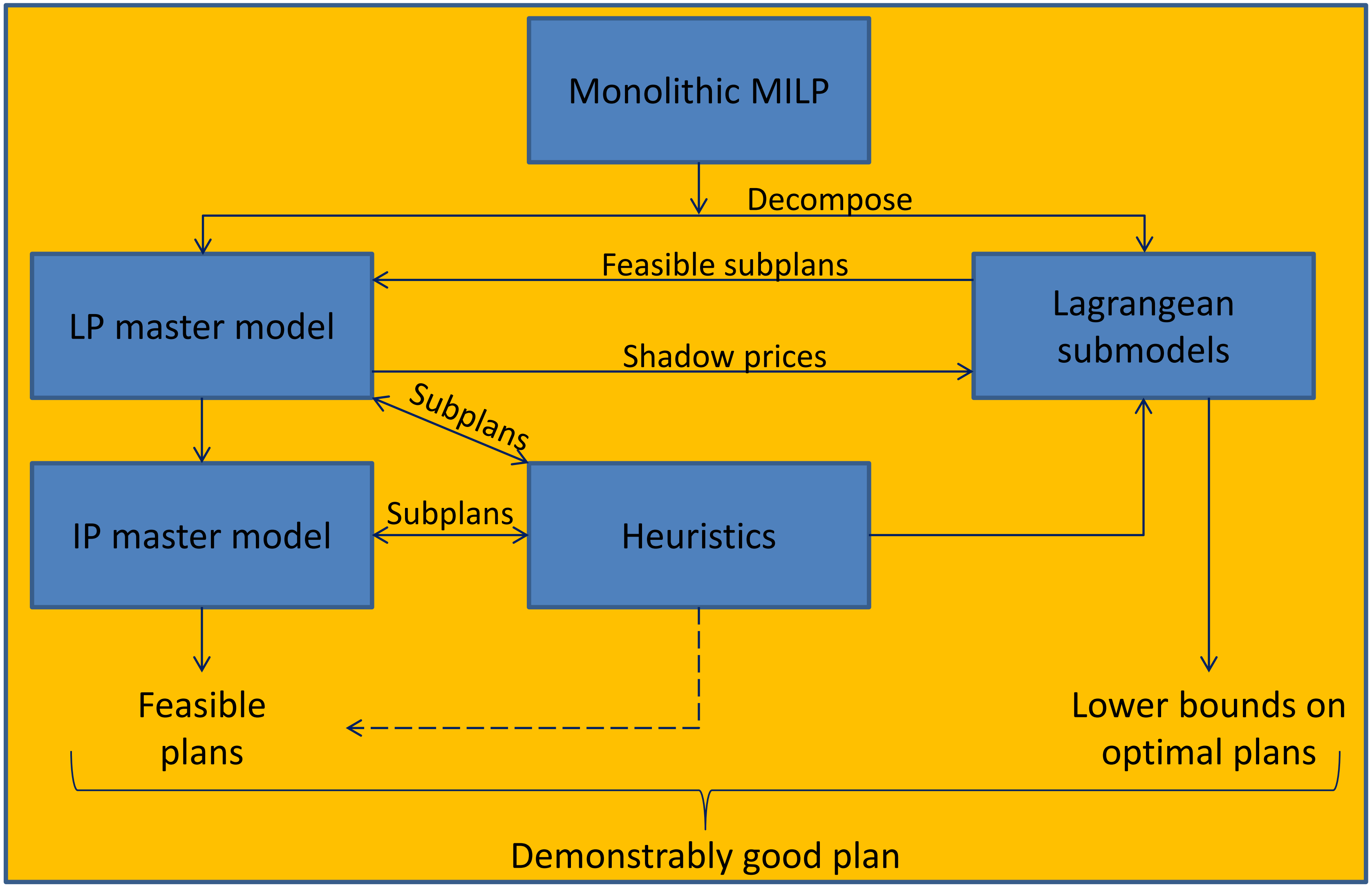
5.2.2. UNIFIED OPTIMIZATION METHODOLOGY

MAGYARORSZÁG MEGÚJUL
ÚJ SZÉCHENYI TERV

Introduction

- Generalize the discussion of the pervious section
- We begin from a monolithic MILP model
  - Decomposing into an LP master model
  - Generating Lagrangean submodels

Methodology



Lagrangean submodels

- The spirit of the decomposition is to place resource capacity and other cross-cutting constraints in the LP master model
  - This allows to separately and independently optimize the Lagrangean submodels
- We refer to the submodels as Lagrangean because the shadow prices used are Lagrangean multipliers

Heuristics

- Initialization of LP master model
- Generation of subplans with crossover
- Using subplans of similar problems

5.3. METHODOLOGY APPLIED TO VEHICLE ROUTING

Example 5.1

- Chemtech’s vehicle routing problem
- Describing IP master model
- LP master model is a relaxation of IP master model
- Solving LP master model and Lagrangean submodels

5.3.1. STATEMENT OF THE OPTIMIZATION MODELS

Parameters

- Let M be the number of customers with orders to delivered, which is indexed by i
- Suppose, we have somehow generated N feasible routes, which we index by j
- Let c_j denote the cost of route j
- Let $a_{ij} = 1$, if route j visits customer i; 0 otherwise

Decision variable

- x_j : equals 1 if route j is selected; 0 otherwise

Local delivery LP master model

- Relaxation of IP master model
  - The binary decision variables become continuous variables in 0-1 interval
  - $0 \leq x_j \leq 1$

IP and LP master model

$z_{\mathrm{LP}N}, z_{\mathrm{IP}N} = \min c_1x_1 + c_2x_2 + \cdots + c_Nx_N$

s.t.

$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N = 1$

$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N = 1$

…

$a_{M1}x_1 + a_{M2}x_2 + \cdots + a_{MN}x_N = 1$

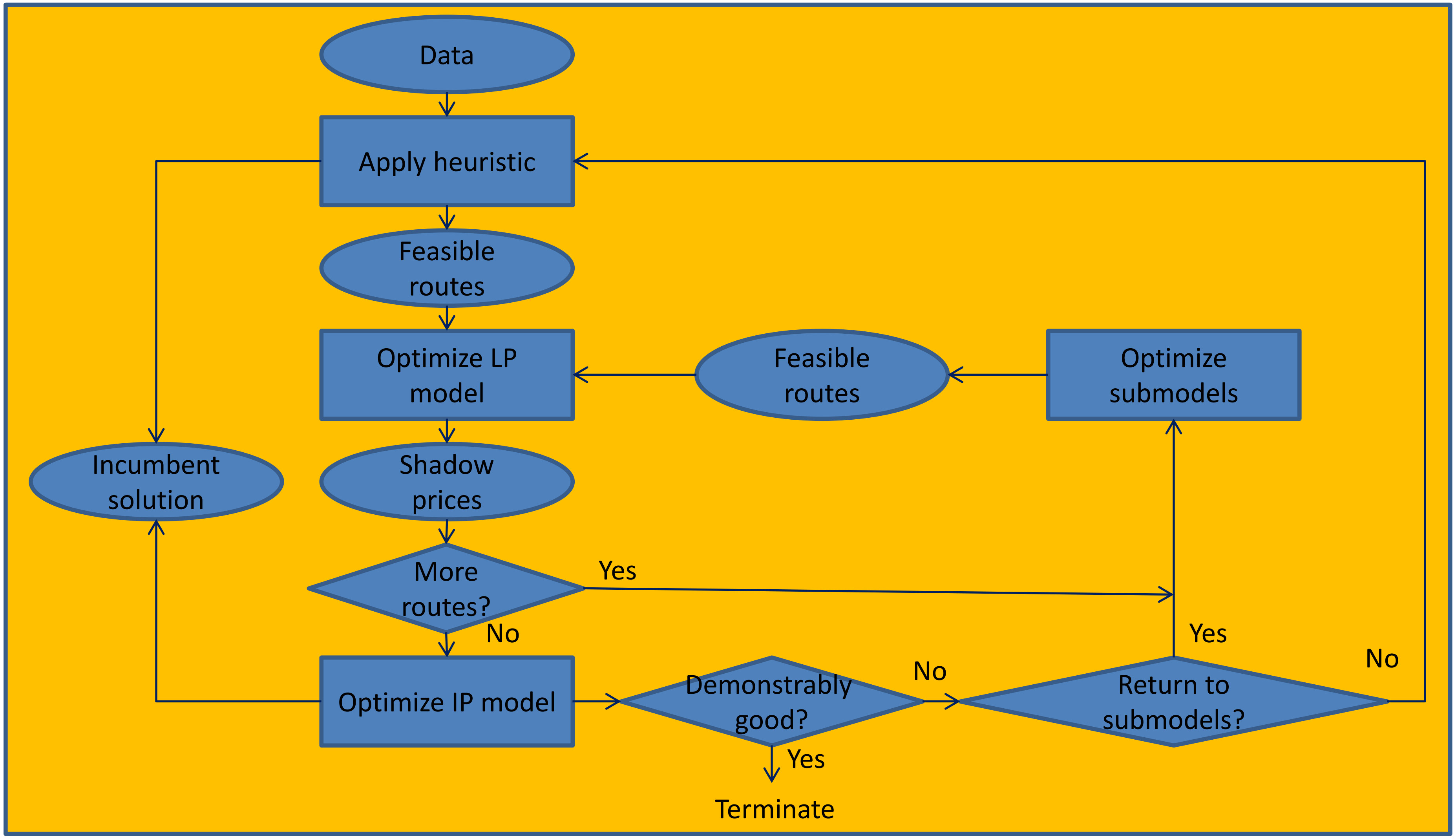
$0 \leq x_j \leq 1$ (linear programming)

$x_j \in \{0, 1\}$ (integer programming)

IP and LP (master) models

- Models are different in form from those displayed in previous section
  - Do not involve constraints summing weights to 1
- We are able to construct these models because we are not seeking routes for individual trucks, but rather a number of routes for identical trucks

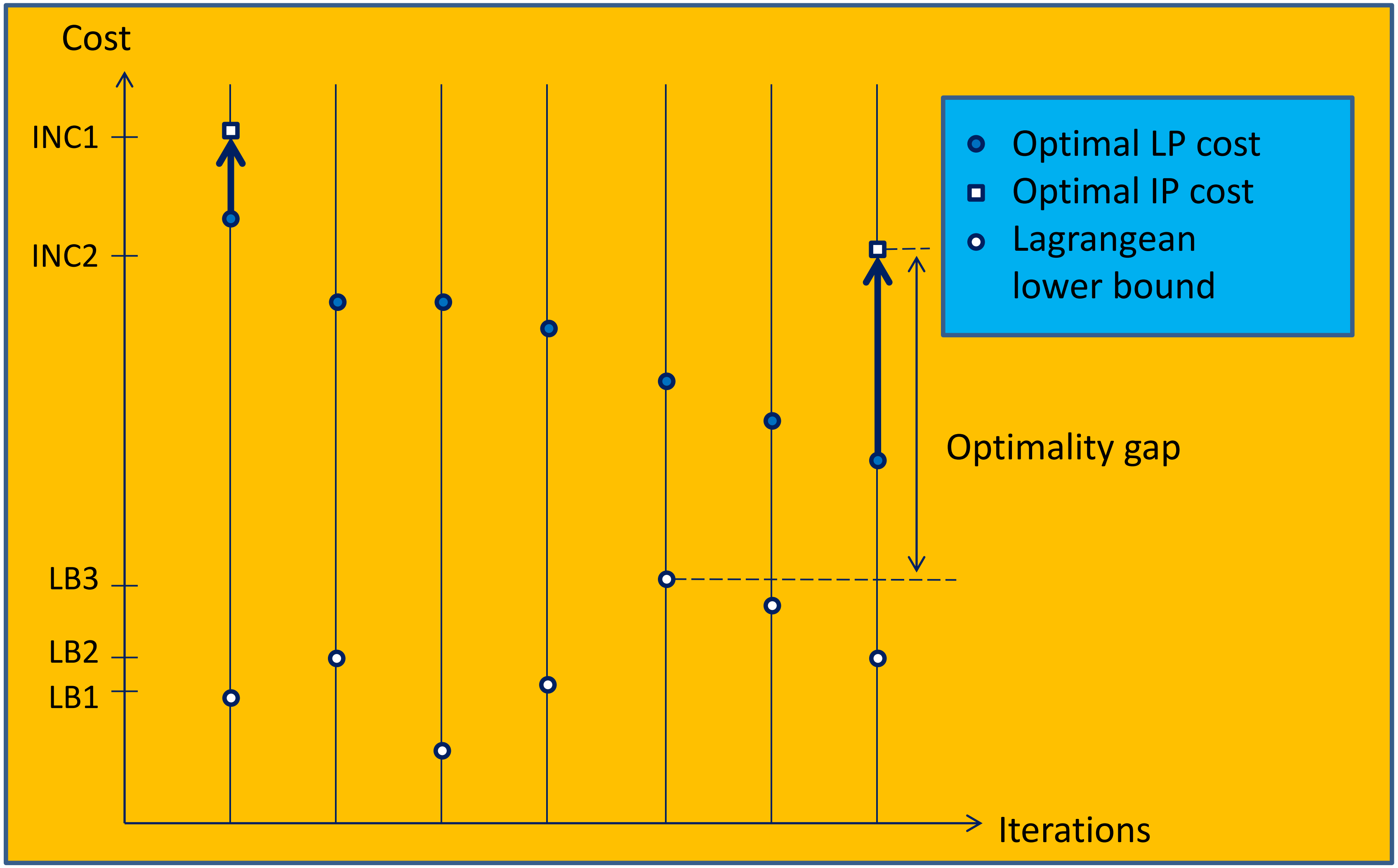
Unified methodology for local delivery problem



Lagrangean submodels

- Submodel employs Lagrange multiplier
  - π_i associated with each customer *i* as a reward for visiting the customer
- Optimizing with dynamic programming
  - Computing a feasible route
- Objective maximize net reward (gross reward – cost of the route)
  - If it is positive, we add the route to the LP model

Optimization process



5.3.2. GENERALIZATIONS

Possible generalizations

- Delivery windows with customers' orders
- Vehicles of different size for various routes
- More than one depot
- Routing may be constrained to obey laws relating to breaks
- Travel time and delivery time
  – Latter depends on the size of the order
- Limitation on the number of trucks

Longer-haul delivery problems

- Trucks, trains, barges, ships
- Cross-docking in depots

5.4. UNIFIED OPTIMIZATION METHODOLOGY APPLIED TO PRODUCTION SCHEDULING

Example 5.2

- Goodstone Tire Company
  - 8-weeks planning horizon
- 6 different types of tires
- Single machine

5.4.1. NUMERICAL DATA

Operating the machine

- 2 shifts a day, 6 days a week (from Monday to Saturday)
  - 96 hours/week
- Maintenance on the 7th day (Sunday)
- A third shift may be run 4 days a week (Monday through Thursday) at a high ($100) hourly cost
  - Up to 32 hours a week

Demands

Week\tire	1	2	3	4	5	6
1	254	359	303	412	501	622
2	306	388	283	428	484	702
3	339	416	276	465	437	645
4	328	447	258	488	463	717
5	340	470	206	358	497	677
6	315	505	208	334	552	745
7	284	533	222	516	589	728
8	327	562	236	477	640	576

Aim

- Minimize costs
  - Setup cost
  - Inventory-holding cost
  - Overtime cost
- Direct manufacturing costs are constant over planning horizon

Production data

Tire	1	2	3	4	5	6
Setup cost ($)	450	500	500	525	600	475
Holding cost ($/item/week)	0.33	0.38	0.35	0.41	0.48	0.34
Setup hours	2	2.5	3	4	3.6	2.8
Machine hours/tire	0.029	0.03	0.033	0.037	0.036	0.026
Initial inventory	301	578	446	667	355	678
Terminal inventory	200	300	150	250	350	450

Feasible plan

- Feasible item schedule is a plan for an item allowing demand for it to meet
- A feasible plan is a collection of feasible item schedules, one for each item, where machine capacity, including overtime, is not violated
- Sometimes, feasible plan might not exists

MILP model

- Industrial size problems are not solvable
  - Scheduling of 1000 products in 13 week planning time → 13,000 binary variable

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése
TÁMOP-4.1.2.A/1-11/1-2011-0088

5.4.2. METHODOLOGY SPECIALIZED TO GOODSTONE'S PROBLEM

Dynamic lot-size theorem

- Consider a production planning problem for a single item over a planning horizon of *T* weeks
- Minimize the sum of setup and inventory-holding costs
- Assuming initial inventory is zero
  - Produce a lot size equal to the sum of demand over the first *k* weeks
    - *k* is unknown ($1 \leq k \leq T$)
  - If $k \leq T - 1$, produce in week $k + 1$ a lot size equal to the sum of demand over the next *q* weeks ($1 \leq q \leq T - k - 1$)
  - Repeat until demand for entire planning horizon has been covered
- Production occurs only in weeks when inventory falls to zero

Remark

- Can be optimized by dynamic programming (list processing) computation
- Possible decisions for each week t
  - $0,\ d_t,\ d_t + d_{t+1},\ \ldots\ ,\ d_t + d_{t+1} + \cdots + d_T$
    - Where d_s equals demand in week s

Production-scheduling heuristic

- Suppose that t is the last week before inventory runs out
  - Select an integer $k_1 \leq T - t$ such that net demand over weeks t through $t + k_1$ does not exceed the allowed bound on production
    - Set up to produce a quantity of the item covering net demands in this time interval
- The inventory will runs out at week $t + k_1$
  - Select an integer $k_2 \leq T - t - k_1 - 1$ …
- Requirements can exists for ending inventory

Notations

- I : number of items to be scheduled
- K : number of trial feasible schedules
- C_{ik} : cost ($) of feasible item schedule k for item i
- r_{ikt} : machine capacity (hours) used in week t by feasible item schedule k for item i
- R_t : machine capacity (hours) in week t without overtime
- UB_t : upper bound (hours) on overtime available in week t
- P : cost ($/hour) for overtime

Decision variables

- w_{ik} : weight assigned to feasible item schedule *k* for item *i*
- O_t : overtime (hours) in week *t*

IP and LP master model

$z_{\mathrm{LPK}}, z_{\mathrm{IPK}} = \min C_{11}w_{11} + \cdots + C_{1K}w_{1K} + \cdots + C_{N1}w_{N1} + \cdots + C_{NK}w_{NK} + P(O_1 + \cdots + O_T)$

s.t.

$r_{111}w_{11} + \cdots + r_{1K1}w_{1K} + \cdots + r_{N11}w_{N1} + \cdots + r_{NK1}w_{NK} - O_1 \le R_1$

$r_{112}w_{11} + \cdots + r_{1K2}w_{1K} + \cdots + r_{N12}w_{N1} + \cdots + r_{NK2}w_{NK} - O_2 \le R_2$

…

$r_{11T}w_{11} + \cdots + r_{1KT}w_{1K} + \cdots + r_{N1T}w_{N1} + \cdots + r_{NKT}w_{NK} - O_T \le R_T$

$w_{11} + \cdots + w_{1K} = 1$

…

$w_{N1} + \cdots + w_{NK} = 1$

$0 \le O_t \le \mathrm{UB}_t$ ($t = 1, 2, \ldots, T$)

$0 \le w_{ik} \le 1$ (linear programming)

$w_{ik} \in \{0, 1\}$ (integer programming)

Lagrangean submodels

- Determines a feasible production schedule
- Objective
  - Minimize total resource adjusted costs = total avoidable costs + resource costs
    - Resource cost = shadow price * resource usage of the schedule
- May be solved by
  - MILP solver
  - Dynamic programming algorithm

Feasible plans

- If the cost of an optimal schedule for item *i* is less than the shadow price on the weight constraint for item *i*, the schedule is added to the LP master model
  - Reoptimization of LP master model
- If there is no such a schedule, solving IP master model or generating new heuristic product schedule

5.4.3. PRODUCTION-SCHEDULING SOLUTION

Heuristic

- Three trial feasible item schedules for each of the six tires

An initial feasible item schedule

Week	Demand	Setup	Production	Ending inventory	Cost ($)	Resource utilization
0				301		
1	254	0	0	47	15.51	0
2	306	1	926	667	670.11	28.854
3	339	0	0	328	108.24	0
4	328	0	0	0	0	0
5	340	1	939	599	647.67	29.231
6	315	0	0	284	93.72	0
7	284	0	0	0	0	0
8	327	1	527	200	516	17.283
	2392				2051.25	

First solution

- Solution of LP master model 17,114.99$
  - No overtime
  - Mixed plan for tire 2 and 4
- Solution of IP master model 17,599.39$
  - Overtime in week 1 and 6

Lagrangean submodels

Tire	Shadow price	Cost	Δ_j (difference)
j = 1	2234.73	2037.39	-197.34
j = 2	3092.35	2745.60	-346.75
j = 3	2197.55	2051.95	-145.60
j = 4	3172.98	2691.43	-481.55
j = 5	3945.24	3863.52	-81.72
j = 6	3339.72	2870.36	-469.36
Min. master LP cost: 17,114.99 Lower bound: 15,392.67			

Iterations

Iteration	LP master cost	Sum of Δ_j	Number of improving schedules	Lower bound	Greatest lower bound	Incumbent cost	Optimality gap
1	17,115	-1661	6	15,393	15,393	17,599	0.1253
2	16,740	-1479	6	15,261	15,393	17,599	0.1219
3	16,669	-729	6	15,940	15,940	17,599	0.0943
4	16,604	-303	4	16,301	16,301	16,903	0.0356
5	16,514	-236	4	16,278	16,301	16,889	0.0348

5.4.4. GENERALIZATIONS

Possible generalizations

- More complex manufacturing model
  - More resources
  - Multiple stages
  - …
- Incorporate decision options
  - Which orders to delay and for how long
  - …

Magyarország a Kelet-Európai logisztika központja - Innovatív logisztikai képzés e-learning alapú fejlesztése

TÁMOP-4.1.2.A/1-11/1-2011-0088

PANNON *Egyetem*

Nemzeti Fejlesztési Ügynökség
www.ujszechenyiterv.gov.hu
06 40 638 638

MAGYARORSZÁG MEGÚJUL

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.