ISyE 3103 and ISyE 6203

Transportation and Supply Chain Systems

Notes
Introduction to Logistics

Logistics definitions:

Council of Logistics Management (CLM)
The process of planning, implementing, and controlling the efficient, effective flow and storage of goods, services, and related information from the point of origin to the point of consumption for the purpose of conforming to customer requirements.

Contemporary Logistics (Johnson, et. al 1999)
The entire process of materials and products moving into, through, and out of a firm.

• Inbound logistics: materials from suppliers to a firm
• Materials management: intrafirm control of raw materials and finished goods (not WIP)
• Physical distribution: finished goods from firm to customers

Logistics Systems Analysis (Daganzo 1999)
The science that studies how to convey items from production to consumption in a timely, cost-effective manner:

• the “items” transported may be freight or passengers

What is a supply chain?

First, not a chain but instead is a network. In class, we simulated a very simple supply chain that consisted of a series of organizations that work together to produce and supply Coke to the end consumer. The chain included a raw material supplier, a syrup producer, a canning plant, a wholesaler, and a retailer.

Most firms are involved in complicated supply networks, consisting of many organizations with often competing objectives.

Organizations Involved

• Within your firm
  – Design and production departments
  – Logistics/traffic management/transportation departments
  – Sales and marketing departments
  – Finance department
  – Note that for large firms, these groups may be spread-out in different offices, plants, warehouses, cross-docks, etc.

• Other firms
Distribution centers (warehouses, cross-docks, wholesalers)

Figure 1: A schematic of the supply chain network.

- Suppliers
- Suppliers’ suppliers (and so on!)
- Customers
- Customers’ customers (and so on!)
- Logistics service providers (carriers, 3PLs, forwarders, brokers, etc.)

**Logistics vs. Supply chain management**

Managing the network of inter-related components necessary to provide consumers with a product or set of products. Includes planning production levels, sourcing materials, controlling inventories, as well as transportation logistics. Requires managing relationships between supply chain partners, and coordinating information flows.

Modern view: supply chain management and logistics are equivalent!

**Key issues**

**Tactical:** medium-term, updated once a quarter or once a year. Ex. Production and purchasing decisions, equipment leasing, vendor selection, transportation contracts, inventory policies, storage allocation.

**Operational:** day-to-day decisions. Vehicle routing and scheduling, order filling, order placing.

- Transportation management: *modes, vendors, equipment, routes, schedules*

**Network configuration**

- Facilities location and capacity
- Production allocation
- Selection of suppliers
- Material flow pattern

**Network configuration**

- Facilities location and capacity
- Production allocation
- Selection of suppliers
- Material flow pattern

**Inventory management**

- Demand forecasting: customer finished goods, through network
- Locations and ownership
- Inventory levels
- Replenishment strategies, stocking policies
- Push vs. pull
- Materials handling systems
- Order management and processing
**Holding cost concept:** Holding a unit of inventory requires costs. Why? (1) Rent cost (for space in facilities) and (2) Waiting cost (inventory represents an investment. Value of inventory * interest rate * time).

**Product and production design**

- Cost-effective product design for logistics (storage, transportation, packaging, handling costs)
- Mass-customization vs. fixed product line
- Delayed differentiation

**Relationship management**

- Identify customer needs, set customer service levels
- Core competencies vs. outsourcing
- Ownership of components
- Contractual control of components
- Balancing competing objectives
- Information-sharing and communications

**Communication and information flow**

- Intrafirm and external coordination
- Real-time access to data
- Information to share (inventories, orders)
- EDI standards for info exchange
The terminology of transportation logistics

*Shipper/consignor*
An individual or firm that sends freight. A freight *originator*.

*Consignee*
An individual or firm to whom freight is shipped. A freight *receiver*.

*Carrier*
A firm that provides transportation services, typically owning and operating transportation equipment. Examples include: trucking company, railroad, airline, steamship line, parcel/express company.

*Freight bill-of-lading (freight bill)*
A document providing a binding contract between a shipper and a carrier for the transportation of freight, specifying the obligations of both parties. Serves as a receipt of freight by the carrier for the shipper. Usually designates the consignee, and the FOB point.

*FOB (free-on-board) point*
Point at which ownership of freight changes hands from shipper to consignee. FOB-origin indicates that consignee owns the goods in transit; FOB-destination indicates that shipper owns goods in transit. Owner of goods in transit is liable for loss and damage to freight, and thus should provide insurance.

*Freight FOB terms-of-sale*

Examples:
1. FOB-origin, freight collect: consignee pays freight charges and owns goods in transit.
2. FOB-destination, freight prepaid: shipper pays freight charges and owns goods in transit.
3. FOB-destination, freight prepaid and charged back: shipper owns goods in transit, pays for freight but bills consignee for the charges.

*Loss and damage*
Loss or damage of freight shipments while in transit or in a carrier-operated warehouse. Terms for the handling of claims are usually stipulated in the freight bill. Shippers/consignees usually take out insurance against L&D with premiums a function of the value of goods shipped, and the likelihood of L&D.
Private carrier
Owned and operated by a shipper. Usually refers to private trucking fleets. Components include: vehicle fleet, drivers, maintenance equipment. Often more expensive than contracting out, but not always. Can serve special needs: fast, high-on-time-reliability delivery; special equipment; special handling; availability. Examples: Safeway (grocery), Office Depot (office products).

Common carrier
A for-hire carrier providing transportation services to the general public. Obligations: to serve, to deliver, to charge reasonable rates, to avoid discrimination. Previously regulated in the United States; most are now deregulated. Examples: Parcel/express carriers (United Parcel Service (UPS), FedEx), LTL trucking (Yellow, Consolidated Freightways, Roadway), TL trucking (Hunt, Schneider), Rail carrier (Norfolk Southern), Air carriers (Delta, Flying Tigers), Ocean carrier (SeaLand, American President Lines (APL)).

Freight forwarder
An agency that receives freight from a shipper and then arranges for transportation with one or more carriers for transport to the consignee. Often used for international shipping. Will usually consolidate freight from many shippers to obtain low, large-volume transportation rates from carriers (through a contract). Often owns some pickup and delivery equipment; uses to transport freight to/from consolidation facilities. Also provide other shipping services: packaging, temporary freight storage, customs clearing.

Transportation broker
An agency that obtains negotiated large-volume transportation rates from carriers, and resells this capacity to shippers. Unlike freight forwarders, will not handle freight and owns no pickup/delivery equipment or storage facilities.

NVOCC
Nonvessel-operating common carrier. Owns no vessels (ships), but provides ocean shipping freight-forwarding services. Provides consolidated, negotiated-rate services for ocean and inland water carriers. Often will affiliate with freight forwarders to provide pickup/delivery, other services.

Shippers’ association
Not-for-profit association of shippers using collective bargaining and freight consolidation to obtain lower, high-volume transportation rates; similar to freight forwarding w/o profit motive. Avoids premium charges paid to forwarders. Only non-competitive shippers may associate, due to monopoly restrictions.

Integrators
Companies that provide door-to-door domestic and international air freight service. Own and operate aircraft, as well as ground delivery fleets of trucks. In contrast, freight-hauling airlines (e.g., Delta, Lufthansa) typically do not provide door-to-door service. Example: UPS, FedEx, BAX Global, Emery Worldwide.
3PL
A third-party, or contract, logistics company. A firm to which logistics services are outsourced. Typically handles many of the following tasks: purchasing, inventory management/warehousing, transportation management, order management. Example: Schneider Logistics, Ryder Logistics, UPS Logistics.

Interline shipment
Shipment moving from origin to destination via two or more carriers. Occurs frequently in rail transportation: for example, each rail container moving from Atlanta to Los Angeles is moved interline (why?), using for example CSX and Union Pacific with an interline junction in New Orleans.

Door-to-door
Transportation service arrangement in which freight is moved from origin (shipper) through to ultimate destination (consignee) for a given rate. Trucking companies typically offer door-to-door service. Railroads do not, unless the shipper and consignee both have rail sidings. Brokers, forwarders, NVOCCs etc. often package together door-to-door service through contracts with multiple carriers.

Consolidation
Bringing together many small shipments, often from different shippers, into large shipment quantities, in order to take advantage of economies of scale in transportation costs. In-vehicle consolidation is when a vehicle makes pickups from many customers and consolidates freight inside the vehicle. Out-of-vehicle consolidation occurs at a terminal facility; shipments to a single customer/region are consolidated before shipment.

Terminal
Transportation facility with one or more of the following roles:

1. System access: terminals are points at which freight enters and leaves the transportation system.
2. Freight consolidation/distribution
3. Mode transfer: freight may change from one mode to another, for example, rail to truck.
4. Vehicle transfer: within a single mode, freight may transfer from one vehicle to another.
5. Storage and warehousing
6. Fleet maintenance

Hub-and-spoke
A transportation system design in which large hub terminals are used for freight consolidation. Medium-volume services serve the spoke-to-hub collection and hub-to-spoke
distribution tasks. Large-volume services are operated in the hub-to-hub markets. In most systems, all outbound/inbound freight for a spoke uses the same hub, and thus larger shipment sizes are realized. Many transportation systems oriented in this way. Examples: Delta airlines, FedEx, LTL, and now ocean shipping. Not TL, however.

Cross-dock
Transportation terminal in which received items transferred directly from inbound to the outbound shipping dock, with storage only occurring temporarily during unloading and loading. No long-term storage is provided. Usually used only for vehicle transfers. Often owned and operated by large shippers. Examples: Home Depot, food service companies, hub passenger airports.

TL/FTL (truckload, full truckload)
A trucking industry term; a truckload shipment is when the shipper contracts an entire truck for direct point-to-point service. Truckload shipments are priced per mile within designated lanes, regardless of the size of the shipment provided it fits (weight, cube) within the vehicle. Less expensive per unit weight shipped than LTL. A truckload carrier is a trucking company specializing in point-to-point truckload shipments. Examples include: J.B. Hunt, Schneider.

LTL (less-than-truckload)
A trucking industry term; a less-than-truckload (LTL) shipment is when a shipper contracts for the transportation of freight that will not require an entire truck. LTL shipments are priced according to the weight of the freight, its commodity class (which generally determines its cube/weight ratio), and mileage within designated lanes. An LTL carrier specializes in LTL shipments, and therefore typically operates a complex hub-and-spoke network with consolidation/deconsolidation points; LTL carriers carry multiple shipments for different customers in single trucks. Examples include: Yellow Freight, Consolidated Freightways, Roadway Express.

Freight size
Freight is most often measured by its weight, and transportation vehicles of varying sizes typically have weight capacities that cannot be exceeded due to engineering or regulatory reasons. Freight may also be measured by cube, which generally refers to the volume of the freight. A vehicle is said to cube-out if it does not exceed its weight capacity, but its volume is completely full.

FCL (full container-load)
An ocean-shipping and intermodal industry term; a full container-load shipment is when a shipper contracts for the transportation of an entire container. The vast majority of intermodal and ocean freight is contracted in this manner. Historically, FCL also stands for full carload which is the primary business of all modern railroads, and is the railroad equivalent of TL trucking.

LCL (less-than-container-load)
An ocean-shipping and intermodal industry term; LTL equivalent in container shipping.
Container freight stations at ports serve as consolidation and deconsolidation terminals. Historically, LCL also stands for less-than-carload. Before the prominence of interstate trucking, railroads offered less-than-carload (LCL) service but this business has largely disappeared.

**Dead-head**
A portion of a transportation trip in which no freight is conveyed; an empty move. Transportation equipment is often dead-headed because of imbalances in supply and demand. For example, many more containers are shipped from Asia to North America than in reverse; empty containers are therefore dead-headed back to Asia.

**Backhaul**
A freight movement in a direction (or lane) of secondary importance or light demand. Backhauls are preferable to deadheads by transportation companies, since revenue is generated. In order to entice shippers to move goods in backhaul markets, carriers may offer lower rates.

**Intermodal**
Transportation that uses a specialized *container* that can be transferred from the vehicle of one mode to the vehicle of another; a single freight bill is used for the shipment. Example: Ocean shipping containers which can be hauled by trucks on chassis, railcars, ocean vessels, and barges. Also: UPS line-haul vans (these vans can be stacked onto railcars for long distance moves).

**Container, chassis, and vans (trailers)**
Standard trucking companies use vans (or trailers) to move standard dry goods. These trailers consist of a storage box that is permanently attached to a set of wheels (the set of wheels is often known as a truck, confused yet?). Intermodal ocean containers are moved on the road by attaching them to a separate piece of equipment, a chassis, which is essentially a set of wheels on a lightweight frame.

**Container**
A single, rigid, sealed, reusable metal box in which merchandise is shipped by vessel, truck, or rail. Container types include standard, high cube, hardtop, open top, flat, platform, ventilated, insulated, refrigerated, or bulk. Usually 8 ft x 8 ft in width and height, 20 to 55 ft long. Specialized containers also exist for air transportation modes, but are much smaller and cannot be directly transferred to truck or rail.

**Reefer**
A refrigerated container. For long storage in transit (or in ports) must be plugged into a ship’s power system (or port’s). Temporary power units can be attached that last for 18-36 hours.

**COFC**
Container-on-flatcar. A term used in intermodal transportation in which containers are stacked onto rail flatcars for rail transportation. No truck chassis is used, and double-
stack cars are possible, thus more containers can be carried by a shorter, lighter train.

**TOFC/piggyback**

Trailer-on-flatcar. A term used in intermodal transportation in which truck trailers or container/chassis combinations are placed directly onto rail flatcars for the rail portion of the trip. TOFC trains are generally heavier and longer per unit ton shipped, but have the advantage that unloaded trailers can be moved out of the intermodal terminal without worrying about finding a chassis; thus, the equipment management issues are simpler.

**Drayage**

Local trucking, typically describing truck movement of containers and trailers to and from rail intermodal yards and to and from port facilities.

**Pickup and delivery (cartage)**

Local hauling of freight. Often the trucking service used for transferring freight from the shipper to a terminal, or from a terminal to a consignee.

**Switching**

Switching is a railroad term denoting the local movement of freight rail cars. Rail cars are switched from the private siding of a shipper to the terminal, or switched from the terminal to the private siding of the consignee. (Note: a siding is a section of rail line that runs from a railroad’s line into an industrial facility. If an industry using rail shipping does not have a siding, they will likely use (1) intermodal containers, or (2) use a cartage service to transfer goods to/from a rail terminal.)

**Longhaul**

Sometimes, linehaul. Terminal-to-terminal freight movements in transportation. Such long distance moves are distinguished from local freight movements.

**Detention/demurrage**

Penalty charges assessed by a carrier to a shipper or consignee for holding transportation equipment, i.e. trailers, containers, railcars, longer than a stipulated time for loading or unloading.

**Diversion/reconsignment**

Diversion is a tactic used by shippers to change the destination (consignee) of freight while the goods are in transit. The shipper will notify the carrier prior to the arrival of freight at the destination of the new consignee, and the carrier will adjust the freight routing accordingly. Reconsignment is a similar concept, except that the shipper notifies the carrier of the new consignee after the freight arrives at the destination, but (obviously) before delivery/unpacking. Carriers impose extra charges for these services typically, but they provide flexibility to the shipper.

**Transit privileges/stopoff charges**

Carriers may allow cargo to be stopped in transit from initial origin to final destination to be unloaded, stored, and/or processed before reloading and final shipment. Extra
charges are imposed for these transit privileges. Stopoff charges are levied for when shippers request that a shipment may be partially loaded at several locations and/or partially unloaded at several locations en route.

**Postponement**
A deliberate delay in committing inventory to shipment by a shipper. Usually, shippers utilize postponement in order to consolidate freight into larger shipments that have a lower unit transportation cost.

**Bulk cargo**
Cargo that is stowed loose on transportation vehicles, in a tank or hold without specific packaging, and handled by pump, scoop, conveyor, or shovel. Examples: grain, coal, petroleum, chemicals.

**Break-bulk cargo**
Cargo in-between bulk and containerized, that must be handled piece-by-piece by terminal workers (stevedores). Often stored in bags or boxes and stacked onto pallets. Smaller lift equipment (forklifts, small cranes) used than for containerized cargo, but more labor intensive.

**Pallet/skid**
A small platform, 40x48 inches usually, on which goods are placed for handling within a warehouse or a transportation vehicle such as a ship. Good for grouping break-bulk cargo for handling.

**Dunnage**
Wood and packaging materials used to keep cargo in place inside a container or transportation vehicle.

**SKU**
Stock-keeping unit. A line-item of inventory, that is a different type or size of good.

**Hundredweight/CWT**
100 pounds. A common shipping weight unit.

**Freight weight measures**
Short ton (American) 2000 lbs. Long ton (English) 2240 lbs. Metric ton (1000 kg.) 2204.6 lbs.

**Deadweight**
The number of long tons that a vessel can transport of cargo, supplies and fuel. It is the difference between the number of tons of water a vessel displaces “light” (empty) and the number of tons it displaces when submerged to the “load line.”

**TEU**
Twenty-foot equivalent unit. Method of measuring vessel load or capacity, in units of containers that are twenty feet long. A 40’ long container measures 2 TEUs. Exam-
ple: the maximum capacity for carrying 40' containers for a 3000 TEU vessel is 1500 containers; it actually might be less. Why?

**FEU**
Forty-foot equivalent unit. Method of measuring vessel load or capacity, in units of forty-foot long containers.

**Slot**
A place for a container onboard a container ship; typically, one TEU fits in a slot.

**Liner shipping**
Liners are vessels sailing between specified ports on a regular schedule; schedule is published and available to the public. Most large container shipping companies operate liner services.

**Tramp shipping**
An ocean carrier company operating vessels not on regular runs or schedules. They call at any port where cargo may be available. Sometimes used for bulk cargo shipping.

**Ocean conference**
Cartel of vessel operators operating between specific trade areas. Set cargo rates for liners between ports.

**Alliance**
Group of airlines or ocean carriers who coordinate and cross list schedules, and sell capacity on each other’s flights/voyages.

**Container leasing/railcar leasing**
Some companies specialize in the business of owning transportation equipment (containers or railcars), and renting them out to shippers or carriers. These companies often face significant equipment management problems.

**Lo-lo**
“Lift-on, lift-off”. Conventional container or cargo ships, in which quay cranes are used to load and unload containers or generalized cargo.

**Ro-ro**
“Roll On/Roll Off”. A method of ocean cargo service using a vessel with ramps which allows wheeled vehicles to be loaded and discharged without cranes.

**Hi-lo**
Container yard jargon for a forklift truck used for heavy lifting of containers.

**Straddle carrier**
Mobile truck equipment with the capacity for lifting a container within its own framework, and transporting containers around yards. Containers stacked in rows one across. Pros: Versatility, mobility, cost, labor. Cons: Maintenance, damage.
**Transtainer/RTG**
Rail or rubber-tired gantry crane. Large yard (ship or rail) container crane. Lifts from a stack of containers 5,6,7 wide, and deposits onto truck chassis or rail flatcar. Pros: Land utilization, maintenance. Cons: investment.

**Quay crane/portainer crane**
A quay is the dock. The portainer crane are the large cranes used to lift containers from truck chassis (or rail flatcar, or from the quay) and load onto a ship.

**EDI**
Information system design

Information system uses in SCM

- Book-keeping (orders, inventories)
- Fulfillment (through EDI)
- Tracking and tracing
- Decision support (forecasting, optimization)

Components of an information system

1. Input/data sources

1. Primary sources
   
   (a) Customers (current orders, past orders, surveys, ...)
   (b) Suppliers (contracts, invoices, ...)
   (c) Company records (operating reports, accounting reports, bills, ...)

2. Secondary sources
   
   (a) Trade journals
   (b) Published research reports
   (c) Consultants
   (d) Research contracts

Data captured on paper and/or electronic media
Real-time feeds of data (orders, fulfillment, tracking/tracing)

2. Database management system

Need: system to convert raw data into useful information, present information in a useful way to users, interface information with decision-support components

Relational Database

- data are organized into files or tables (sometimes, entities)
- data in different tables are linked to form links/relations between related data – hence “relational database”
- normalization of data – organizing data in different files to avoid storage of redundant information.
For ex., in an order processing system, the data for each customer order could include all details about the items ordered, but this leads to duplication/redundancy. Instead, data for each order contains only an item code, and a separate table contains the item codes and detailed descriptions/specifications of the item.

- Data in tables organized into fields and records
  - a field contains data for a specific type of data item (an attribute of the entity)
  - a record contains specified values for the fields, and represents an instance of the entity
    * For ex., a customer table may have fields for name (last, first), street, apt. number, city, state, zip code, tel. number, account balance, credit limit, and a customer ID number.
    * Each customer’s entry in this table is a record.

- Different types of relations:
  - one-to-one, eg. item description ⇒ item specification
  - one-to-many (many-to-one), eg. customer ⇒ many orders, order ⇒ many order items
  - many-to-many, eg. warehouse ⇒ many item types and item type ⇒ many warehouses

Important factors in database design and management

1. **Security**: only authorized persons can read, add, modify, delete records, fields, tables, queries, reports.

2. **Validity**: constraints on the contents of a field. e.g. weights all in the same units, prices in dollars or cents per unit

3. **Integrity**: cannot create unresolvable relations/links. e.g. cannot delete an entry in item type file as long as there are records in the order line file that refer/link to that item record.

4. **Consistency**: e.g. item number in the order line file must exist in the item type file.

5. **Redundancy**: minimize duplication of data by using a normalized data structure.

6. **Accident recovery**: ability to recover from incompletely-entered transactions, loss of data (backups), etc.
Demand forecasting in supply chains

Role of demand forecasting
Effective transportation system or supply chain design is predicated on the availability of accurate inputs to the modeling process. One of the most important inputs are the demands placed on the system. Forecasting techniques are used to predict, in the face of uncertainty, what the demands on the system will be in the future so that appropriate designs and operating plans can be devised.

A supply chain example
Draw a picture of the supply network for a single product. Explain that before production or supply systems are planned, we must understand the pattern of demand for the product that will be seen at the end customer locations. This demand pattern occurs over time. Production and distribution schedules, network configuration, mode choice, and routing decisions follow from there. Note that locating the end facilities (if they exist) may be part of the design problem; thus, customer demand may need to be predicted not necessarily at points in space, but over geographic regions.

A transportation sub-example
There are countless examples of the necessity of demand forecasting in designing a transportation system. Here are some:

- Airline demand forecasting: how many passengers from A to B, at what times of day, on what days of week; also, what customer types (willingness-to-pay)?

- FedEx demand forecasting: how much freight to be collected in a region, by day of week, destinations of that freight. Also: what is the distribution (for any region) of when the company will find out about the collections they need to make. Draw picture.

- TL forecasting: what truckloads, what origins, destinations, service requirements?

- Container leasing forecasting: how many containers, where, when?

Ideas to remember

- Forecasts are usually wrong. (Do not treat as known information!).

- Forecasts must include analysis of their potential errors.

- The longer the forecast horizon, the less accurate the forecast will generally be.

- Benefits can often be obtained from transforming a forecasted quantity into a known quantity (when this can be done, it comes at a cost however).
• Systems that are agile in responding to change are less dependent on accuracy in forecasts.

Forecasting Methodology Tree

![Exhibit 5-2 FORECASTING METHODOLOGY TREE](image)

Figure 2: Forecasting methodology tree.

**Extrapolation forecasting for time series data**

Often, problems in forecasting for logistics systems require the analysis of univariate time series data; often we are interested in the evolution of customer demand for a single product over time, and what the future demand will be for that product. (Recall: univariate means of a single variable, like demand).

One of the simplest and most prevalent means for forecasting time series data is through the use of extrapolation-based techniques. Assume for now that for our time series data, we have collected a number of historical observations $X_t$, for $t = 1, 2, ..., n$ periods up to and including the present. Further, suppose that these observations have been made at uniform intervals; for example, once a day, once a week, once a month, etc.

**Evaluating quality**

Any forecasting method can and should be evaluated by analyzing the errors produced by the forecasts; this is easily done once observations are made. Suppose that $e_t$ are the errors over a set of $n$ time periods:

$$e_t = Y_t - X_t$$
Errors should be compared against any assumptions made in the modeling process; for example, mean zero is common. In regression models, we might assume that the errors are normal; this should be verified.

The errors also can give a measure of the quality of the forecast, but this may be confounded with the uncertainty in the quantity to be forecast. One way to do this is to calculate the MSE:

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} e_i^2
\]

Note of course that this value has units (something\(^2\)), and thus the performance of a method for forecasting demand in barrels of oil is not directly comparable to forecasting number of microprocessors.

**Stationary time-series data**

Often it is reasonable to assume that an underlying time series process is inherently stationary around some mean value. Thus, an appropriate statistical model for the observed quantity at time \(t\) might be:

\[
Y_t = \mu + \epsilon_t
\]

where \(\mu\) is some unknown mean, and \(\epsilon_t\) are i.i.d. random variables with mean zero and variance \(\sigma^2\). When it is reasonable to model data in this way, it is appropriate to use moving-average or exponential smoothing methods.

**\(m\)-period Moving Average**

An \(m\)-period moving average is simply the arithmetic mean of the \(m\) most-recent observations:

\[
Y_t = \frac{1}{m} \sum_{i=t-m}^{t-1} X_i
\]

Example: Weekly orders of Atlanta Gateway CountryStore for a computer model: 200, 250, 175, 186, 225, 285, 305, 190. 3-period moving averages in periods 4,5,6,7,8: 208, 204, 195, 232, 272. Errors: 22, -21, -90, -73, 82.

Multiple step-ahead forecasts: Let \(Y(t-1)_T\) be a forecast in period \(t-1\) for future period \(t\) (so, \(Y_t \equiv Y(t-1)_t\)). With moving averages, since we assume that the underlying mean does not change and that our current moving average is the best estimate for that mean, \(Y(t-1)_T = Y(t-1)_t\) for all \(T \geq t\).

**Pros:**

- Simple to calculate, implement
• Easy update procedure: $Y_{t+1} = Y_t + \frac{1}{m}(X_t - X_{t-m})$

• Seasonality issues can be avoided if $m$ includes a full cycle of “seasons”

**Cons:**

• Problems with trend: Average “age” of data used is $\frac{m-1}{2}$. Always lags a trend.

Forecast error for moving average

\[
\text{var}(Y_t - X_t) = \text{var}(Y_t) + \text{var}(X_t)
\]

\[
= \frac{1}{m^2} \sum_{i=t-m}^{t-1} \text{var}(X_i) + \text{var}(X_t)
\]

\[
= \frac{1}{m^2}(m \sigma^2) + \sigma^2 = \sigma^2 \left( \frac{m + 1}{m} \right)
\]

**Simple exponential smoothing**

Another prevalent and popular forecasting method is exponential smoothing, in which the most recent forecast is combined with the most recent observation to generate a new forecast for a stationary series:

\[
Y_t = \alpha X_{t-1} + (1 - \alpha)Y_{t-1}
\]

where $\alpha \in [0, 1]$, is the smoothing constant. One should note that this method applies a smoothing constant to correct for the error in the previous forecast!

\[
Y_t = Y_{t-1} + \alpha(X_{t-1} - Y_{t-1}) = Y_{t-1} - \alpha e_{t-1}
\]

Since the specification equation in this case defines a recursion, we can expand it:

\[
Y_t = \alpha X_{t-1} + (1 - \alpha)\alpha X_{t-2} + (1 - \alpha)^2 Y_{t-2}
\]

\[
Y_t = \sum_{i=0}^{\infty} (1 - \alpha)^i \alpha X_{t-i-1} = \sum_{i=0}^{\infty} a_i X_{t-i-1}
\]
Hence, each previous demand observation is included to produce the newest forecast, but the weights given to each of these observations decreases exponentially (hence, the name). $\alpha > \alpha(1 - \alpha) > \alpha(1 - \alpha)^2 > \ldots$

Exponential smoothing methods will still lag the trend, but less so since the average age of the data used in the forecast can be made younger with an appropriate choice of $\alpha$. Weight given to data $i$ periods old (data in period $t - 1$ has age 0) is $\alpha(1 - \alpha)^i$. Therefore, we can calculate the average age as follows:

$$\sum_{i=1}^{\infty} i \alpha(1 - \alpha)^i = \frac{1 - \alpha}{\alpha}$$

To equate the average age of moving average data and exponential smoothing data, we can let:

$$\frac{m - 1}{2} \leq \frac{1 - \alpha}{\alpha}$$

And therefore, when $\alpha \geq \frac{2}{m+1}$ the average age of the exponential smoothing data is “younger” than the moving average data.

Finally, we can derive the theoretical variance of the exponential smoothing errors.

$$\text{var}(Y_t) = \text{var} \left( \sum_{i=0}^{\infty} (1 - \alpha)^i \alpha X_{t-i-1} \right)$$

$$= \alpha^2 \sum_{i=0}^{\infty} \text{var} \left( (1 - \alpha)^i X_{t-i-1} \right)$$

$$= \alpha^2 \sigma^2 \sum_{i=0}^{\infty} (1 - \alpha)^{2i}$$

Now, we recognize that we are left with a geometric series with rate $r = (1 - \alpha)^2$; the sum is then $\frac{1}{1 - r}$.

$$\text{var}(Y_t) = \frac{\alpha^2 \sigma^2}{1 - (1 - \alpha)^2} = \frac{\alpha \sigma^2}{2 - \alpha}$$

Now, we can compute the variance of the errors:

$$\text{var}(Y_t - X_t) = \sigma^2 (\alpha/(2 - \alpha) + 1) = \sigma^2 \frac{2}{2 - \alpha}$$

An exponential smoothing example:
Example: Weekly orders of Atlanta Gateway CountryStore for a computer model: 200, 250, 175, 186, 225, 285, 305, 190. 3-period moving averages in periods 4, 5, 6, 7, 8: 208, 204, 195, 232, 272. Errors: 22, -21, -90, -73, 82.

Now, let’s try ES with $\alpha = 0.3$: Suppose that the initial forecast is right on the money, $Y_1 = 200$. Forecast for period 2 is now $(0.3)(200) + (0.7)(200) = 200$. Period 3 is $(0.3)(250) + (0.7)(200) = 215$. Period 4: $(0.3)(175) + (0.7)(215) = 203$. 5, 6, 7, 8 are 198, 206, 230, 252.

MSE(MA) = 4198, MSE(ES) = 3362. However, neither really doing that well here.

**Double exponential smoothing with Holt’s Method**

Moving-averages and the exponential smoothing methods tend to lag behind trends, if they exist in the data. Sometimes, exponential smoothing can do a better job tracking trends although this is not always the case. Therefore, there are extrapolation methods specifically designed to capture trends in data.

A simple extension of exponential smoothing, known as Holt’s method, does a reasonable job of modeling trends in data. In Holt’s method, we will track two forecasted quantities for the time series. The first, $R_t$, is an estimate of an intercept in period $t$. The second, $G_t$, is an estimate of the trend (or slope) in period $t$.

Given these two forecasted quantities, the multiple-step ahead forecast is given as follows:

$$Y_{t+\tau}(t-1) = R_{t-1} + (1+\tau)G_{t-1}$$

and our standard one-step forecast $Y_t$

$$Y_t = R_{t-1} + G_{t-1}$$

Let’s now explain how to update our estimates $R_t$ and $G_t$ each period after the the data $X_t$ becomes known. First, we will now employ two smoothing constants, $\alpha \in [0, 1]$ for the intercept and $\beta \in [0, 1]$ for the slope.

Update equations: first we update the estimate of the intercept by combining the observed demand in this period, with the previous period’s forecast of demand.

$$R_t = \alpha X_t + (1-\alpha)(R_{t-1} + G_{t-1}) = \alpha X_t + (1-\alpha)Y_t$$

Now, we re-estimate the slope by combining a new estimate based on the difference in intercepts, with our previous slope estimate.

$$G_t = \beta(R_t - R_{t-1}) + (1-\beta)G_{t-1}$$

**Simple linear regression methods**
When we suspect trend in a time-series, we might postulate that the time series data fits a linear model, as follows:

\[ X_t = a + bt + \epsilon_t \]

where \( a \) and \( b \) are unknown constants, and \( \epsilon_t \) are the error random variables that we assume are i.i.d. normal random variables with mean 0 and variance \( \sigma^2 \). Given, therefore, some estimates of \( \hat{a} \) and \( \hat{b} \), we can use this type of model for forecasting:

\[ Y_t = \hat{a} + \hat{b}t \]

with the multiple step-ahead forecast given by:

\[ Y_{t+\tau}(t-1) = \hat{a} + \hat{b}(t + \tau) \]

Given a time series, \( X_i, i = 1, 2, ..., t-1 \), we can create a prediction for the next period \( t \) by determining reasonable estimates of \( \hat{a} \) and \( \hat{b} \). The most logical way to do so is to use a set of \( m \) most recent observations of \( X_i \), in fact as far back as you believe the current trend should hold.

It can be shown that the least-squares method can be employed with this data to develop good, unbiased estimates of \( a \) and \( b \). In this method, we attempt to minimize the sum of the squared errors in our data. Let \( \hat{X}_i \) be the predicted value for observation \( i \), \( \hat{X}_i = \hat{a} + \hat{b}i \). The goal then is to minimize:

\[ g(\hat{a}, \hat{b}) = \sum_{i=t-m}^{t-1} (X_i - \hat{X}_i)^2 = \sum_{i=t-m}^{t-1} [X_i - (\hat{a} + \hat{b}i)]^2 \]

It turns out this is easy to do by simply setting the partial derivatives of this function with respect to \( \hat{a} \) and \( \hat{b} \) equal to zero and solving simultaneously. The details of this process are quite simple, and will be skipped in this section.

Dealing with Seasonality

None of the methods described up to this point explicitly deal with a common feature of time-series demand data: seasonality. Seasonality is the tendency of time-series data to exhibit behavior the repeats itself every \( q \) periods. When talking about seasonality in time-series data, we use the term “season” to represent the period of time before behavior begins to repeat itself; \( q \) is therefore the length of the season.

For example, air travel is known to exhibit several types of seasonal behavior. For example, if we example pax travel demand week after week, we may see that certain days of the week always exhibit relatively higher demands than others; for example, travel on
Fridays, Sundays might be highest, followed by Mondays, followed by Thursdays, and finally Sats, Tues, Weds. In this case, the length of the “season” is 7 periods (days). In another example, suppose we look at monthly demand for PCs. This may also exhibit a cyclical pattern that ramps up for the Christmas holiday. Here the season length is 12 periods (months).

**Using seasonal factors to remove seasonality from a series**

One of the most common and popular ways to remove seasonality from a time-series is through the use of seasonal factors. Suppose a season has \( q \) periods; then, let \( S_j \) for \( j = 1, ..., q \) be the seasonal factor for season \( j \). The set of seasonal factors is used to transform the time-series as follows:

\[
\bar{X}_i = \frac{X_i}{S_j} \quad \forall \ i = 1, ..., t - 1
\]

where \( j = i \mod q \). Think of the transformation in this way: the seasonal factor represents how the expected value in a particular period relates to the mean. A seasonal factor of 0.5, for example, would mean that values in this season tend to be about one-half the size of the series mean. A factor of 1.2 would mean they tend to be 20% higher than the mean. By transforming the series as above, the result is a time-series that no longer exhibits seasonality; recognize that it may still contain trend.

The transformed series \( \bar{X} \) can now used in any of the methods described above, such as MA, ES, Holt’s, or a regression-based model, creating a forecast \( \bar{Y}_t \). Remember, of course, that as the method progresses each new observed data point must be scaled by its seasonal factor. Since now the resulting forecast is for the transformed series, it must be now transformed back:

\[
Y_t = \bar{Y}_t S_k
\]

where again, \( S_k \) is the seasonal factor for time period \( t \), \( k = t \mod q \).

**Calculating seasonal factors**

Since seasonal factors represent deviation around a suspected mean, we want the average multiplicative seasonal factor to have a value of 1. Thus, we enforce the following:

\[
1 = \frac{1}{q} \sum_{j=1}^{q} S_j
\]

One simple method for calculating reasonable seasonal factors when at least two prior seasons of data is available is to use a centered moving average approach, where the number of periods averaged is equal to the length of the season. In a centered moving average, the average for \( q \) adjacent periods \( 1, 2, ..., q \) is centered at period \( \frac{q+1}{2} \). If \( q \) is
even, the resulting averages are off-period, so they are moved back on period by averaging adjacent periods.

<table>
<thead>
<tr>
<th>Period</th>
<th>Demand</th>
<th>Cent MA(4)</th>
<th>On Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>18.25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>18.50</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>19.125</td>
<td>19.50</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>19.50</td>
<td>20.00</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
<td>20.50</td>
<td>21.125</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>21.75</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Period</th>
<th>Demand</th>
<th>Cent OP MA(4)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>18.81</td>
<td>0.532</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>18.81</td>
<td>1.063</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>18.50</td>
<td>1.405</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>19.125</td>
<td>0.888</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>20.00</td>
<td>0.600</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
<td>21.125</td>
<td>1.089</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>20.56</td>
<td>1.463</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>20.56</td>
<td>1.070</td>
</tr>
</tbody>
</table>

3. Average Factors

Period 1: \((0.532 + 0.600)/2 = 0.566\)
Period 2: \((1.063 + 1.089)/2 = 1.076\)
Period 3: \((1.405 + 1.463)/2 = 1.434\)
Period 4: \((0.888 + 1.070)/2 = 0.979\)

4. Normalize

Sum of factors is 4.055. To make the sum 4 (so the average factor is 1), multiply each factor by \(4/4.055\):

Period 1: \(S_1 = 0.558\)
Period 2: \(S_2 = 1.061\)
Period 3: \(S_3 = 1.415\)
Period 4: \(S_4 = 0.966\)
**Updating seasonal factors via a smoothing method: Winter’s**

Finally, instead of recalculating seasonal factors as new information becomes known during a forecasting cycle, it might make sense to update the factors themselves using a smoothing process. Let $\gamma \in [0, 1]$ be a smoothing constant. Then, we could update factors each time a new piece of data becomes available as follows:

$$S_k \leftarrow \gamma \frac{X_t}{Y_t} + (1 - \gamma)S_k$$

where $k = t \mod q$. Our most recent observation of the seasonal factor is simply the ratio between the observed data and the de-seasonalized forecast estimate. Note that each time a factor is updated in this way, the set of factors must be re-normalized.
The Bullwhip Effect in Supply Chains

Consider the simplest representation of a supply network; a simple chain of suppliers from a provider of raw materials to a retail customer. Raw material supplier, production facility, distribution center, retailer. Each of these so-called components will face a stream of demands from a buyer (sales) and must in turn place orders to a supplier (or, decide on production levels).

Bullwhip effect
Now consider a single component in this chain, say first the retailer. It has been observed frequently in supply chains that sales and order patterns look like this:

[Draw picture]

The bullwhip effect refers to phenomenon where:

1. Orders to the supplier have larger variance than sales to the buyer
2. Variability in orders amplifies as you move up the supply chain

The key are these larger variances; create many problems in supply chain operations.

Examples

- P&G pampers supply chain (customers, distributors, orders to suppliers)
- Hewlett-Packard printers (customers, reseller, printer division to IC)

Symptoms of demand distortion

- Excessive inventory
- Insufficient or excessive capacity
- Poor customer service
- Uncertain production planning
- High costs for recovery (expedite, overtime)

Inventory problems in supply chains

- grocery supply chains: (100 days of inventory; $75 billion inventory, $300 billion industry)
- pharmaceutical: 100 days (manu: Bristol-Myers, dist: McKesson, retail: Longs)
Causes of the bullwhip effect

Demand signal processing
Each member in the supply chain makes its own forecast of the demand that it must meet. These forecasts are based on the order history of immediate customers; the signal. Thus, “orders” do not reflect underlying demand; distortion.

Forecasting example with a smoothing method
When using a smoothing-based method, such as simple, double (Holt’s), or triple (with seasonal factors) exponential smoothing, or any forecasting method that assumes demands are serially-correlated (autocorrelation), each new piece of information updates future forecasts of demand.

Order quantity batching
Discuss: transportation scale economies with shipment size. Truckload vs. LTL; even LTL shipment rates.
To reduce order processing costs and transportation costs, orders are often “batched” together. Periodic ordering is when orders are placed on a weekly, biweekly, or monthly schedule; variability is that “demand” is high one week and then nonexistent for following weeks (MRP—once a month). Often, this effect is enhanced because many customers might have identical periodic policies. Push ordering is when company faces regular surges in demand, maybe as result of salesperson evaluation procedures.

Quantity discounts
Larger orders are placed than required, and they come less frequently. This distorts the demand information by increasing its variance.

Price fluctuations and promotions
Price incentives tend to lead to a “batching” effect on orders; they all come when discounts are present. 80% of grocery distributors orders to manufacturers in forward-buy price incentive arrangements. Again, customer buying does not represent demand, and order variation is greater than demand variation.

Gaming by buyers
Purposeful distortion of demand data to a supplier! For example, to encourage the supplier to increase production capacity, or to obtain the desired amount of goods if production capacity is less than total demand and the supplier rations supply in proportion to order supply. True demand is then distorted to manufacturer, resulting in poor production scheduling and capacity expansion possibly. Ex. HP Laser Jet III, DRAM markets, IBM Aptiva. This same effect also occurs inside companies, for example, the
sales department may provide biased sales forecasts to the production department to encourage large production runs, in turn because sales department is afraid of missing sales bonuses due to insufficient production.

**Long lead times**
When the time between order placement and the receiving of an order is long, components must keep safety stock inventory to buffer against demand variability. The longer the lead time and higher demand variability, the more safety stock required. Thus, in systems suffering from bullwhip, safety stocks get larger up-the-chain. Since orders are placed to replenish safety stocks, again they distort true demand information.

**Irrational behavior**
For example, panic ordering when a backlog develops. Again, true demand information is distorted.

**Measures to reduce the bullwhip effect**

**Sharing of information between components of the supply chain**
In particular, providing members further upstream in the supply chain with raw, accurate, and timely end-consumer demand information. Both sites update demand forecasts with the same raw data. *Sell-through* data: goods withdrawn from reseller’s warehouses. EDI enables information sharing via established standards and protocols.

**Reduce ordering and transportation costs**
Reduce order processing costs via EDI, reduce the transportation costs of small batches by encouraging customers to determine which different goods they might be able to pool into larger shipments. Mixed-SKU inducements. Use of third-party logistics providers, who can do the pooling shipments between companies.

**Develop incentives to buyers to place regular, predictable orders**
Orders can be spaced at a regular frequency, and be kept within narrow predictable ranges. Preferably, resupply to a number of customers can be *staggered* throughout the month, week, etc. Incentives to stick to an agreed-upon plan can be in the form of pricing penalties, or discounts, or other preferential treatment. Revenue management.

**Avoid demand-distorting price incentives**
Eliminate to the extent possible quantity discounts. Get rid of regular promotions.

**Implement pricing that induces regular purchasing**
Supply (orders) should be matched to demand (sales). Every-day low-pricing (EDLP) for example is a good course of action.

**Eliminate shortage gaming**
In shortage situations, allocate products to buyers based on past sales records rather than on the amount requested in a particular order. Share production information with buyers to eliminate the amount of purchasing gaming.
Shorten lead times
Improve operational efficiencies in order processing and transportation so that orders are filled more quickly.

Vendor-managed inventory
Supplier component controls resupply to a buyer component. For example, supplier forecasts its buyer’s customer demand using buyer’s sales information, it decides when to resupply, and the order quantities etc. One step beyond info sharing, the downstream component becomes a passive partner. Supplier has price incentives from buyer to perform well.

Problems in sharing of information in supply chains

Exercise care when sharing cost data
When buyers know your production yields or the costs of goods sold, they can exhibit pressures on you to provide lower prices or better deals.

Confidentiality among competitors
Suppliers may supply competing buyers. Sales data should not be made available to a supplier unless a guarantee is provided that it won’t be provided to your competitors. Measures (contracts, incentives) must be taken to avoid this sharing behavior.

Determination and specification of confidential information
Some info is confidential and some should be shared. This must be established, and the means for separation must be generated.

Development/retooling of compatible systems
Data formats, units, financial years, accounting procedures may need to be compatible among groups. Consistent formats and procedures might need to be developed.

Systems integration problems
Meshing together potentially different hardware or software systems.

Paying for the Information Sharing System
Who should contribute how much to the development and maintenance of these systems.

Antitrust laws
Data sharing strategy must be checked to ensure that no collusive behavior is present. Shared data relayed back between competitors can be used for price-fixing schemes.

Data quality
Make sure the data you share is timely and accurate.

Information sharing may solve the following problems

1. Irrational behavior by members of the supply chain. Hopefully!

2. Variance in buyer’s order quantities due to batching of orders, quantity discounts, promotions, trade deals etc. may be reduced. Buyer’s orders should be more
predictable by the supplier! Further, this should no longer influence a supplier’s demand forecasts as significantly since he can forecast consumer demand directly; no amplification of variance is now necessary for example when planning safety stocks etc.

3. The effect of gaming can also be reduced or eliminated; suppliers know buyers true demands, and buyers know suppliers capacities.

**Information sharing may not help**

1. Buyers orders may not become any more predictable just because information is shared. And suppliers must still fill the orders received by their suppliers. Better forecasts of consumer demand do not solve all problems if the buyer still generates highly variable orders.

2. Information sharing cannot shorten lead times. Efficient order processing and communication of orders can reduce lead times.

**Shortcomings of information sharing can be dealt with by**

1. Supplier-buyer relationships such as vendor-managed inventory.

2. Other non-information-sharing strategies mentioned earlier that induce predictable ordering behavior.

**Some details from the Lee Management Science Article (1997)**

*Demand Signal Processing*

In this section of the paper, the authors show that when faced with a certain type of serially-correlated demand function, a cost-optimizing ordering strategy will result in the decision-maker increasing the order-up-to level $S_t$ when demand surges, and decreasing it when demand comes in significantly below expected. The net result is that the variance of the order sizes tends to be larger than the underlying demand variance.

To understand this result, you must first understand the assumed demand process upon which the result is based:

$$X_t = \mu + \rho X_{t-1} + \epsilon_t$$

where $\mu$ is a constant, $-1 < \rho < 1$ is a coefficient that relates the demand in the current period to that in the prior period, and $\epsilon_t$ is assumed to be normal with zero mean and variance $\sigma^2$. In forecasting terms, this model is referred to as an AR(1) model with constant. The idea here is that significant autocorrelations exist between the errors in adjacent periods, but decay rapidly after that. Autocorrelation in a time series is
expressed in terms of the lag; for example, lag-$k$ autocorrelation refers for the tendency of the error in the current period to depend on the error in period $t-k$. 
Note that this process exhibits a drift in the mean over time, when $0 < \rho < 1$. Since:

$$X_t = d \frac{1 - \rho^t}{1 - \rho} + \rho^t X_0 + \sum_{i=1}^{t} \rho^{t-i} \epsilon_t$$

and therefore:

$$E[X_t|X_0] = d \frac{1 - \rho^t}{1 - \rho} + \rho^t X_0$$

Notice that this is a somewhat bizarre functional, whose limit approaches $\frac{d}{1-\rho}$ asymptotically in the limit. It’s not clear that many true demand processes will fit this statistical model well.

Nonetheless, the result is the following:

**Theorem 0.1** In the above setting, we have:

(a) If $0 < \rho < 1$, the variance of orders is strictly larger than that of sales;

(b) If $0 < \rho < 1$, the larger the replenishment lead time, the larger the variance in orders.

In the paper, it is shown that the variance in the order size in period $t$ is strictly greater than the demand variance:

$$\text{var}(z_t) = \text{var}(X_0) + \frac{2\rho(1-\rho^{v+1})(1-\rho^{v+2})}{(1+\rho)(1-\rho)^2} \sigma^2$$

To show that (b) is true, we can show that the second expression is always increasing. Rewrite that expression as:

$$h(v) = C f(v) g(v)$$

where $f(v) = 1 - \rho^{v+1} > 0$ and $g(v) = 1 - \rho^{v+2} > 0$, and $C$ is a positive (clearly) constant for $\sigma^2 > 0$.

$$h'(v) = C (f'(v) g(v) + f(v) g'(v))$$

Now, since $f'(v) = -\rho^{v+1} \ln \rho$ and $g'(v) = -\rho^{v+2} \ln \rho$ are both positive as well, the result holds.
Order Batching

In this section, the author shows that when a number of different buyers correlate their orders in time, that is all order at the same time, the bullwhip effect tends to increase. Suppose there are $N$ buyers, each using a periodic review inventory system cycling every $R$ periods. Suppose each buyer $j$ faces a demand of $\xi_{jt}$ in period $t$, where these variables have identical mean $\mu$ and variance $\sigma^2$: stationary demand. First, we calculate the total demand faced by the buyers in a given period:

$$X_t = \sum_{j=1}^{N} \xi_{jt}$$

Since we assume these variables are independent, $E[X_t] = N\mu$ and $\text{var}(X_t) = N\sigma^2$.

We can now compare the variance in the ordering patterns under two different configurations. First, consider a situation where a random number $n$ of retailers order in each period: $n$ is clearly binomial with probability $\frac{1}{R}$ and number $N$. Hence, $E[n] = \frac{N}{R}$ and $\text{var}(n) = N\frac{1}{R} \left(1 - \frac{1}{R}\right)$.

$$Z_t^* = \sum_{j=1}^{n} \sum_{k=t-R}^{t-1} \xi_{jk}$$

[here, each supplier replaces the expended demand in the previous $k$ periods]

Now, we can calculate the expected value and variance:

$$E[Z_t^*] = E_n[E[Z_t^* | n]] = E[nR\mu] = N\mu$$

since $E[\sum_{k=t-R}^{t-1} \xi_{jk}] = R\mu$. To calculate the variance, we use the conditional variance formula:

$$\text{var}(Z_t^*) = E_n[\text{var}(Z_t^* | n)] + \text{var}_n(E[Z_t^* | n])$$

$$= E_n[nR\sigma^2] + \text{var}_n(nR\mu)$$

$$= N\sigma^2 + R^2 \mu^2 \frac{N}{R} \left(1 - \frac{1}{R}\right)$$

$$= N\sigma^2 + \mu^2 N(R - 1)$$

We can now compare this scenario to the case when the set of buyers completely coordinates its orders; positively-correlated ordering. The case when all buyers order simultaneously can be represented by changing the distribution of $n$, the number of buyers ordering on a given day:
Given this distribution, it is clear that $E[n] = \frac{N}{R}$. The variance can be computed as follows: $E[n^2] - E[n]^2 = \frac{N^2}{R} - \frac{N^2}{R^2} = \frac{N^2}{R}(1 - \frac{1}{R})$. Now, defining the total ordered amount in any given time period in this scenario as $Z_t^c = \sum_{j=1}^{n} \sum_{k=t-R}^{t-1} \xi_{jk}$, and using the same steps as earlier:

$$E[Z_t^c] = E_n[E[Z_t^c|n]] = E[nR\mu] = R\mu \cdot \frac{N}{R} = N\mu$$

$$\text{var}(Z_t^c) = E_n[\text{var}(Z_t^c|n)] + \text{var}_n(E[Z_t^c|n])$$
$$= E_n[nR\sigma^2] + \text{var}_n(nR\mu)$$
$$= N\sigma^2 + \mu^2 R^2 \frac{N^2}{R} \left(1 - \frac{1}{R}\right)$$
$$= N\sigma^2 + \mu^2 N^2 (R - 1)$$

The result is of course, that the correlated order quantities have greater variance. Note that in this problem since we are comparing identical supply chains with the same per period demand, comparing variances is a reasonable way to compare the magnitudes of the bullwhip effect under the different scenarios. Note, though, that a true measure of the effect should scale the variance against the underlying customer demand in a unitless way:

$$BWE = \frac{\text{var}(Z_t)}{E[X_t]^2}$$

and in fact, a better measure would be to scale the effect against the underlying customer demand:

$$BWE = \frac{\text{var}(Z_t)}{\text{var}(X_t)}$$
Order Management

Order cycle
Collection of activities and events beginning with order placement by a buyer until the delivery of all ordered items by the supplier.

order cycle time: Time from placement of order to receipt of goods ordered.

Order cycle time distribution

Order Management

Activities

1. Order preparation
2. Order transmittal
3. Order entry
4. Order filling
5. Order status reporting

Order preparation
Preparation of a “shopping list” by a buyer. How:

- Handwritten form
- Visiting salesperson
- Telephone
- Electronic order placement
  1. e-mail
  2. web interface electronic forms
  3. proprietary software electronic forms
Generating SKUs for orders

- “Catalog” searching and manual entry
- Electronic DB searching and entry (website, sales rep laptop, proprietary software)
- Barcode scanning
- Automated order generation from MRP/ERP, SCM system

Order transmittal
Communication of prepared order from buyer to seller. How:

- Mail
- Sales representative (traveling or on-site)
- Fax
- Telephone
- E-mail
- Web
- EDI/XML message

Order entry
Receipt and data processing of order. Into system.

1. Check accuracy
2. Check availability
3. Check credit status
4. Placing of backorders or production orders, order cancellation
5. Transmitting order filling instructions (to WH, DC, plant, etc.)
6. Billing and accounting

Order filling (picking)

1. Purchasing and/or production
2. Stock retrieval and order picking
3. Packing
4. Shipment scheduling/transportation contracting

5. Freight documentation

**Order status reporting**

- Communication with customer re: order status, estimated delivery time
- Informing customer of delays
- Tracing and tracking

**Factors affecting cycle time**

1. Processing priorities
2. Parallel vs. serial processing
3. Order filling accuracy
4. Order condition
5. Order batching for more efficient picking
6. Shipment consolidation
7. Transportation mode and service
Cheezy-Wheezy Case

Idea: Gourmet cheese provider wants to augment retail store sales with a catalog business. They decide to use a mailed-in order form. This creates some order processing and customer service challenges.

Handling stockouts

stockout: a customer order cannot be filled due to lack of inventory/capacity. Negative consequences: customer may choose a different product yielding less profit, or may cancel order completely, choose a competitor. Since order form is mailed, difficult to inform customers of stock problems before receipt of order.

1. Contact customer with options
   - How to contact. Require customer to give phone, e-mail address, preferred method
   - Options to present

2. Backorder item, give customer an approximate due date

Packaging costs

Costs of the packaging material used to send the product. Of course, for larger shipments the packaging costs are larger. In the case, the sales manager estimated that packaging costs for a 1-pound unit of cheese would be approximately $0.50, significant in that the same pound of cheese sells in retail for $2.00.

It would appear in this case that it would be easiest to incorporate packaging costs directly into one of two other prices quoted to the consumer: the unit price of the cheese, or the shipping cost quoted.

   - Shipping costs might be a better place to inflate prices, however, so customer does not find the cheese price inflated.

   - Additionally, since packaging costs are “about the same per order” it would be simple to inflate shipping costs by a fixed amount + variable amount by weight.

   - If packaging costs truly don’t vary with weight, it could be a separate line item. But, more customer confusion can result.

Transportation costs

Costs for shipping orders. “Orders” should probably be restricted to a single destination; multiple orders would have to be place for multiple destinations. Cheezy-Wheezy has chosen a parcel delivery option with distance and weight “tapers”. As distance or weight (or both) increase, the unit rates decrease (economies of scale!). UPS Ground generates rates this way. USPS Priority Mail has weight tapers, but no distance tapers (within the U.S.)
• A small table with distance from New Glarus on one axis and total shipment weight on another axis could allow customers to determine shipping costs. Determine appropriate maximum distance, weight to include. Number of levels (space vs. accuracy).

• How far am I from New Glarus? Who knows. Table with each state less accurate, and more space. Regions?

• If customers calculate incorrectly, then refunds/bills?

**Three order preparation/transmittal options**

1. Mailed order form
   
   (a) Transmittal time and accuracy
       • 2-7 days from mail date; extra time for “mailing”
       • high potential for errors: wrong SKUs, customer information
       • no immediate verification
   
   (b) Calculation/presentation of shipping charges
       • Rougher approximation
       • Uses order form space
       • Customer sees and calculates; high potential for confusion, error
   
   (c) Order processing costs
       • Prepaid postage
       • Order entry costs low
       • Cancellations and return costs high

2. Toll-free phone number
   
   (a) Transmittal time and accuracy
       • Instantaneous
       • low potential for error
       • immediate verification
   
   (b) Calculation/presentation of shipping charges
       • Can be calculated exactly, and relayed to customer -or-
       • Catalog table could be provided for customer
   
   (c) Order processing costs
       • Order entry costs high: number + staffing
       • Low costs of errors
3. Well-designed web site

(a) Transmittal time and accuracy
  - Instantaneous
  - very low potential for error
  - immediate verification

(b) Calculation/presentation of shipping charges
  - Can be calculated exactly and displayed for customer

(c) Order processing costs
  - Order entry costs low
  - Very low costs of errors
Handy-Andy Case

Idea: Seller of trash compactors. Retailers set prices, but stocked only an item or two. Factory distributor would fill the order placed by retailers, received 9% of wholesale. During a customer service survey to decide whether to extend a warranty from 1 to 2 years, distribution department worker uncovered a different problem. Distributors would contact customers, tell them their product was out-of-stock, and have them cancel their original order. Distributors would then attempt to sell a higher line model, at the same price, to these customers directly. Often, the sales went through. Additionally, they claimed that customers would receive better service when buying direct, and that they, not Handy-Andy, Inc. stood behind the 1-year warranty. They provided better installation service (faster, more follow-up) to customers than if orders were being filled for dealers. Appeared not to be happening in all markets.

Customer service problem?
Little evidence of a customer service problem, since the end customers are being kept happy. Except the anecdotal evidence that distributors are not installing the dealer-sold units well. No evidence of lost sales. There is probably customer confusion.

Marketing channel problem?
*marketing channel:* arrangement of components designed to bring a product to market. Here is where the big problem lies. The channel was designed in a certain way: dealers sell to customers, and distributors fill the orders (delivery and installation). Distributors are obviously finding higher margins by selling directly to the customers. Existing system may be working out better for the customers, but it may or may not be sustainable and it was certainly not planned.

*Why is this happening:* Under current setup, dealer receives retail price and pays a wholesale price to distributor. Distributor receives 109% of wholesale for an installed product. Since retail prices are probably larger (say 125% of wholesale), distributor can earn larger profits by selling direct.

Distribution channel problem?
No. Factory distributors were always to be the main source of delivered and installed compactors, and this remains the case. Although, it could be argued that there is a problem in order fulfillment.

Exploitation of dealers
If dealers felt exploited and losing money, one would expect them to complain to Handy-Andy. The text does not indicate any complaining was found by the dealers. Dealers may be receiving “kick-backs” from the distributors.

Warranty issues
*distributors standing behind the warranty:* This would only be a customer service problem if it created a perception in customers’ minds that Handy-Andy was unwilling or unable to offer the advertised warranty protection. Otherwise, customers probably do not care who offers the warranty protection, and perhaps going through the distributor for warranty
service would occur anyway.

*if it were perceived as an image problem:* Handy-Andy should probably alter the wording on the warranty card and perhaps in store displays emphasizing the company’s role in servicing warranty claims.
Transportation Rates and Costs

Rate characteristics by mode
Measured in cents/ton-mile of revenue, for domestic shipping.

<table>
<thead>
<tr>
<th>mode</th>
<th>1990</th>
<th>1998</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rail</td>
<td>2.66</td>
<td>2.34</td>
</tr>
<tr>
<td>Truck</td>
<td>24.4</td>
<td>26.2</td>
</tr>
<tr>
<td>Barge</td>
<td>0.76</td>
<td>0.74</td>
</tr>
<tr>
<td>Pipeline</td>
<td>1.44</td>
<td>1.40</td>
</tr>
<tr>
<td>Air</td>
<td>64.6</td>
<td>84.1</td>
</tr>
</tbody>
</table>

So, what is the basis for the rates that are charged by transportation carriers? The two main influences are the costs of supplying the service, and the demand for the service by customers. Let’s examine this further.

Rate structures of common carriers

Related to shipment size
Rates depend on shipment size, and most typically size is measured in units of weight or mass. Rates are quoted in $ per cwt (100 lbs). Usually several categories, or “rate breaks”, based on several weight cutoffs.

1. For LTL or LCL shipments, minimum total rate for quantities below a minimum threshold, then several weight categories with different rates.

2. For FTL or FCL, rate only depends on equipment size ordered.

3. Incentive rates: for loads utilizing multiple vehicles, or even trainload rates in the case of unit trains (why? because terminal costs are reduced)

4. Time-volume rates: encourage shippers to send minimum quantities regularly, in effort by carriers to ensure regular flow of business. (here “volume” refers to freight throughput, not cube)

Related to distance

1. Uniform rates: rate independent of distance (e.g. USPS Priority Mail. Why? Most of the cost is in handling.)

2. Proportional rates: Fixed rate + variable rate per distance. (no taper, but still includes EOS. why?) Truckload rates often like this.

3. Tapered rates: Increase with distance, but at a decreasing rate. Especially in air transportation.

Related to demand
Often, carriers adjust rates to what shippers are willing/can afford to pay: “what the market will bear.”

Related to product shipped
Products are grouped into a classification for the purposes of determining rates. Classes depend on the factors of density, stowability, ease of handling, and liability:

1. weight/volume
2. value/weight or value/volume
3. liability to loss, damage, or theft
4. likelihood of injury/damage to other freight, to personnel, to vehicles and equipment
5. risk of hazardous materials
6. security of container or packaging
7. expense of and care in handling
8. rates on analogous articles, competitive products

Class rates
Not quoted for every imaginable product, but products grouped into classes according to the above characteristics. Uniform Freight Classification; some others use National Motor Freight Classification. Rates then based on product class, distance, and quantity.

Contract rates
Regular shippers and carriers enter into contracts for regular transportation of specified products. Reflect the class rates, but are often substantially less to encourage regular shippers to use the carrier. There may be minimums, but often are not enforced strictly.

Freight-all-kinds
Single freight for a variety of product classes; e.g. products sold by large department stores or products consolidated by freight forwarders. Still depends on distance and quantity.

Miscellaneous rates

1. *Cube rates*: For low-density product classes, rates may be quoted per volume rather than per weight.

2. *Import/export rates*: Domestic transportation of products imported or to be exported may quote special lower rates.

3. *Deferred rates*: Special low rates if shipper is willing to accept delays in shipment. Carriers use such freight to fill up spare capacity.
4. *Released value rates*: Carrier’s liability for loss and damage to goods is limited to a fixed amount per weight. Special rates are offered in this case, e.g. with household goods where value is difficult to determine.

**Special service charges**

1. Stopoff charges
2. Diversion and reconsignment
3. Transit privileges
4. Special protection (equipment) e.g. refrigeration
5. Pickup and delivery/cartage from terminal
6. Switching
7. Demurrage/detention

So, why are the rates like they are? Again, they reflect both the costs borne by the transportation company (supply), and the value of the transportation service to the customers (demand).

**Cost characteristics by mode**

*Fixed costs*: right-of-way, vehicles and other equipment, terminals, other overhead.

*Variable costs*: fuel, maintenance (oil, tires, service), drivers.

**Rail**

**Motor Truck**
Lowest fixed costs: trucks, trailers, and terminals. High variable costs: drivers, fuel, oil, tires, maintenance. Fuel costs include fuel taxes. Lower terminal costs than rail.

**Water**
Medium-high fixed costs: expensive vehicles and terminals, no right-of-way infrastructure. Variable costs low: low speed, high capacity vehicles. Terminal costs lower than rail, higher than motor trucks; harbor fees and loading/unloading costs.

**Air**
High fixed costs: very-expensive vehicles, terminal facilities. Variable costs medium-high: high fuel costs, relatively low-capacity vehicles: decrease with distance due to take-off
and landing fuel inefficiencies, labor, maintenance. Terminal costs: airport landing fees, storage, space leasing, load/unload costs.

**Pipeline**
High fixed costs: infrastructure (pipe, pumping stations, storage tanks). Low variable costs: energy for operating pumps. Efficiency of pumping: diameter and roughness of pipeline, more efficient at low speed.

**Choice of Mode and Carrier**
Ten most important factors for shippers (Chow and Poist, 1984):

1. Costs (total door-to-door)
2. Loss and damage: likelihood
3. Loss and damage: claims handling
4. Transit time reliability
5. Rate negotiation experiences
6. Shipment tracking/tracing
7. Door-to-door transit time
8. Pickup/delivery service quality
9. Single-line service availability (no interlines)
10. Equipment availability
The Costs in a Logistics System

Recall that transportation is a component of a logistics system, but it is not the only important cost component. When we model these systems, it is often advisable to account for a variety of costs. This is especially true when analyzing a supply network, since the costs of holding inventory are often important.

Cost Classification
It is useful to classify costs into four essential types:

1. Transportation costs: movement via vehicle, loading/unloading
2. Handling costs: packing/unpacking “containers” (boxes, bags, pallets), intra-facility movements such as moving into and out of storage
3. Inventory costs: opportunity cost of capital for time waiting
4. Facility rent costs: economic “rent” for facility space, storage infrastructure and maintenance

Who bears the costs
It should be noted that every logistics system will incur the costs as defined above, but different parties might pay for these costs. How should a system be optimized when considering one party? For example, shippers may bear some of the costs, carriers may bear others, and some costs may be passed on to shippers from carriers. Since the objective is efficiency, in many problems it is reasonable to consider all costs regardless of who pays them. For example, in optimizing a system for a producer of goods, it may be short-sighted to not count inventory costs at the consumer. Ignoring these costs may lead to a situation in which the producer sends large shipments that increase the inventory costs for the consumer, which may (a) decrease the price the consumer would pay for the goods, (2) make the consumer less likely to participate.

Normalizing costs When analyzing systems, costs should be normalized into a cost per time period, or a cost per quantity. In this discussion, we imagine that we ship some unit size of good called an “item” and normalize all costs into per item costs. It is also common to look at cost per time period, such as cost per year or cost per month.

To explain the basic concepts, consider the case where a single origin producing items at a constant rate $D_0$ is supplying a single destination. The destination consumes items also at the same constant rate, $D'_0$.

Explain cumulative count curves.

Holding Costs

Trading-off Transportation and Inventory Costs
When shippers compare costs of carriers and modes, it is important to account for all possible cost components. For example, transportation costs, handling costs, and inventory costs should be balanced.

Example
Hypothetical company has:

- 1 plant, producing 1 product
- 1 warehouse, storing the product from which customers supplied directly
- Plant keeps 0 safety stock inventory, warehouse maintains 100 units
- Average customer demand rate at warehouse: 10 units/day
- Plant produces 10 units/day

Would like to compare two modes of transportation for moving product from the plant to the warehouse.

Rail:

- Shipments in cars containing 400 units
- Cost: $3 /unit
- Transit time: 20 days

Truck:

- Truckloads of 100 units
- Cost: $7 /unit
- Transit time: 3 days

Handling costs: $60 per shipment regardless of mode

Inventory costs

- Value of items at plant: $500 / unit
- Value of items at warehouse: $505 / unit
- Inventory holding cost rate: 30% of value per year held

Which mode should be used?

Rail analysis

Inventory costs at origin/plant:
• Average inventory at plant = 400/2 = 200 units

• Annual inventory holding cost at plant = (avg inventory quantity) x (value/unit) x (annual holding cost rate) = 200 x 500 x 0.3 = $30,000

Inventory costs at warehouse/destination:

• Average inventory at warehouse = 100 + 400/2 = 300 units

• Annual inventory holding cost at WH = (avg inventory quantity) x (value/unit) x (annual holding cost rate) = 300 x 505 x 0.3 = $45,450

In-transit inventory costs:

• Each item spends a certain number of days in transit: an inventory cost

• Annual in-transit inventory cost: (annual demand) x (years in Transit) x (value/unit) x (annual holding cost rate) = 365 x 10 x 20/365 x 502.5 x 0.3 = $30,150

Transportation cost:

• Charged for each unit sent

• Annual cost: (annual demand) x (transportation cost/unit) = 365 x 10 x 3 = $10,950

Handling cost:

• Annual number of shipments: (annual demand) / (shipment size) = \(\frac{365 \times 10}{400}\) = 9.125

• Annual cost: (annual shipments) x (cost per shipment) = 9.125 x 60 = $547.50

Total Annual Cost: 30,000 + 45,450 + 30,150 + 10,950 + 547.50 = $117,097.50

*Truck analysis*

Inventory at origin/plant:

• Average inventory at plant = 100/2 = 50 units

• Annual inventory holding cost at plant = 50 x 500 x 0.3 = $7,500

Inventory at warehouse/destination:

• Average inventory at warehouse = 100 + 100/2 = 150 units
• Annual inventory holding cost at WH = 150 x 505 x 0.3 = $22,725

In-transit inventory:
  • Number of shipments per year: \( \frac{(365)(10)}{100} = 36.5 \) (total demand/shipment size)
  • annual in-transit inventory cost: 36.5 x 100 x 3/365 x 502.5 x 0.3 = $4,522.50

Transportation cost:
  • annual cost: 365 x 10 x 7 = $25,550

Handling cost:
  • annual cost: 36.5 x 60 = $2,190

Total Annual Cost: 7,500 + 22,725 + 4,522.50 + 25,550 + 2,190 = $62,487.50
So, trucking is actually much cheaper in this case. Important question: solve for the inventory holding cost rate for which trucking and rail would yield equivalent costs.
Logistics modeling

Shortest path models

Frequently, decision-support systems for transportation require methods for finding a \textit{minimum-cost path} between points. Points could be geographic locations, or more abstractly represent any type of point in time and/or space. “Costs” could be:

- distances
- times
- more general costs, represented in $\$

Shortest-path calculations are based on a \textit{network model}: a network or graph $G$ includes a set of points called nodes $N$, and a set of arcs connecting the points $A$: $G = (N, A)$. Arcs are often referenced as $(i, j)$; connecting node $i$ to node $j$. Arcs may have a number of characteristics, but for shortest paths must include a \textit{cost} characteristic: the cost $c_{ij}$ of arc $(i, j)$ is the cost for travel from $i$ to $j$.

Decision arcs usually have a direction, from a head node $i$ to a tail node $j$. If a network is specified without arc directions, assume that it is \textit{undirected}. In an undirected network, it is assumed that a connection between $i$ and $j$ implies a connection between $j$ and $i$ with identical characteristics.

It’s important to recognize the decision structure of a network model. Each node $i$ represents a “point” at which a decision is made, and each arc $(i, j)$ represents a decision that moves us from decision point $i$ to $j$.

\textit{Transportation application}: Given a network model representing the transportation system (roadways, railways, air traffic lanes) for a given mode, determine the minimum time (distance, cost) path from an origin point to a destination point.

\textit{Global supply chain application}: Given a network model for a shipper representing its full set of transportation contracts for a particular commodity, determine the minimum cost (time) path of transport options from an origin to a destination.

Minimum Cost Path Question: Find the minimum-cost paths from a given source node $s$ to all other nodes $j$, with the costs $v(j)$. 
Dijkstra’s algorithm

- Requires that $c_{ij} \geq 0$

**Initialization:**

$v(j) \leftarrow \begin{cases} 
0 & j = s \\
+\infty & \text{otherwise}
\end{cases}$

$pred(j) \leftarrow \begin{cases} 
0 & j = s \\
-1 & \text{otherwise}
\end{cases}$

$PERM = \emptyset; PERM = N$

**Iterations:**

while $PERM \neq \emptyset$:

1. Let $p \in PERM$ be node for which $v(p) = \min\{v(j), j \in PERM\}$
2. $PERM \leftarrow PERM \cup \{p\}$
3. $PERM \leftarrow PERM \setminus \{p\}$
4. For every arc $(p, j) \in A, j \in \overline{PERM}$ leading from node $p$:
   (a) if $v(p) + c_{pj} < v(j)$, then:
   - $v(j) \leftarrow v(p) + c_{pj}$
   - $pred(j) \leftarrow p$

**Pros**

1. Achieves fastest running times (theoretically and practically) for dense networks. Let $|N| = n$ and $|A| = m$. A network is dense if $m \rightarrow n^2$. Typical geographic networks are not dense, though.

2. Can be truncated. Each iteration, one node $p$ is given a permanent, optimal minimum cost label and predecessor. Thus, if you are interested only in paths to some subset of the nodes $T \subset N$, you can stop the algorithm once $PERM \supseteq T$.

**Cons**

1. Alternative methods are faster (practically) on sparse networks.

2. Efficient implementations require programming a priority queue (heap).
Bellman-Ford algorithm
Sometimes, label-correcting algorithm. No requirement of non-negative arc costs. Often more efficient in practice for sparse transportation problems.

Initialization:
\[
v(j) \leftarrow \begin{cases} 
0 & \text{if } j = s \\
+\infty & \text{otherwise} 
\end{cases}
\]
\[
pred(j) \leftarrow \begin{cases} 
0 & \text{if } j = s \\
-1 & \text{otherwise} 
\end{cases}
\]
\[
LIST = \{s\}
\]

Iterations:
while \(LIST \neq \emptyset\):

1. Remove node \(p\) from top of \(LIST\)

2. For every arc \((p, j)\) leading from node \(p\):
   (a) if \(v(p) + c_{pj} < v(j)\), then:
       • \(v(j) \leftarrow v(p) + c_{pj}\)
       • \(pred(j) \leftarrow p\)
       • if \(j \notin LIST\), \(LIST \leftarrow LIST + \{j\}\)

Pros

1. Achieves fastest running times (practically) for sparse networks, especially when implemented with Pape’s Modification.

2. Very simple to implement.

Cons

1. Labels and predecessors not optimal until full completion of algorithm.

2. Running time may be slow on dense networks due to frequent label updates.
Worst-case Computational Complexity

- A method to theoretically estimate the worst case running time (in steps) of an algorithm or heuristic.

- A method is said to have worst-case complexity of $O(f(x))$, where $x$ is a vector of inputs, if for values of $x$ sufficiently large the number of steps required is at most $cf(x)$ where $c$ is a fixed constant (does not depend on $x$).

- If $f(n,m)$ is a polynomial, a network method is said to be efficient. Examples for networks: $O(n)$, $O(m^4)$, $O(m + n)$. An inefficient, or exponential, algorithm heuristic does not have a polynomial $f(x)$: $O(e^n)$, $O(2^n)$, $O(n!)$.

Complexity of Dijkstra’s and Bellman Ford

- **Dijkstra’s**: Two primary operations. (1) Node selections: Must find the minimum temporarily-labeled node $n$ times, and must scan all the temp nodes each time. Since the number of temp nodes is reduced by one each iteration, this requires $n + (n - 1) + (n - 2) + ... + 1$ steps: $\frac{n(n+1)}{2}$. (2) Label updates: Scans each arc exactly once: $O(m)$ steps. Total complexity: $O(0.5n^2 + 0.5n + m)$, but since $n, m \leq n^2$: $O(n^2)$.

- **Bellman Ford**: It can be shown that each arc will be scanned (step 2) at most $n - 1$ times (we will not show this here, but it has to do with the optimality of paths of length less than or equal to $k$ after the $k$-th examination of each arc). Thus, the complexity is $O(mn)$.

- Clearly, Dijkstra’s has better worst-case complexity than B-F.

Dijkstra Improvements

The difficult step, computationally, in Dijkstra’s algorithm is the selection of the temporarily labeled node with minimum current cost label. To do so requires scanning each temporary node label. This operation can be improved in practice by storing the labels in some type of sorted fashion. A common way to do so is to use a partially-ordered heap data structure for the nodes in $\text{PERM}$. A heap is a data structure in which each element (parent) has a number of descendants (children); the heap has the property such that the label of the parent element is always less than or equal to that of its children. Thus, when labels are stored in a heap, you can identify the minimum cost label by picking the node off the top of the heap.

A binary heap is commonly used in Dijkstra’s implementations. In a binary heap, each parent element has exactly two children. The algorithm is initialized with $s$ on the top of the heap, and the remaining nodes placed anywhere in the heap since they all have equal label. The search step is replaced by removing the node from the top of the heap, and readjusting the heap. Next, whenever a node’s cost label is modified (decreased),
we need to move that node up the heap. This is simple; we simply exchange its position with its parent until its parent is less than.

The best known theoretical worst-case complexity of Dijkstra’s uses a Fibonacci heap data structure; $O(m + n \log n)$; in practice, this method is difficult to program and not worth the effort. The binary heap is $O(m \log n)$, which is theoretically worse than $O(n^2)$ but better for sparse networks, and better in practice.

**Bellman-Ford Improvements: Pape’s Modification**

Bellman-Ford works very well for sparse networks in practice, but works even better when modified with Pape’s Modification (also known as a dequeue modification). One way that B-F can get bogged down is in the handling of the LIST. Often, some node $p$ is pulled off the list early in the process and a number of nodes downstream from $p$ get label updates and are inserted into the LIST. If the label at $p$ is now modified again, it is inserted onto the end of the LIST. Unfortunately, there may be a number of nodes ahead of $p$ in the LIST whose current labels are the result of $p$. It would be better to “fix” those labels before modifying even more downstream labels. To do so, we can use Pape’s simple LIST management modification:

(iii) if $j \notin \text{LIST}$: if $j$ has never been in LIST, $\text{LIST} \leftarrow \text{LIST} + \{j\}$, else $\text{LIST} \leftarrow \{j\} + \text{LIST}$

**Intuition**

Let me now provide some intuition to why the Bellman-Ford algorithm correctly solves the shortest path problem. At each step of that algorithm, we pull a node out of the list $p$ and check all of the arcs leaving that node $(p, j)$ for violation of the optimality condition:

$$v(p) + c_{pj} < v(j), \text{ then } v(j) \leftarrow v(p) + c_{pj}$$

In words, this says “If we find a better path to node $j$ by traveling first to $p$ and then from $p$ to $j$, then update $j$.” Each time we update a node like $j$, it is added into the LIST so that we can then see if we need to update any nodes reachable from $j$. And so on.

**Optimality conditions**

Given $G = (N, A)$ and a start node $s$, a set of path cost labels $v(j)$ for all $j \in N$ represent an optimal set of minimum cost path labels from $s$ if and only if:

1. $v(s) = 0$

2. $v(j) \leq v(i) + c_{ij} \quad \forall \ (i, j) \in A$

The existence of polynomial conditions for optimality usually imply the existence of an efficient optimal algorithm. Optimality conditions are important to understand for sensitivity analysis. Given a set of labels, have you found the right ones? If some data changes, are the labels still optimal? Is there an easier update procedure?
Dijkstra’s algorithm: binary heap implementation

Heap implementation:

- Suppose for convenience that node $s$ is node 1
- A vector $h(j)$ contains the node in position $j$ in the heap; the variable $\text{heapsize}$ gives the current size of the heap.
- The node $h(1)$ is the one on the top of the heap.
- The heap is set up so that the children of $h(k)$ are $h(2k)$ and $h(2k+1)$; i.e., the children of $h(1)$ are $h(2)$ and $h(3)$, the children of $h(2)$ are $h(4)$ and $h(5)$, etc.
- The heap property is that $h(k) \leq h(2k)$ and $h(k) \leq h(2k+1)$ for all $k$. (note: if $2k$ or $2k+1$ is greater than $\text{heapsize}$, these comparisons aren’t considered).
- Functions $\text{moveup}$ and $\text{removetop}$ handle readjusting the heap.
- When all nodes permanent, $v^*$ gives the optimal cost labels.

Initialization:

\[
v(j) \left\{ \begin{array}{ll}
0 & j = 1 \\
+\infty & \text{otherwise}
\end{array} \right.
\]

\[
\text{pred}(j) \left\{ \begin{array}{ll}
0 & j = 1 \\
-1 & \text{otherwise}
\end{array} \right.
\]

$h(j) = j \quad \forall \ j \in N$

$\text{notperm} = n$

$\text{perm}(j) = 0 \quad \forall \ j \in N$

Iterations:

while $\text{notperm} > 0$:

1. Let $p \leftarrow h(1)$
2. $\text{perm}(p) \leftarrow 1$
3. $v^*(p) \leftarrow v(p)$
4. $\text{removetop}(v, h, n)$
5. $\text{notperm} \leftarrow \text{notperm} - 1$
6. For every arc $(p, j) \in A$, $\text{perm}(j) = 0$ leading from node $p$:

(a) if $v^*(p) + c_{pj} < v(j)$, then:
\[ v(j) \leftarrow v^*(p) + c_{pj} \]
\[ \text{pred}(j) \leftarrow p \]
\[ \text{moveup}(j, v, h) \]

The function \texttt{removetop} moves the top element in the heap to a bottom position by exchanging it with the smaller of its children (in terms of label) until it reaches the bottom of some branch.

\texttt{removetop}(v, h, heapsize)

1. \( k = 1 \);
2. \( t = h(1) \);
3. \( v(t) = M \) (let the temporary label be a large number \( M \))
4. while ( \( 2k \leq \text{heapsize} \))
   (a) \( j = 2k \)
   (b) if \( j < \text{heapsize} \) then if \( v(h(j)) > v(h(j + 1)) \) then \( j \leftarrow j + 1 \)
   (c) \( h(k) \leftarrow h(j) \)
   (d) \( k \leftarrow j \)
5. \( h(k) = t \)

The function \texttt{moveup} takes a current element in the heap which is now out of order because of a label update, and moves it up the heap by exchanging positions with parents until the heap is properly re-ordered.

\texttt{moveup}(j, v, h)

1. \( \textit{node}j = j \) (j is currently a node number; first find it in the heap)
2. \( \textit{cost}j = v(\textit{node}j) \)
3. \( j = \text{find}(\textit{node}j, h) \) (j is now node j's position in the heap)
4. \( k \leftarrow \lceil \frac{j}{2} \rceil \)
5. while ( \( k > 0 \) and \( v(h(k)) \geq \textit{cost}j \))
   (a) \( h(j) \leftarrow h(k) \)
   (b) \( j \leftarrow k \)
   (c) \( k \leftarrow \lfloor \frac{k}{2} \rceil \)
6. \( h(j) = \textit{node}j \)
Multi-label algorithm for shortest paths

In some problems, we wish to find paths that balance multiple objectives. For example, we might want to find the path with minimum distance that does not exceed a time duration constraint. Alternatively, we might want to find a minimum-time path that does not exceed a certain distance. For these types of scenarios, a multi-label algorithm similar to the Bellman-Ford algorithm can be used to determine all potential paths.

To model problems with two measures of cost, for example distance and time, nodes are given labels with two components, \((c_k, t_k)\), where \(c_k\) represents the total distance (or some other cost) required to reach the node from \(s\) and \(t_k\) the total time from \(s\) along some path.

In this problem, nodes may have multiple labels resulting from different paths. This may result since one path to a node may have longer distance but shorter time than another path; it is then up to the decision-maker to decide which path should be chosen.

Paths in the network are represented again using a pred variable. In this case, however, each label \((c_k, t_k)\) has a predecessor label, denoted \(\text{pred}(k)\). Since each label is associated with a node, tracing back from the “chosen” label at the destination node from label-to-label will give the node sequence of the path. An example is given at the end.

At each node \(j\), we maintain a set of non-dominated labels; these labels are at least as good as other labels at the node in terms of cost or time. Thus, a label \((c_q, t_q)\) is dominated by \((c_k, t_k)\) if both \(c_q > c_k\) and \(t_q > t_k\).

Finally, the \(LIST\) now contains labels, not nodes. At each step of the algorithm, we remove a label from the \(LIST\) and examine the arcs that leave from the node associated with that label.

\[\text{Initialization:}\]

Initially, the origin node \(s\) is given label \((0, 0)_1\). (The subscript indicates the reference number \(k = 1\) for this label) All other nodes receive no labels. The origin label \(\text{pred}\) is 0.

\[\text{LIST} = \{(0,0)_s\}\]

\[\text{Iterations:}\]

while \(\text{LIST} \neq \emptyset\):

1. Remove some label \((c_k, t_k)\) from \(\text{LIST}\) (its associated node is \(p\))

2. For every arc \((p, j)\) leading from node \(p\):

   (a) Define a new label \(r\) for node \(j\) as \((c_r, t_r) = (c_k + c_{pj}, t_k + t_{pj})\) and add to \(\text{LIST}\)

   (b) Define \(\text{pred}(r) = k\)

   (c) Compare new label to each existing label \((c_q, t_q)\) associated with node \(j\):

      i. If \(c_k + c_{pj} \geq c_q\) -and- \(t_k + t_{pj} \geq t_q\), then the new label is dominated by \((c_q, t_q)\). Discard the new label, and remove it from \(\text{LIST}\).
ii. Else if \( c_q \geq c_k + c_{pj} \) and \( t_q \geq t_k + t_{pj} \), then the new label dominates this label. Discard the label \((c_q, t_q)\), and if necessary remove it from LIST.

**Stopping criterion:**
When no labels remain in LIST, this indicates that no remaining label corrections can be found, and all of the best cost, time paths have been found.

**Concerns:**
In this algorithm, it appears that problems might be created when discarding a dominated label, since other labels “downstream” might have this label as a predecessor. Although these labels are now invalid, if you execute the algorithm correctly these “invalid” labels will eventually be discarded as dominated themselves. Can you see why?

**Using the results:**
Once you have executed the multi-labeling algorithm until the stopping criterion is met, that is when there are no labels left to process in the LIST, you have now generated a set of good (non-dominated) paths with respect to the two objectives, distance and time in this case.

**An example**
Suppose after running the algorithm on a 6-node example, the following labels and predecessors are determined for each node:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((0, 0))</td>
<td>((3, 4))</td>
<td>((9, 2))</td>
<td>((7, 5))</td>
<td>((5, 7))</td>
<td>((14, 9))</td>
</tr>
<tr>
<td></td>
<td>pred=0</td>
<td>pred=1</td>
<td>pred=1</td>
<td>pred=2</td>
<td>pred=2</td>
<td>pred=8</td>
</tr>
<tr>
<td></td>
<td>((8, 7))</td>
<td>((6, 8))</td>
<td>((8, 6))</td>
<td>((11, 10))</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pred=2</td>
<td>pred=6</td>
<td>pred=5</td>
<td>pred=6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>((7, 11))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pred=6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Choosing some paths**
In this example, we have now enumerated all non-dominated paths from node 1 to the other nodes. Suppose we are interested in paths to node 6. Since node 6 has two labels, we have two choices. We could pick the path with higher distance and shorter time (represented by label \((14, 9)\)), or the path with less distance but more required travel time ((11, 10)).

If we choose the higher distance path, we can determine the path by tracing through the predecessors for that label. First, label \((14, 9)\) has predecessor as label 8. Label 8, \((8, 6)\) is at node 5 and has a predecessor of label 5. Label 5, \((7, 5)\), is at node 4, and has predecessor of label 2. Label 2 \((3, 4)\) is at node 2, and has predecessor of label 1. Label 1 is at the start node, \(s = 1\). So the path yielding this (distance, time) combination is 1-2-4-5-6.

If we choose the lower distance path, we proceed as follows. First label \((11, 10)\) has predecessor label 6. Label 6 \((5, 7)\) is at node 5, and has predecessor of label 2. Label 2 is at node 2, with predecessor of label 1. Label 1 is the start node, \(s = 1\). So the path yielding this (distance, time) combination is 1-2-5-6.
Minimum spanning tree models
First, we start with some technical information. Recall the definition of a network model: $G = (N, A)$. Suppose there are $n$ nodes, and $m$ arcs.

Path
A path $P$ in $G$ is a sequence of nodes $\{i_1, i_2, ..., i_p\}$ such that $i_k \in N$, $i_k$ are all unique, and either $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$ for all $k = 1, 2, ..., p - 1$. A directed path requires that $(i_k, i_{k+1}) \in A$; clearly a directed path is a feasible sequence of decisions for a network user. The cost of a directed path is $\sum_{k=1}^{p-1} c_{i_k i_{k+1}}$. In our study of MCP problems, we identified methods for determining the minimum cost paths in $G$ for $i_1 = s$.

Cycle
A cycle is a set of arcs that forms a closed loop in $G$; you can begin walking at some node, follow only arcs in the cycle regardless of their direction, and then return to that node. Thus, a cycle $C$ is a sequence of nodes $\{i_1, i_2, ..., i_p, i_1\}$ where $i_k \in N$, $i_k$ are unique, and either $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$ for all $k = 1, 2, ..., p - 1$ and either $(i_p, i_1) \in A$ or $(i_1, i_p) \in A$. In an undirected network, a cycle clearly is a permitted sequence. In a directed network, it may or not be permitted. Examples.

Trees
A tree $T$ in $G$ is a collection of arcs such that all $n$ nodes are connected, and no cycles exist. (All nodes connected implies the existence of a path between all pairs of nodes on $T$.) An equivalent definition of a tree is a set of $n - 1$ arcs such that all nodes are connected. An important property of a tree is that it defines, for each pair of nodes, a unique (not necessarily directed) path between them.

Tree examples: Directed network, undirected network.

Minimum spanning tree
A minimum spanning tree $T^*$ in a network where each arc $(i, j)$ has a cost $c_{ij}$ is a tree such that the total cost (sum) of all the arcs in the tree is minimized:

$$T^* = \arg\min_{T \in G} \left\{ \sum_{(i,j) \in T} c_{ij} \right\}$$

Question: Is the minimum spanning tree the same as the shortest path tree?
Answer: No. In the shortest path tree, the length of the path from some origin node to each other node is minimized. In the minimum spanning tree, the total cost of all the arcs in the tree is minimized. This is not the same thing.

Finding the minimum spanning tree
Two algorithms that you can use. Both of them are quite fast, and easy to understand.
Kruskal’s algorithm

Initialization: $TREE = \emptyset$

Iterations:

1. Sort the arcs in order of non-decreasing $c_{ij}$ into the $LIST$
2. while $|TREE| < n - 1$:
   
   (a) Remove $(i, j)$ from the top of the $LIST$
   
   (b) Add $(i, j)$ to $TREE$ if so doing does not create a cycle

At the end, $T^* \leftarrow TREE$.

Cuts
To explain the next algorithm, it necessary to understand the concept of a cut in a network.

Suppose the nodes in $N$ are partitioned into two independent subsets, $S$ and $T$. That is, $S \cup T = N$ and $S \cap T = \emptyset$. The cut set $(S, T)$ is given by all arcs $(i, j)$ such that $i \in S$ and $j \in T$ or $i \in T$ and $j \in S$.

Prim’s algorithm

Initialization:
$TREE = \emptyset$

$S = \{1\}, T = N \setminus S$

(In English, the set $S$ contains node 1, while the set $T$ contains all nodes except node 1)

Iterations:

while $|TREE| < n - 1$:

1. Find the arc $(i, j)$ in $(S, T)$ with minimum cost $c_{ij}$
2. Add $(i, j)$ to $TREE$
3. Remove $j$ from $T$ and add $j$ to $S$

At the end, $T^* \leftarrow TREE$. 
Minimum spanning tree applications
Problems in designing networks where the fixed costs are more important than variable costs.

Mini-max paths
A standard minimum cost path from s to t is often called a minisum path; among all paths P connecting s to t, we know that \( P_s^* = \arg\min_P \sum_{(i,j) \in P} c_{ij} \). In some applications, we would prefer to identify a minimax path, defined as follows: \( P_M^* = \arg\min_P \max_{(i,j) \in P} c_{ij} \). In words, among all paths connecting s to t, \( P_M^* \) is one that encounters the smallest maximum cost. Example. Four nodes. Four arcs. Show how the minisum path and minimax path may differ. 1-4, 3-3.

There are some useful applications of these paths. One type of arc cost that one might wish to “minimize the maximum” is elevation change. Suppose that you are designing a railroad line through a mountainous region, and you want to determine the alignment. Since costs associated with traction power, and the wear and tear associated with braking, increase quickly with absolute elevation change, you may wish to choose an alignment that minimizes the maximum elevation change. In another application, suppose you are transporting extremely hazardous materials, such that any spill would cause great harm to surrounding populations. Since any spill would be disastrous, you might not wish to minimize the expected population exposed, but rather the maximum population exposed.

Solving all-pairs minimax path
Fortunately, one can solve the all-pairs minimax pathing problem on an undirected network \( G \) in which you identify the minimax paths connecting all node pairs by simply solving a MST problem. Suppose you have found \( T^* \). Then, the minimax path between any pair of nodes s and t is identified by the path on \( T^* \). We can justify this claim using the cut optimality conditions. Let \( P_M^* \) be the path connecting s and t on \( T^* \), and suppose \( (i,j) = \arg\max_{(k,l) \in P_M^*} c_{kl} \). Further, let \( (I,J) \) represent the cut created by removing \( (i,j) \) from \( T^* \). From cut optimality, we know that \( c_{kl} \geq c_{ij} \) for all \( (k,l) \in (I,J) \). Since any path from s to t must use at least one arc \( (k,l) \), it follows that \( P_M^* \) has the minimum maximum value.
Single Vehicle Routing: the Traveling Salesman Problem

Pathing problems are only the beginning of our interest in transportation systems modeling. In many problems, instead of simply finding an optimizing path from $s$ to $t$, we are instead interested in visiting a number of locations, say $1, 2, ..., p$, with minimum total cost; we want to determine the order in which these locations are visited, which is known as the problem of routing. When routing vehicles, we are often interested in moving a vehicle back to where it starts. Suppose 0 is some current vehicle location (like, a depot or a yard where vehicles are kept). Then, we often wish to find a route from 0 that visits $1, 2, ..., p$ and returns to 0 with minimum total cost (time, distance); let $V$ represent this set of locations (nodes). Such a visitation sequence is called a tour.

Complete, Undirected Network with $\Delta$-Inequality

Let's assume now that we have a complete undirected network $G$; that is, a direct arc exists connecting each node to every other node. (A complete undirected network with $n$ nodes always contains $\frac{n(n-1)}{2}$ arcs.) Furthermore, suppose that the network satisfies the $\Delta$-inequality.

Definition 1 ($\Delta$-inequality on Complete Networks)

A complete network $G = (N, A)$ satisfies the $\Delta$-inequality iff for all $i, j, k \in N$ the following relationship holds: $c_{ik} \leq c_{ij} + c_{jk}$.

As an aside, there is also a notion of a $\Delta$-inequality for networks that are not complete.

Definition 2 (Triangle Inequality on Arbitrary Networks)

A network $G = (N, A)$ satisfies the $\Delta$-inequality iff for all arcs $(i, k) \in A$ and intermediate nodes $j \in N$ the following relationship holds: $c_{ik} \leq C^*_{ij} + C^*_{jk}$, where $C^*_{ij}$ represents the shortest-path cost from node $i$ to node $j$.

A network satisfies this assumption if the directed arc from $i$ to $j$ always has no greater cost than a path through a different node, say $k$.

Generating Complete, Undirected Network with $\Delta$-Inequality

Often, in transportation problems, the underlying network $G' = (N', A')$ for a problem is not complete, nor does it satisfy the $\Delta$-inequality. For example, $G'$ may be a network of road segments. However, such a network may can be converted to a CUNDI through the use of a minimum cost path procedure. Suppose $V$ is the set of nodes of interest in a routing problem. We can construct a CUNDI $G = (V, A)$ where the arc set $A$ includes an arc from each node in $V$ to every other node in $V$, and the cost on arc $(i, j)$ is the cost of the minimum cost path from $(i, j)$ in $G'$. Clearly, in cases where the cost of the path from $i$ to $j$ differs from the cost from $j$ to $i$, this procedure will not work; however, this assumption is frequently valid. Also, there must exist some path between all nodes in $V$.

Traveling salesperson tours and subtours

Let's now define the traveling salesperson problem on some CUNDI $G = (N, A)$, where $|N| = n$, with arc costs $c_{ij}(= c_{ji})$. Suppose now that the set of nodes that we wish to
visit $V = N$. Of course, this assumption is not restrictive since we can always reduce our network to the set of nodes we are interested in, and the arcs connecting them.

**TSP tour**: a visitation order, or sequence, of all the nodes in $V$: $T = \{v_1, v_2, ..., v_n, v_1\}$, where $v_i \in N$ and unique.

**Subtour**: a sequence of nodes with a common start and end node, such that not all nodes in $N$ are visited: $\{v_1, v_2, ..., v_{n-q}, v_1\}$, $v_i \in N$ and unique, $q$ a positive integer $< n - 2$.

**Tour costs and optimal Traveling salesman tours**

**Tour cost**: the cost of a tour is simply the sum of all of the arc segments traversed in the tour: $C(T) = \sum_{i=1}^{n-1} c_{v_i v_{i+1}} + c_{v_nv_1}$.

**Optimal traveling salesman tour**: the traveling salesman (TSP) tour that requires the minimum total cost. $TSP^* = \text{argmin}_{T \in G} C(T)$.

The problem of determining the optimal TSP tour in a network is in general a very difficult problem, and much more difficult than finding a minimum cost path or tree on a network. It is in the problem type NP-hard, which essentially means that no one has discovered an efficient optimal algorithm to solve all instances of the problem. In fact, the TSP problem is perhaps the most studied of all operations research problems.

**An enumeration algorithm**

One inefficient way to determine an optimal TSP tour is to simply enumerate each possible tour, and pick the one with minimum total cost. How many tours in a network with $n$ nodes? $\frac{(n-1)!}{2}$. Why? From a start node, one can go to $(n-1)$ nodes next. Then, a choice of $n-2$ nodes, and so on. We divide by 2 since tours are the same forward and backwards. Show three node example. So, the enumerative method is an $O(n!)$ algorithm, optimal but not efficient.

Be quite confident that an inefficient algorithm will not be aided by improvements in computation speed for large problem instances. Consider the following chart:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n^2$</th>
<th>$n^3$</th>
<th>$2^n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>100</td>
<td>1000</td>
<td>1024</td>
<td>3,628,800</td>
</tr>
<tr>
<td>50</td>
<td>2500</td>
<td>125,000</td>
<td>1.13e+15</td>
<td>3.04e+64</td>
</tr>
<tr>
<td>100</td>
<td>10,000</td>
<td>1,000,000</td>
<td>1.27e+30</td>
<td>9.33e+157</td>
</tr>
</tbody>
</table>

Since there are only about 31.5 million seconds in a year, a computer that can perform 1 billion operations per second would still require $4.02e + 13$ years to complete $2^{100}$ operations and $2.96e + 141$ years to complete $100!$ operations.

**Heuristic algorithms**

A **heuristic or heuristic algorithm** provides feasible solution to an optimization problem, which may or may not be optimal. Good heuristics give solutions that are close to the optimal solution, and usually are efficient in terms of theoretical or practical running time. A desirable property of a heuristic is that the worst-case quality of the solution provided can be guaranteed. For example, some heuristics guarantee that the solution they identify is at worst $\alpha$-optimal; that is, if the optimal cost is $C^*$, the heuristic finds
a solution no greater than \( \alpha C^* \) (for minimization problems).

**Importance of Lower Bounds for Hard Minimization Problems**

When solving hard minimization problems like TSP problems, we usually do not know the cost of the optimal solution, let alone an efficient means to identify the solution. So, how do we know if a heuristic solution is good? Suppose that \( T^H \) is some feasible tour identified by a heuristic. Clearly, we know that \( C(T^H) \geq C(TSP^*) \). But how close is it? Often, we can compare our heuristic solution cost instead to a lower bound on the optimal cost. Suppose \( C(TSP^*) \) is a lower bound for the TSP cost for a given problem. Then, \( C(TSP^*) \geq C(TSP^*) \). If the bound is close to the optimal solution (tight), then comparing \( C(T^H) \) to \( C(TSP^*) \) is a good proxy. And, if a heuristic solution is \( \alpha \) percent higher than a lower bound, it is clearly no greater than \( \alpha \) higher than the true optimal cost.

Developing good lower bounds for hard minimization problems is often nearly as hard as developing the optimal solution. However, any reasonable lower bound is often a good start.

**A simple lower bound for the TSP**

Often, we generate lower bounds using a decomposition strategy. For example, suppose a feasible solution to a problem can be decomposed into two components. If these two components are optimized separately, the resulting solution may not be feasible for the original problem, but the resulting cost is a lower bound on the optimal cost of the original problem.

Consider any feasible solution to the TSP, that is a tour visiting all nodes in \( G \). Suppose \( TOUR \) is a set containing all of the arcs used in the tour; clearly \( |TOUR| = n \). If one arc \((i,j)\) from the tour is removed, the remaining structure is a \( TREE \) on \( G \); all nodes are connected and the \( TREE \) contains \( n - 1 \) arcs. In set terminology, we could write this decomposition as:

\[
TOUR = TREE + \{(i,j)\}
\]

The cost could be written as follows:

\[
C(TOUR) = C(TREE) + c_{ij}
\]

Now let’s minimize the decomposition. Clearly, the minimum cost tree in a network is given by the MST. And, in general, \((i,j)\) could be any arc in the network, but more specifically is some arc that is not in the \( TREE \). Therefore, a lower bound on the optimal TSP cost is the following:

\[
C(TSP^*) \geq C(MST^*) + c_{ij}^*
\]

where \( c_{ij}^* \) is minimum cost arc not in \( MST \).
Heuristics for the TSP

I. An $\alpha$-approximation heuristic: MST 2-approximation

Steps:

1. Generate a minimum spanning tree $\text{MST}^*$ of the nodes $N$.

2. Let $\text{TOUR}$ contain two copies of each arc in the minimum spanning tree, $\text{MST}^*$.

3. Generate a $\text{WALK}$ of nodes of $G$ using only arcs in $\text{TOUR}$; \{$v_1, v_2, ..., v_{2(n-1)}, v_1\}$. Note that $v_i$ are not unique.

4. Eliminate duplication: let $t(i) = 0$ for all $i \in N$; $\mathcal{T}^H = \emptyset$.

5. for $i = 1$ to $2(n - 1)$:
   
   (a) if $t(v_i) = 1$, then continue;
   
   (b) else $\mathcal{T}^H \leftarrow \mathcal{T}^H + v_i; t(v_i) = 1$.

6. $\mathcal{T}^H \leftarrow \mathcal{T}^H + v_1$.

First, recognize that $\text{WALK}$ does not fit with our earlier definition of a tour, since it may contain nodes that are visited more than once. So, we simply eliminate repeat visits in our $\text{WALK}$ to identify $\mathcal{T}^H$, sometimes referred to as an embedded tour.

Generating a walk of the nodes

Generating the walk of the nodes above is always possible, since the arcs in $\text{TOUR}$ form what is known as an Eulerian graph. Walks in such sets can be identified, using a simple recursive procedure. Pick an arbitrary node $v_1$ as a starting point, and create a walk of some (but possibly not all) of the nodes: $\text{WALK}_1 = \{v_1, v_2, v_3, v_2, v_1\}$. Now remove the arcs you just used from $\text{TOUR}$, and examine the remaining structure. You now may have additional disconnected subwalks that may need to be attached to nodes in the main walk. For example, perhaps you find a new subwalk attached to $v_3$: \{$v_3, v_4, v_5, v_4, v_3\}$. Again, remove these arcs from $\text{TOUR}$. Then, you update $\text{WALK}_2 = \{v_1, v_2, v_3, v_4, v_5, v_4, v_3, v_2, v_1\}$. Keep proceeding recursively in this fashion until no arcs are left in $\text{TOUR}$.

For example, in Figure 3 suppose a TSP tour is started at node 1. One example of a $\text{WALK}$ that could be generated would be $\text{WALK} = \{1, 2, 1, 3, 4, 3, 1\}$. This tour contains unnecessary return visits to nodes, and has a total cost of 12 = 2 * (3 + 2 + 1). Applying the shortcut procedure, a tour would begin at 1 and move to 2. The walk specifies a return to 1, so we look ahead to the next unvisited node, which is 3. The shortest path from 2 to 3 is along arc (2, 3), so we add this arc to the tour. $T$ is now $T = \{(1, 2), (2, 3)\}$. Next, $\text{WALK}$ tells us to visit node 4 and since it has not yet been visited, we move along the MST: $T = \{(1, 2), (2, 3), (3, 4)\}$. Next, the $\text{WALK}$ specifies a return to 3. Since it has already been visited, we move ahead to the next unvisited node.
Figure 3: The dark black lines represent the minimum spanning tree.

Since all nodes have been visited, we return to the start node 1 along the shortest path, which happens to be arc (4, 1). The final tour is $T = \{(1, 2), (2, 3), (3, 4), (4, 1)\}$, with a cost of 10.

**Proof of the 2-approximation**

The MST 2-approximation heuristic will generate a TSP tour with a cost no greater than double the optimal cost for networks satisfying the triangle inequality. This is not good, but at least it’s a guarantee!

**Proof:**

First, we know that the tour we generated has a cost less than twice the minimum spanning tree, by the $\Delta$-inequality:

$$C(T^H) \leq 2C(MST^*)$$

Now, suppose that you were to remove a single arc from the optimal traveling salesman tour; the result is a tree. Let’s call it $TSPTREE$. It’s clear that the minimum spanning tree cost is always no greater than the cost of any $TSPTREE$:

$$C(MST^*) \leq C(TSPTREE)$$

and of course, the cost of any $TSPTREE$ is no greater than the cost of the optimal TSP tour.

$$C(TSPTREE) \leq C(TSP^*)$$

The result now follows:

$$C(T^H) \leq 2C(MST^*) \leq 2C(TSPTREE) \leq 2C(TSP^*)$$
B. Christofides’ Heuristic

Christofides’ heuristic has the best known worst-case performance bound for the traveling salesman problem on complete networks satisfying the triangle-inequality. Similar to the Twice-Around MST heuristic, Christofides’ heuristic utilizes the concept of minimum spanning trees. In addition, the heuristic relies on the idea of minimum cost perfect matchings (MCPM). Consider the following definitions and facts:

**Definition 3 (Node Degree)**
The degree of a node \( i \in N \) with respect to a set of arcs \( T \) is the number of arcs in \( T \) that are adjacent, or touching, node \( i \).

**Lemma 1 (An Euler Lemma)**
Given a set of nodes \( N \) and a set of arcs representing a spanning tree \( T \), the number of nodes with odd degree with respect to the arc set \( T \) is even.

**Definition 4 (Perfect Node Matching)**
Given a set of nodes \( S \) such that \(|S|\) is even, a perfect node matching \( W \) is a set of \( \frac{|S|}{2} \) node pairings such that each node in \( S \) is in exactly one pairing. In a complete network, each node pairing \((i, j), i, j \in S\) represents an arc in the network. The cost of matching \( W \) is given by \( C(W) = \sum_{(i,j) \in W} c_{ij} \).

**Lemma 2 (Another Lemma deduced from Euler)**
Suppose given a complete network, you find a spanning tree \( T \). Then, you find a perfect node matching \( W \) of the nodes in \( N \) with odd degree with respect to \( T \). Now consider the set of arcs \( R = T \cup W \). Now, the nodes in \( N \) each have an even degree with respect to \( R \) and are all connected. Thus, a traveling salesman tour of the nodes can always be constructed using only the arcs of \( R \) (with no duplicates).

The above definitions and lemmas motivate Christofides’ heuristic. Here is a sketch of the method:

**Christofides’ Heuristic**

Steps:

1. Given an undirected, complete network \( G \) satisfying the \( \Delta \)-inequality, find a minimum spanning tree \( MST^* \) on \( G \).
2. Let \( S \) be the nodes of \( G \) with odd degree with respect to \( MST^* \).
3. Find a perfect node matching \( W^* \) of the nodes in \( S \) with minimum total cost. (This problem can be solved in \( O(n^3) \) time by the method of Lawler).
4. Generate a tour of the nodes \( T^C \) using arcs in the set \( MST^* \cup W^* \) exactly once. (\( T^C \) can be improved by introducing shortcuts to avoid revisiting nodes).
It remains to prove the worst-case performance bound for this heuristic.

**Theorem 0.2 (Christofides Worst-case Performance Bound)** Christofides heuristic is 1.5-optimal for every complete network \( G \) satisfying the \( \Delta \)-inequality, that is:

\[ C(T^C) \leq \frac{3}{2} C(TSP^*) \]  

where \( TSP^* \) is an optimal traveling salesman tour on \( G \).

**Proof:**

1. First, it is clear that \( C(T^C) \leq C(MST^*) + C(W^*) \), since \( T^C \) uses only arcs in the set \( MST^* \cup W^* \) and may include shortcuts.

2. It is additionally clear that \( C(MST^*) \leq C(TSP^*) \). (This was shown using earlier arguments: if you remove one arc from \( TSP^* \) you create a spanning tree, which has a cost necessarily no less than \( MST^* \), the minimum spanning tree.)

3. Thus, we now have shown from (1) and (2) that \( C(T^C) \leq C(TSP^*) + C(W^*) \).

4. We now claim that \( C(W^*) \leq \frac{1}{2} C(TSP^*) \). The proof is as follows:

   (a) Let \( t_1, t_2, \ldots, t_{2k} \), where \( k = \left\lfloor \frac{|S|}{2} \right\rfloor \), be the nodes in \( S \) listed in the order they would be visited by the optimal tour \( TSP^* \) on \( G \). Note that we need not know what this tour is for the proof. And further note that \( TSP^* \) visits all the nodes in \( N \), not just the nodes in \( S \). As a quick example, suppose we have 7 nodes and \( TSP^* = \{3, 4, 7, 2, 5, 1, 6, 3\} \). Suppose that \( S = \{2, 4, 6, 7\} \). Then, \( t_1 = 4, t_2 = 7, t_3 = 2, t_4 = 6 \).

   (b) Consider the following two feasible perfect matchings on \( S \):

   \[
   M_1 = \{(t_1, t_2), (t_3, t_4), \ldots, (t_{2k-1}, t_{2k})\} \quad \text{and} \quad \ M_2 = \{(t_2, t_3), (t_4, t_5), \ldots, (t_{2k}, t_1)\}.
   \]

   Note that the union of \( M_1 \) and \( M_2 \) describes a tour that visits only the nodes in \( S \). This tour is not necessarily optimal for the nodes in \( S \), but that does not matter.

   (c) First, it is clear that \( C(M_1) + C(M_2) \leq C(TSP^*) \). To see why, recognize that \( M_1 \cup M_2 \) is a tour that visits only the nodes in \( S \), but in the same order in which they would be visited by \( TSP^* \) which visits more nodes. Thus, \( M_1 \cup M_2 \) essentially includes some “shortcuts” that exclude visits to the nodes not in \( S \) (i.e., in \( N \setminus S \)). By the \( \Delta \)-inequality, therefore, the expression holds.

   (d) Second, it is also true that \( C(W^*) \leq \frac{1}{2}(C(M_1) + C(M_2)) \). This is well-known; the cost of an optimal solution to any minimization problem is always no greater than the average cost of two feasible solutions.

   (e) From (c) and (d), it follows that \( C(W^*) \leq \frac{1}{2}(C(M_1) + C(M_2)) \leq \frac{1}{2} C(TSP^*) \)

5. Thus, from (3) and (4) the result follows.
II. Construction Heuristics

Nearest-Neighbor heuristic
The nearest-neighbor heuristic is a greedy local search method for TSP which maintains at each step a current path. Starting at an arbitrary starting node, it scans “neighbors” of either one or both current path endpoints and adds the closest neighbor to the tour. The last step of the method connects the remaining two endpoints together.

Initialization:
Given an undirected network $G = (N, A)$, with costs $c_{ij}$ associated with each $(i, j) \in A$, where $c_{ij}$ satisfies the $\Delta$-inequality.

Alternative 1: $P = \{i, j\}$ where arc $(i, j)$ has minimum cost among all arcs
Alternative 2: Choose an arbitrary start node $i$, and find $j \in N \setminus \{i\}$ such that $c_{ij}$ is minimized. Let $P = \{i, j\}$.

Steps:
while $|P| < n$:

1. Let $s = \text{begin}(P)$ and $t = \text{end}(P)$, the nodes at the beginning and end of the current path.

2. Choose the node $k \in N \setminus P$ to enter the path:
   
   Alternative 1: Let $k$ be such that $c_{tk}$ is minimized. $P = P + \{k\}$.
   
   Alternative 2: Let $k$ be such that $c_{tk}$ or $c_{ks}$ is minimized. The new path $P$ is either $P + \{k\}$ or $\{k\} + P$ respectively.

end;

Let $T = P + \{\text{begin}(p)\}$.

Tour Generation: The nodes in $T$ now define a TSP-tour when followed in order.

This heuristic is simple to understand without resorting to precise pseudocode. At each step, the current tour $T$ contains two endpoints, that is, nodes connected to only one arc in the tour. At each step of the algorithm, the arc with minimum cost connecting an endpoint to a node not yet visited by the tour is added to the tour. If no unvisited nodes remain, a connection is formed between to the two endpoints to complete the tour.

Nearest insertion heuristic

Initialization:
Given an undirected network $G = (N, A)$, with costs $c_{ij}$ associated with each $(i, j) \in A$, where $c_{ij}$ satisfies the $\Delta$-inequality.

Find the arc $(i, j)$ with minimum cost in $A$. Let $T = \{i, j, i\}$

Steps:
while $|T| < n + 1$:

1. Find the node $j \notin T$ that is closest to a node currently in $T$. That is, find $j \notin T, i \in T$ such that $c_{ij}$ is minimized.

2. Find the insertion location for node $j$ into the existing tour. To do so, find the arc $(\ell, k)$ in the current tour with the minimum insertion cost: $c_{\ell j} + c_{jk} - c_{\ell k}$. Insert $j$ between $\ell$ and $k$ in $T$.

end;

The nearest-insertion heuristic is a $2 - \frac{2}{n}$-optimal heuristic.

**Farthest insertion heuristic**

*Initialization:*

Given an undirected network $G = (N, A)$, with costs $c_{ij}$ associated with each $(i, j) \in A$, where $c_{ij}$ satisfies the $\Delta$-inequality.

Find the arc $(i, j)$ with maximum cost in $A$. Let $T = \{i, j, i\}$

*Steps:*

while $|T| < n + 1$:

1. Let $C_j(T)$ be $\min_{i \in T} c_{ij}$ for all nodes $j \notin T$. The node $j$ to be inserted into the tour is one with maximum $C_j(T)$.

2. Find the insertion location for node $j$ into the existing tour. To do so, find the arc $(\ell, k)$ in the current tour with the minimum insertion cost: $c_{\ell j} + c_{jk} - c_{\ell k}$. Insert $j$ between $\ell$ and $k$ in $T$.

end;

In English, Step 1 finds the node $j$ not on the tour for which the minimum distance to a vertex on the tour is maximum, and adds that node to the current tour.

**Cheapest insertion heuristic**

*Initialization:*

Given an undirected network $G = (N, A)$, with costs $c_{ij}$ associated with each $(i, j) \in A$, where $c_{ij}$ satisfies the $\Delta$-inequality.

Alternative 1: Find the arc $(i, j)$ with maximum cost in $A$. Let $T = \{i, j, i\}$

Alternative 2: Use some procedure to develop an initial subtour, $T$. Frequently, convex hulls (or close approximations) are used.

*Steps:*

while $|T| < n + 1$:
1. Let $C_{jt}(T)$ be $c_{tj} + c_{jnext(t)} - c_{lnext(t)}$ for all nodes $j \notin T$.

2. Let $j^*$ and $\ell^*$ be the indices of the minimum cost $C_{jt}$. Insert node $j^*$ after node $\ell^*$ in $T$.

end;

In this heuristic, we simply calculate an insertion cost matrix at each step representing the cost of inserting node $j \notin T$ after node $\ell \in T$. We choose the minimum cost insertion at each iteration.
III. Improvement Heuristics

Given that a tour $T$ has already been created via some method, it may be possible to improve the tour with an improvement heuristic. One often employed improvement heuristic is called $\lambda$-opt. A $\lambda$-opt heuristic is based on the concept of $\lambda$-optimality, which is a local neighborhood condition. Essentially, a tour is said to $\lambda$-optimal if there is no way to replace $\lambda$ arcs in $T$ with $\lambda$ arcs not in $T$ to create a new feasible tour with lower cost:

**Definition 5 (\(\lambda\)-optimality)**

Given a CUNDI $G = (N, A)$ and a tour $T_i$, let $A(T_i)$ be the set of arcs defined by the nodes in $T_i$: $A(T_i) = \{(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n), (v_n, v_1)\}$. Then, tour $T_i$ is $\lambda$-optimal if and only if there does not exist two sets of arcs, $X \in A(T_i)$, $Y \in A \setminus A(T_i)$ such that:

1. $|X| = |Y| = \lambda$
2. $\sum_{(i,j) \in X} c_{ij} - \sum_{(i,j) \in Y} c_{ij} > 0$
3. The set $A(T_i) + Y \setminus X$ allows a new feasible TSP tour sequence $T_{i+1}$.

These ideas motivate the following generic heuristic. Note that technically, the $\Delta$-inequality is not necessary for the validity of this heuristic.

**\(\lambda\)-opt heuristic**

*Initialization:*

Given some initial traveling salesman tour $T^0$

$k = 0$

*Iterations:*

1. Identify a feasible $\lambda$-exchange with positive savings. That is, find two sets $X \in A(T_k)$ and $Y \in A \setminus A(T_k)$ such that $|X| = |Y| = \lambda$, $\sum_{(i,j) \in X} c_{ij} - \sum_{(i,j) \in Y} c_{ij} > 0$, and that the set of arcs $T_k + Y \setminus X$ admits a feasible tour sequence. If no sets can be identified, STOP. $T_k$ is $\lambda$-optimal.

2. Let $T^{k+1}$ be the new tour sequence generated from $X$ and $Y$.

3. Repeat from 1.

*Analysis:*

How many options for removing $\lambda$ arcs in $T_k$ and replacing them with $\lambda$ new arcs exist? In a network with $n$ nodes, each tour contains $n$ arcs. Thus, there are $\binom{n}{\lambda}$ ways of removing $\lambda$ arcs from a tour. If removing a set of $\lambda$ arcs results in exactly $x$ nodes that are no longer adjacent to any arc in the tour, the number of new ways to reconnect the nodes is $[(\lambda - 1)!2^{\lambda-1-x}] - 1$. 
The two most commonly employed heuristics are 2-opt and 3-opt. Let’s analyze how many new tours are created in each iteration in these cases.

2-opt: In this case, there are $\binom{n}{2}$ ways of removing 2 arcs. Additionally, in $n$ of these cases, one node is left no longer adjacent to any arc. In these $n$ cases, there is no new way to connect the nodes: $[(2-1)!2^{2-1-1}] - 1 = [1!2^0] - 1 = 0$. In the remaining cases, there is exactly one new way to reconnect the nodes: $[(2-1)!2^{2-1}] - 1 = [1!2^1] - 1 = 1$. Therefore, the total number of possible 2-opt exchanges is:

$$\left( \binom{n}{2} - n \right)$$

3-opt: In this case, there are $\binom{n}{3}$ ways of removing 3 arcs. In $n$ of these cases, two nodes are left no longer adjacent to any arc. In these $n$ cases, there is exactly one new way to connect the nodes: $[(3-1)!2^{3-1-2}] - 1 = [2!2^0] - 1 = 1$. In $n(n-4)$ cases, one node is left no longer adjacent to an arc. In these cases, there are $[(3-1)!2^{3-1-1}] - 1 = [2!2^1] - 1 = 3$ new ways to reconnect. In the remaining cases, there are $[(3-1)!2^{3-1}] - 1 = [2!2^2] - 1 = 7$ new ways to reconnect. Therefore, the total number of new tours to be examined at each iteration is:

$$7 \left( \binom{n}{3} - n - n(n-4) \right) + 3n(n-4) + n$$

Using these equations for a network 50 nodes, the 2-opt heuristic would examine at most 1175 exchanges each iteration while 3-opt would examine no more than 127,700 exchanges each iteration.

**Implementation Issues**

Implementing an efficient 2-opt or 3-opt procedure efficiently requires addressing a number of issues. Most implementations attempt to identify the set of removed arcs $X$ and added arcs $Y$ as follows. Consider first the case with 2-opt:

1. Choose a node $t_1$ arbitrarily in $N$. 

![Figure 4: The dark black lines represent the TSP tour. Note that in this case, $\binom{4}{2} = 6$, but 4 of these choices leave one node no longer adjacent to any arc in the tour. The removal of arcs (1, 2) and (1, 3) is such an example.](image-url)
2. Choose \( t_2 \) such that \( T \) includes \{\ldots, t_1, t_2, \ldots\} or \{\ldots, t_2, t_1, \ldots\}. Thus, (undirected) arc \( x_1 = (t_1, t_2) \).

3. Choose now an arc \( y_1 \) that will be added to the tour following the removal of \( x_1 \). It is clear that \( x_1 \) and \( y_1 \) must share an endpoint! So, suppose \( y_1 = (t_2, t_3) \): the choice required is to choose node \( t_3 \)!

4. Choose now another arc to leave the tour. There are two choices dictated by \( t_3 \), since the leaving arc must be adjacent! \( x_2 = (t_3, t_4) \) such that \( T \) contains \{\ldots, t_3, t_4, \ldots\} or \{\ldots, t_4, t_3, \ldots\}.

5. Finally, \( y_2 \) is given since the tour must be reconnected: \( y_2 = (t_4, t_1) \).

This method is used to identify a potential exchange with reduction of \( c_{y_1} - c_{x_1} + c_{y_2} - c_{x_2} \); many implementations will simply find an exchange with negative reduction, make the exchange and then search again.

The Lin-Kernighan heuristic

Since a blend of different \( \lambda \)-opt exchanges is likely to lead to the most improvement to a TSP tour, a systematic method for performing variable \( \lambda \)-opt exchanges would likely be an effective practical heuristic. This is the idea behind the LK heuristic, which is widely recognized as the best-performing practical heuristic for solving TSP problems. For the small (fewer than 50 node) geographic network problems that we tend to find in transportation and logistics, most implementations of LK will actually find optimal solutions! In fact, in the most recent DIMACS TSP Challenge conducted in 2000, a publicly-available LK code solved every random 1,000 node instance to within 0.2\% of optimality (verified with an optimal code) in no longer than 20 seconds on standard personal computers.

In LK, we consider sequential exchanges where we identify a pair of arcs at each step that we will swap; by definition, they must have an endpoint in common. We will choose at least two exchanges such that the sequence of exchanges defines a new feasible tour, and the reduction of each exchange improves the tour. Here’s how the generic form works.

Define the reduction \( r_i = c_{y_i} - c_{x_i} \), and let \( R_i = \sum_{k=1}^{i} r_k \):

**Initialization:**
Given some initial tour \( T_0 \), let \( T \leftarrow T_0 \)

**Iterations:**

1. Let \( i = 1 \). Choose \( t_1 \).
2. Choose \( x_1 = (t_1, t_2) \in T \).
3. Choose \( y_1 = (t_2, t_3) \notin T \) such that \( R_1 < 0 \). If no such exists, go to Step 11
4. Let \( i = i + 1 \)
5. Choose \( x_i = (t_{2i-1}, t_{2i}) \in T \) such that:
(a) If \( y_i = (t_{2i}, t_1) \) is added, a feasible tour \( T_N \) results.
(b) \( x_i \neq y_s \) for all \( s < i \).

If \( R_k < 0 \), then \( T_N \) is a better tour than \( T \). Let \( T \leftarrow T_N \) and go to Step 1.

6. Choose \( y_i = (t_{2i}, t_{2i+1}) \notin T \) such that:
   (a) \( R_i < 0 \)
   (b) \( y_i \neq x_s \) for all \( s < i \).
   (c) \( x_{i+1} \) exists; that is, there is a possible feasible choice for the next arc to leave.

   If such \( y_i \) exists, go to Step 4

7. If there is an untried alternative for \( y_2 \), let \( i = 2 \) and go to Step 6.

8. If there is an untried alternative for \( x_2 \), let \( i = 2 \) and go to Step 5.

9. If there is an untried alternative for \( y_1 \), let \( i = 1 \) and go to Step 3.

10. If there is an untried alternative for \( x_1 \), let \( i = 1 \) and go to Step 2.

11. If there is an untried alternative for \( t_1 \), go to Step 1.
The Assignment Problem

In the assignment problem, we desire to determine the optimal way to assign resources to tasks. Since each element of an assignment problem is either a resource or a task, we can represent the problem using a bipartite network as shown in the figure:

Bipartite network: A network $G = (N, A)$ in which $N$ is partitioned into two distinct sets of nodes, $R$ and $T$, such that each arc connects a node in $R$ with a node in $T$. Formally, $N = R \cup T$, $R \cap T = \emptyset$, and if $(i, j) \in A$ then either $i \in R$ and $j \in T$ or $i \in T$ and $j \in R$.

Figure 5: A bipartite assignment network. The left-hand nodes in circles represent resources, and the triangles represent tasks.

To formulate an assignment problem, one must identify a set of resources and a set of tasks. Additionally, one must determine the cost $c_{ij}$ for assigning resource $i$ to task $j$. Typically in a problem, there will be $m$ resources and $n$ tasks where $m \geq n$.

Assignment Problem Formulation (all assignments feasible)

The simplest form of the assignment problem assumes that any resource can be assigned to any task. In such problems, the bipartite assignment network would have arcs connecting each resource to each task. In this case, the binary integer programming formulation of the problem is given as follows:

Let $x_{ij}$ be the binary decision variable for this problem. $x_{ij}$ will be set equal to 1 if resource $i$ is assigned to task $j$, and 0 otherwise:
\[ x_{ij} = \begin{cases} 1 & \text{if resource } i \text{ is assigned to task } j \\ 0 & \text{otherwise} \end{cases} \]

\[
\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \quad \text{(OF)}
\]

subject to:

\[
\sum_{i=1}^{m} x_{ij} = 1 \quad \forall \ j = 1 \ldots n \quad \text{(C1)}
\]

\[
\sum_{j=1}^{n} x_{ij} \leq 1 \quad \forall \ i = 1 \ldots m \quad \text{(C2)}
\]

\[ x_{ij} \in \{0, 1\} \quad \forall \ (i, j) \quad \text{(C3)} \]

It turns out that when solving an assignment problem formulation, the special structure of the constraints make strict enforcement of (C3) unnecessary. In fact, if this formulation is solved using a linear programming solver (such as the one inside of Visual Xpress), the variables \( x_{ij} \) will naturally become 0’s or 1’s. Thus, (C3) above can be replaced by the following:

\[
x_{ij} \geq 0 \quad \forall \ (i, j) \quad \text{(C3a)}
\]

**Assignment Problem Formulation (only some assignments feasible)**

Usually, assignment problems are used to model scenarios where some resource-to-task assignments are not possible. For example, a resource may not be available to serve a certain task because of timing issues. Further, a resource may not be qualified to serve a certain task. For example, if a resource were a small truck, it may not be able to move a large load of freight. In these cases, the formulation of the assignment problem is slightly different.

Suppose that \( A_i \) is the set of all tasks that resource \( i \) can serve feasibly, and \( B_j \) is the set of all resources that can feasibly serve task \( j \). Once you have drawn a bipartite network depiction of the assignment problem, it is easy to identify these sets. With these definitions, the assignment problem can be formulated as follows:
\[
\min \sum_{i=1}^{m} \sum_{j \in A_i} c_{ij}x_{ij} \quad \text{(OF)}
\]

subject to:
\[
\sum_{i \in B_j} x_{ij} = 1 \quad \forall \ j = 1...n \quad \text{(C1)}
\]
\[
\sum_{j \in A_i} x_{ij} \leq 1 \quad \forall \ i = 1...m \quad \text{(C2)}
\]
\[
x_{ij} \geq 0 \quad \forall \ (i, j) \quad \text{(C3a)}
\]

**Solving Assignment Models**
Looking at the assignment problem formulations, it should be clear that these problems are linear programs that can be solved with any linear programming software, such as Visual XPress. It can also be shown that the assignment problem model is a special case of a minimum cost network flow model, and can alternatively be solved using a specialized MCNF algorithm such as network simplex or a flow augmentation algorithm. Finally, special algorithms exist to solve assignment problems explicitly, and one such algorithm is known as the Hungarian algorithm.
Application: Transportation Task-Time Problems (Tanker Scheduling)
General Matchings
Bin Packing

The bin-packing problem is another important problem in operations research with applications in logistics and transportation system operations.

**Problem definition:**
Suppose we have a supply of bins, and each bin has a capacity to hold $Q$ units. Furthermore, suppose we have a number of items that need to be packed into bins. Each item $i$ has a size $q_i \in (0, Q]$ which represents the portion of a bin’s capacity that this item requires. Items cannot be “split” between multiple bins.

Suppose for explanatory purposes the item sizes have been arranged into a list, $L = \{q_1, q_2, q_3, \ldots, q_n\}$.

Problem BPP: Assign each item to a bin such that:

1. The total weight of all items in a bin is no greater than $Q$ (no bin violates capacity)
2. The smallest number of bins is used

**Integer Programming Formulation**
Consider now the following mathematical programming formulation for the bin-packing problem. Here, we will assume that $n$ bins are available for packing, since in the worst-case, each item would require its own bin.

*Decision variables*

$$y_j = \begin{cases} 1, & \text{if bin } j \text{ is used} \\ 0, & \text{otherwise} \end{cases} \quad \forall j = 1\ldots n$$

$$x_{ij} = \begin{cases} 1, & \text{if item } i \text{ is assigned to bin } j \\ 0, & \text{otherwise} \end{cases} \quad \forall i = 1\ldots n, \ j = 1\ldots n$$

(BPP)

$$\min \sum_{j=1}^{n} y_j \quad \text{(OF)}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1\ldots n \quad \text{(C1)}$$

$$\sum_{i=1}^{n} q_i x_{ij} \leq Q \quad \forall j = 1\ldots n \quad \text{(C2)}$$

$$x_{ij} \leq y_j \quad \forall i = 1\ldots n, \ j = 1\ldots n \quad \text{(C3)}$$

$$0 \leq x_{ij} \leq 1 \quad \forall (i, j) \quad \text{(C4)}$$

$$x_{ij} \text{ integer} \quad \forall (i, j) \quad \text{(C5)}$$
The minimization nature of this formulation attempts to set \( y_j = 1 \) for as few bins as possible, and the objective function counts the number of bins opened. Constraints (C1) ensure that each item is assigned to some bin, and (C2) require that no bin exceeds its capacity \( Q \). Constraints (C3) require that closed bins \( y_j = 0 \) can have no items assigned to them.

Like the traveling salesman problem, no efficient algorithm has been developed for the bin-packing problem. If one wishes to solve the problem optimally, it is necessary to enumerate all possible ways of assigning items to bins, and this process is inefficient.

**Applications in Logistics**

The primary application of bin-packing in logistics is in the loading of freight. Suppose that you have \( n \) pieces of freight to load with various sizes, and you wish to use the fewest containers as possible to transport the freight. This becomes a bin-packing problem. This situation also arises when designing vehicle routing tours. One potential problem in vehicle routing is to determine the minimum number of transportation vehicles required to serve a set of customer demands, under the assumption that customer demands cannot be split between vehicles.

For example, suppose that you operate a fleet of vehicles each with capacity of 20 tons. Further, suppose that you need to serve 13 customers on a given day, and the following is their demands in tons: \( L = \{16, 10, 14, 12, 4, 8, 3, 12, 7, 14, 9, 6, 3\} \). How many trucks are required? Which trucks should serve which demands?

**Online heuristics for BPP**

First, a general definition. An online heuristic is one in which input data are received over time, and irrevocable decisions are made based on known data within some time limit. Online heuristics have no foresight; they cannot anticipate future data that has not yet “arrived.” Often, real-world problems require online heuristics. For example, if you are operating a truck fleet and planning which trucks will serve which demands, the demands from customers may arrive dynamically and you may not have anticipatory knowledge of what the next customer demand will be.

In the following online heuristics for BPP, let’s assume that customer lot sizes become known one at a time. Upon the arrival of a lot, an irrevocable assignment of the lot to a bin is made. Such heuristics have practical significance in the loading of transport containers, since changing a decision of where to stow a particular piece of cargo would require unloading it (and potentially other lots) and reloading it into a different container. Suppose that customer lots arrive one at a time, in the order specified by \( L \). Let’s develop some heuristics.

*Next-Fit Heuristic*

Description: This heuristic places the next item in the currently open bin. If it does not fit, the bin is closed and a new bin is opened.
**Initialization:**

Given a list of item weights $L = \{w_1, w_2, ..., w_n\}$.

Place item 1 in bin 1 and remove from $L$. Let $i = 1$, $j = 2$.

**Iterations:**

1. If item $j$ fits in bin $i$, place $j$ in $i$. If not, open a new bin $i + 1$ and place $j$ in bin $i + 1$. Let $i = i + 1$.
2. Remove item $j$ from $L$. Let $j = j + 1$.
3. While items remain in $L$, repeat from Step 1.

The next-fit heuristic has a special advantage over other online heuristics in this case, since it allows immediate dispatch. In the heuristic, whenever a container is closed it will not receive any future items and therefore can be immediately dispatched. This might mean a truck can leave a depot, or a container can be loaded onto a ship or train, or something similar.

**First-Fit Heuristic**

Description: This heuristic keeps all unfull bins open. It places the next item in the lowest-numbered bin in which the item fits. If it does not fit in any bin, a new bin is opened.

**Initialization:**

Given a list of item weights $L = \{w_1, w_2, ..., w_n\}$.

Place item 1 in bin 1 and remove from $L$. Let $j = 2$, $m = 1$.

**Iterations:**

1. Find the lowest numbered bin $i$ in which item $j$ fits, and place $j$ in $i$. If $j$ does not fit in any bin, open a new bin and number it $m + 1$ and let $m = m + 1$.
2. Remove item $j$ from $L$. Let $j = j + 1$.
3. While items remain in $L$, repeat from Step 1.

**Best-Fit Heuristic**

Description: This heuristic attempts to create the fullest bins possible. Again, all unfull bins are kept open. It places the next item $j$ in the bin whose current contents are largest, but do not exceed $1 - w_j$ (so the item fits). If it does not fit in any bin, a new bin is opened.

**Initialization:**
Given a list of item weights \( L = \{w_1, w_2, ..., w_n\} \).

Place item 1 in bin 1 and remove from \( L \). Let \( j = 2, m = 1 \).

**Iterations:**

1. Find the bin \( i \) whose contents are maximum but not greater than \( 1 - w_j \) (if \( S_i \) are the items in bin \( i \), \( \sum_{k \in S_i} w_k \) is the contents of bin \( i \)), and place \( j \) in \( i \). If \( j \) does not fit in any bin, open a new bin and number it \( m + 1 \) and let \( m = m + 1 \).

2. Remove item \( j \) from \( L \). Let \( j = j + 1 \).

3. While items remain in \( L \), repeat from Step 1.

**Performance guarantees**

The next-fit heuristic is 2-optimal. It has been shown that the first-fit and best-fit heuristics are 1.7-optimal; that is, for any input item sizes \( L \) the number of bins that they will generate is no worse than 1.7 times the optimal number of bins.

**Practical issues with first-fit and best-fit**

Both of these online heuristics lack the dispatch advantage of the next-fit heuristic. Of course, if a container is completely filled at some stage of the processing it can be immediately dispatched. However, in the pure form of these heuristics, bins that are open but not yet full cannot be dispatched. Thus, to use these methods in practical settings it will be necessary to augment the methods with dispatch rules. For example, a container that has reached some minimum fill percentage \( \beta \) and/or has been open for some minimum time \( \gamma \) might be dispatched.

**Offline heuristics for BPP**

Offline heuristics assume that you have access to all the item sizes when you start the heuristics. There are three important offline heuristics for bin-packing that all work by first sorting the items by non-increasing size. They are called respectively **next-fit decreasing**, **first-fit decreasing**, and **best-fit decreasing**.

A good example string of numbers is 6,6,5,5,5,4,4,4,4,2,2,2,2,3,3,7,7,5,5,8,8,4,4,5.

Sorted becomes 8,8,7,7,6,6,5,5,5,5,5,5,4,4,4,4,4,4,4,3,3,2,2,2,2,2.

**Performance guarantees**

It has been shown that the first-fit and best-fit decreasing heuristics perform better (usually) than the online versions. They are both 1.5-optimal.
Vehicle Routing

Definition
Suppose a fleet of vehicles is based at a depot (referred to as node 0). Each vehicle in the fleet can transport a maximum of $Q$ units (weight, volume, pallets, items, etc.). Suppose further that a set of $n$ customers (nodes 1 through $n$) are dispersed in a region near the depot. Each customer $i$ demands that a quantity $q_i$ (measured in the same units as $Q$) be picked up or delivered, and that this quantity may not be split between vehicles. Travel between locations $i$ and $j$ incurs some cost $c_{ij}$. Then, the vehicle routing problem (VRP) is to find a set of vehicle tours each beginning and ending at the depot, such that each customer’s demand is fully served, no vehicle violates its capacity, and the total travel cost of all vehicles combined is minimized.

A “combination” of TSP and Bin Packing
The standard vehicle routing problem can be thought of as a combination of the bin-packing problem and the traveling salesman problem. First, it should be clear that if trucks were very large, for example so large that a single truck could carry the quantities demanded by all customers (e.g. $Q = \infty$), the VRP reduces to a traveling salesman problem. The solution to the VRP would simply be a TSP tour through all customers and the depot. Thus, since the TSP is a special case of the VRP, the VRP is at least as hard to solve as the TSP.

The bin packing problem is also a special case of the VRP. To see why this is true, suppose that the costs $c_{ij}$ on all arcs between customer locations were zero, and the costs on arcs between customers and the depot had a cost of $\frac{1}{2}$. In this formulation, the optimal solution to the VRP uses as few vehicles (bins) as possible in order to minimize the amount of travel in and out of the depot, and the solution to the problem is the minimum number of bins required. Of course, this is exactly the objective of the bin packing problem.
Heuristics for the VRP

Since VRP is at least as hard as TSP and bin packing, no one to date has been able to develop an efficient optimal algorithm for the VRP. Therefore, we will again focus our attention on heuristic algorithms that lead to reasonably good solutions. In general, however, the heuristics that have been developed for VRP do not result in solutions of comparable quality to the TSP heuristics.

I. Clarke-Wright Savings Heuristic

The Clarke-Wright method begins with a poor initial solution in which each vehicle visits a single customer. Tours are then combined as long as total cost savings are produced. Suppose the depot is node 0.

Generic Clarke-Wright

1. For each node $i \in N$, construct a tour $T_i = \{i\}$ beginning at the depot, traveling to node $i$, and returning to the depot. Let $S$ be the set of tours.

2. For each pair of tours $T_i$ and $T_j$ in $S$ such that the total customer demands of the two tours combined does not exceed $Q$ (i.e., $\sum_{k \in T_i \cup T_j} q_k \leq Q$), compute the savings: $S_{ij} = c_{end(i)0} + c_{0begin(j)} - c_{end(i)begin(j)}$.

3. Choose the pair of tours $i$ and $j$ that maximize $S_{ij}$. If $S_{ij} \geq 0$, then add $T_i + T_j$ to $S$ and remove $T_i$ and $T_j$ from $S$. Else, end the heuristic.

4. If no tours were joined, end the heuristic. Else, repeat from Step 2.

Implemented Clarke-Wright Heuristic

You may note in the preceding presentation that savings need not be calculated more than once for each pair of nodes. Consider the following version of the heuristic:

1. For each node $i \in N$, construct a tour $T_i = \{i\}$ beginning at the depot, traveling to node $i$, and returning to the depot. Let $S$ be the set of tours.

2. Calculate the savings $S_{ij} = c_{0i} + c_{j0} - c_{ij}$ for all pairs of nodes $i, j$.

3. Sort the savings in nonincreasing order.

4. Walk down the savings combination list, and find the next feasible combination pair $(i, j)$ such that:

   (a) $i$ and $j$ are on different tours, and $i$ is the last node on its tour $T_k$ and $j$ is the first node on its tour $T_l$.

   (b) $\sum_{k \in T_i \cup T_j} q_k \leq Q$.

   (c) $S_{ij} \geq 0$.
Add the combined tour $T_i + T_j$ to $S$, and remove $T_i$ and $T_j$. Repeat this step as long as feasible combinations remain in the savings list.

II. Route-first Cluster-second Heuristics

Heuristics in this class begin first by creating an ordering of the nodes through some procedure, and then create feasible customer clusters to be served by a vehicle using the fixed ordering. After clusters are created, most of these heuristics simply serve customers following the initial order. However, others (like the Sweep heuristic) perform yet another step in which optimal or near-optimal TSP tours are created for each customer cluster and the depot. Technically, the Sweep Heuristic might be called an Order-first Cluster-second Route-third heuristic.

Generic TSP Partitioning Heuristic

Steps:

1. Let $T$ be an optimal or near-optimal TSP tour that visits all of the customers and the depot. Suppose wlog $T = \{0, i_1, i_2, ..., i_n, 0\}$.
2. Define the node ordering $T_O = \{i_1, i_2, ..., i_n\}$.
3. Create customer clusters by running the next-fit bin packing heuristic with lot sizes $q_i$ and vehicle capacity $Q$ on the customer order defined by $T_O$.
4. Suppose the generated clusters are $\{i_1, i_2, ..., i_k\}$, $\{i_{k+1}, i_{k+2}, ..., i_{\ell}\}$, $\{i_{\ell+1}, i_{\ell+2}, ..., \}$. The final VRP tours are then given as $\{0, i_1, i_2, ..., i_k, 0\}$ etc. where the order of the customers in the VRP tours is identical to the order in the original TSP tour $T$.

Optimal TSP Partitioning Heuristic

This method does not lead to an optimal VRP solution, but it is in some sense the best partitioning heuristic one could develop for a fixed order. It is due to Beasley (1983).

Steps:

1. Let $T$ be an optimal or near-optimal TSP tour that visits all of the customers and the depot. Suppose wlog $T = \{0, i_1, i_2, ..., i_n, 0\}$.
2. Define the node ordering $T_O = \{i_1, i_2, ..., i_n\}$.
3. Define a set of costs between pairs of nodes $0 \leq j < k \leq n$ as follows:

$$D_{jk} = \begin{cases} 
  c_{0,j+1} + \sum_{t=j+1}^{k-1} c_{it_i,t_{i+1}} + c_{i_k,0} & \sum_{t=j+1}^{k} q_{i_t} \leq Q \\
  \infty & \text{otherwise} 
\end{cases}$$

(2)
4. Find the shortest path using $D_{jk}$ from depot 0 to $i_n$, and use the path to define the optimal partition and the resulting tours. For example, if the shortest path is $0 \rightarrow i_u \rightarrow i_t \rightarrow i_n$, then the 3 tours defined are $\{0, i_1, ..., i_u, 0\}$, $\{0, i_{u+1}, i_{u+2}, ..., i_t, 0\}$, and $\{i_{t+1}, i_{t+2}, ..., i_n, 0\}$.

**Worst-case Performance of Optimal Partitioning**

Suppose you run Optimal Partitioning using an initial TSP tour produced by a $\alpha$-optimal heuristic:

1. In the special case where $q_i = 1$ for all customers (in other words, all customers have the same demand; this can be a useful problem when the vehicle capacity constraint dictates only the number of customers that can be visited by a single tour), the worst-case bound is $1 + \left(1 - \frac{1}{Q}\right) \alpha$.

2. In the general, the worst-case bound is $2 + \left(1 - \frac{2}{Q}\right) \alpha$.

**Order-first Cluster-second Route-third Heuristic**

*Sweep Heuristic*

In the sweep heuristic, nodes must have geographic positions. Without loss of generality, suppose that the depot is given cartesian coordinates $(0,0)$, and suppose each customer $i$ has polar coordinates $(r_i, \theta_i)$ relative to the depot. Here, suppose $0 \leq \theta_i < 2\pi$ is measured clockwise from the positive $x$-axis.

1. Choose a start ray $r$, with a polar angle $\theta$.

2. For each customer, calculate $\Delta_i$ as the relative angle between $r$ and the customer location, measured clockwise.

3. Sort the customers in order of non-decreasing $\Delta_i$. Suppose the order created is $T_O = \{i_1, i_2, ..., i_n\}$.

4. Create customer clusters by running the next-fit bin packing heuristic with lot sizes $q_i$ and vehicle capacity $Q$ on the customer order defined by $T_O$.

5. Create final VRP tours by generating a separate TSP tour for each customer cluster and the depot.

**Cluster-first Route-second Heuristics**

Often, in practical vehicle routing problems, the subproblem of determining which customers to group together in a cluster served by a single vehicle is often more important than routing considerations. Since effective clustering can lead to reduced fleet sizes through better packing, problems in which dispatch or fixed vehicle costs are high may include packing as a primary concern. However, it can be shown that by focusing on clustering before routing even in the traditional vehicle routing problem focused on distance-based costs can result in good heuristic solution methods.
A mathematical-programming based heuristic in this class using the generalized assignment model was proposed by Fisher and Jaikumar [1981]. This model relies on the concept of *seed* locations for vehicles, and then creates a cluster of customers for each vehicle by solving an integer program for assigning customers to seed locations. To ensure that vehicle capacities are not violated, generalized assignment constraints are used. To create “good” customer clusters, the cost of assigning a customer to a seed is developed to account for the approximate cost of adding that customer to the vehicle tour represented by the seed.

**Generalized Assignment Heuristic for VRP**

Suppose that each customer \( i = 1 \ldots n \) requires service of an unsplittable demand of size \( q_i \) and additionally is located at \( x_i \in \mathbb{R}^2 \), where \( x_i = (a_i, b_i) \). Further, suppose wlog that the depot is labeled node 0 and is located at \((0, 0)\). Suppose that the travel costs \( c_{ij} \) are given, \( i = 0 \ldots n, j = 0 \ldots n, j \neq i \). Finally, assume that no more than \( m \) vehicles can be used in any feasible solution.

**Steps**

1. Choose a number of vehicles to use in the solution, \( m \); the following expression provides bounds for \( m \):

   \[
   \left\lfloor \frac{\sum_{i=1}^{n} q_i}{Q} \right\rfloor \leq m \leq \overline{m}
   \]

2. Locate the \( m \) seeds geographically in the service region, such that seed \( j \) is at position \( y_j \in \mathbb{R}^2 \). Determine the cost \( c_{ij} \) for travel from each node \( i = 0 \ldots n \) to each seed \( j = 1 \ldots m \). For example, this cost might be proportional to the Euclidean distance between the points: \( c_{ij} = \gamma \| x_i - y_j \| \)

3. Talk about how to determine the assignment costs, \( C_{ij} \) from customer \( i \) to seed location \( j \)

**Decision variables**

\[
x_{ij} = \begin{cases} 1, & \text{if customer } i \text{ is assigned to seed } j \\ 0, & \text{otherwise} \end{cases} \quad \forall i = 1 \ldots n \,, \ j = 1 \ldots m
\]
\[(GAVRP)\]

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij}x_{ij} \quad \text{(OF)}
\]

subject to:

\[
\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i = 1 \ldots n \quad \text{(C1)}
\]

\[
\sum_{i=1}^{n} q_i x_{ij} \leq Q \quad \forall j = 1 \ldots m \quad \text{(C2)}
\]

\[
0 \leq x_{ij} \leq 1 \quad \forall (i,j) \quad \text{(C3)}
\]

\[
x_{ij} \text{ integer} \quad \forall (i,j) \quad \text{(C4)}
\]

**Decision variables**

\[
y_j = \begin{cases} 
1, & \text{if seed } j \text{ is used} \\
0, & \text{otherwise}
\end{cases} \quad \forall j = 1 \ldots n
\]

\[
x_{ij} = \begin{cases} 
1, & \text{if customer } i \text{ is assigned to seed } j \\
0, & \text{otherwise}
\end{cases} \quad \forall i = 1 \ldots n, \ j = 1 \ldots n
\]

\[
z_{kj} = \begin{cases} 
1, & \text{if vehicle } k \text{ is assigned to seed } j \\
0, & \text{otherwise}
\end{cases} \quad \forall k = 1 \ldots m, \ j = 1 \ldots n
\]
\( \text{(LBH)} \)

\[
\begin{align*}
\min & \sum_{j=1}^{n} v_j y_j + \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_{ij} & \text{(OF)} \\
\text{subject to:} & \\
\sum_{j=1}^{n} x_{ij} &= 1 \quad \forall \ i = 1 \ldots n & \text{(C1)} \\
\sum_{j=1}^{n} z_{kj} &\leq 1 \quad \forall \ k = 1 \ldots m & \text{(C2)} \\
\sum_{i=1}^{n} q_i x_{ij} &\leq \sum_{k=1}^{m} z_{kj} Q_k \quad \forall \ j = 1 \ldots n & \text{(C3)} \\
x_{ij} &\leq y_j \quad \forall \ i = 1 \ldots n, \ j = 1 \ldots n & \text{(C4)} \\
0 &\leq x_{ij} \leq 1 \quad \forall \ (i,j) & \text{(C5)} \\
x_{ij} &\text{ integer} \quad \forall \ (i,j) & \text{(C6)} \\
0 &\leq z_{kj} \leq 1 \quad \forall \ (k,j) & \text{(C7)} \\
z_{kj} &\text{ integer} \quad \forall \ (k,j) & \text{(C8)}
\end{align*}
\]

Asymptotic Analysis of the VRP

Why does it make sense to focus on clustering before routing? One motivation for this idea can be found through asymptotic analysis of the standard vehicle routing problem. In this type of analysis, we focus on the behavior of the optimal solution of random vehicle routing problems as the size of the problem (measured in number of customers) grows large. What has been shown is that in the case of vehicle routing, the asymptotic optimal solution value is related to the asymptotic optimal solution value of the bin packing problem using the customer demands as item sizes.

**Theorem 0.3 (Bin Packing Asymptotic Solution Value)** Suppose that the customer lot sizes are random in the sense that each lot \( q_i, i = 1 \ldots n \) is drawn independently from some probability distribution \( \Phi \) on \([0, Q] \). If \( b_n^* \) is the optimal number of bins needed to pack all the lots \( q_i \), then:

\[
\lim_{n \to \infty} \frac{b_n^*}{n} = \gamma
\]

**Theorem 0.4 (Vehicle Routing Asymptotic Solution Value)** Suppose that the customer lot sizes are random in the sense that each lot \( q_i, i = 1 \ldots n \) is drawn independently from some probability distribution \( \Phi \) on \([0, Q] \). Further, suppose that the depot is located at \((0, 0)\), customer locations \( x_i \) are a sequence of customer locations drawn independently from some distribution \( \Psi \) on \( \mathbb{R}^2 \), and distances between points are given by the Euclidean metric. If \( Z_n^* \) is the optimal solution to the VRP through these \( n \) customers then:

\[
\lim_{n \to \infty} \frac{Z_n^*}{n} = 2E(d) \gamma
\]
where $E(d) = \int_{\mathbb{R}^2} ||x||d\Psi(x)$

This theorem can be “interpreted” as follows: the total distance required, per point, in the optimal solution to large VRPs is equal to the number of vehicles required per point (given by the optimal bin packing constant) multiplied by an “average” distance to and from the depot. This is logical!

**Theorem 0.5 (Asymptotic Optimality of the Location-based Heuristic)** Under the assumptions of the previous theorem, there are versions of the location-based heuristic which return solution values $Z_{\text{LBH}}^n$ such that:

\[
\lim_{n \to \infty} \frac{Z_{\text{LBH}}^n}{n} = 2E(d)\gamma
\]  

(5)

A Location Model for Clustering (CFLP)

**Decision variables**

- $y_j = \begin{cases} 1, & \text{if a vehicle seed is located at customer } j \\ 0, & \text{otherwise} \end{cases}$

- $x_{ij} = \begin{cases} 1, & \text{if customer } i \text{ is assigned to the vehicle at seed } j \\ 0, & \text{otherwise} \end{cases}$

**Location-based Heuristic**

1. Create customer clusters for each vehicle by solving a CFLP formulation.
2. Choose a start ray $r$, with a polar angle $\theta$.
3. For each customer, calculate $\Delta_i$ as the relative angle between $r$ and the customer location, measured clockwise.
Exact Solution of TSP and VRP Problems via Branch-and-Cut

As we have discussed, both the TSP and VRP problems are in general very difficult combinatorial optimization problems for which no known polynomial-time (efficient) exact algorithm is known. Of course, however, there is an inefficient exact algorithm for these problems: explicit enumeration. In the case of the TSP on a complete undirected graph $G$ with $n$ nodes, full enumeration of all possible tours would require calculating the cost of $(n-1)!$ tours, a large number. The objective of branch-and-cut methods is somewhat of a compromise: smart partial enumeration. Using the concepts of problem relaxations, these methods attempt to enumerate only parts of the solution, and only if necessary.

An integer programming formulation for TSP
First, we will develop an integer programming formulation for the traveling salesman problem. Note that this formulation requires as input a complete network $G=(N,A)$ with costs $c_{ij}$ required on each arc. (Note that we do not explicitly require a symmetric network here.)
The TSP formulation developed is essentially a modified assignment problem. Every feasible TSP tour in a set of nodes $N$ has an assignment-like property: each node $i$ must be “assigned” to exactly one other node $j$ which will follow $i$ in the TSP tour. Thus, in our formulation, we will use binary decision variables $x_{ij}$ for all pairs of nodes $i$ and $j$. In the solution, $x_{ij}=1$ indicates that node $j$ immediately follows node $i$ in the TSP tour, while $x_{ij}=0$ if $j$ does not immediately follow $i$.
While this assignment property is necessary for a TSP tour, it is not sufficient to define a complete TSP solution since it allows subtours of the nodes in $N$. Recall that a subtour in $N$ is a cycle that does not include all nodes. Thus, we must add constraints to a simple assignment problem formulation of the TSP to eliminate these subtours.

Subtour elimination constraints
Consider some subset of the nodes $S \subset N$, and suppose the subset has 2 or more members ($|S| \geq 2$). In order to eliminate the possibility of a subtour on the nodes $S$, we can restrict the total number of arcs connecting members of the subset. Recall that given a set of $p$ nodes, a tree on the nodes will contain exactly $p-1$ arcs; if more arcs are used, a cycle (or subtour!) must exist. To prevent this, we simply restrict the allowable connections in a set $S$ as follows:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1$$

To see how such a constraint works, consider the example in the figure. In this case, we have a subset of nodes $S = \{1, 2, 3, 4\}$. The subtour elimination constraint for this case would be:

$$x_{12} + x_{13} + x_{14} + x_{21} + x_{23} + x_{24} + x_{31} + x_{32} + x_{33} + x_{41} + x_{42} + x_{43} \leq 3$$
Consider now the cases shown in the figure. In case (A), suppose $x_{12} = 1$, $x_{24} = 1$, $x_{43} = 1$, and $x_{31} = 1$. Thus, the subtour elimination constraint is violated since $1 + 0 + 0 + 0 + 1 + 1 + 0 + 0 + 0 + 1 = 4 > 3$. Case (B) and (C) are both acceptable; in case (B), the sum of arcs connecting members of the subset is 3 and in case (C) it is 2. (Note also that in all three cases, assignment-type constraints are satisfied.)

![Diagram](A) (B) (C)

Figure 6: Infeasible and feasible arc selections within a node subset

To create a correct formulation, it is necessary to introduce subtour elimination constraints for every strict subset of $S \subset N$ where $2 \leq |S| \leq n - 1$. The resulting formulation is given as follows:

**TSP(F)**

$$
\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \quad \text{(OF)}
$$

subject to:

$$
\sum_{j=1}^{n} x_{ij} = 1 \quad \forall \ i = 1 \ldots n \quad \text{(C1)}
$$

$$
\sum_{i=1}^{n} x_{ij} = 1 \quad \forall \ j = 1 \ldots n \quad \text{(C2)}
$$

$$
0 \leq x_{ij} \leq 1 \quad \forall (i, j) \quad \text{(C3)}
$$

$$
\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset N, \ 2 \leq |S| \leq n - 1 \quad \text{(C4)}
$$

$$
x_{ij} \text{ integer} \quad \forall (i, j) \quad \text{(C5)}
$$

Unfortunately, the formulation TSP(F) is difficult to solve for two reasons. First, there are an exponential number of constraints $(2^n - n - 1)$ of type (C4) so it is not possible to even efficiently generate or store the constraints as $n$ grows large. Secondly, constraints of type (C4) disrupt the unimodularity of the original assignment constraints and therefore require explicit constraints of type (C5) in order to ensure integer solutions.
A simple branch-and-cut method for TSP
Fortunately, we can overcome the obstacles presented by constraints (C4) and (C5) by solving TSP(F) iteratively using a procedure known as branch-and-cut. For those familiar with integer programming, branch-and-cut is similar to branch-and-bound in that branching is used to create a partially-enumerated search tree. The difference is that in addition to branching, constraints of type (C4) (subtour elimination constraints) will be introduced to the problem as needed when violations are detected.

Some basics of branch-and-bound are now explained; these ideas hold also in branch-and-cut procedures. Branch-and-bound for linear integer programming operates by successively solving a number of relaxations of some original hard integer programming problem; these relaxed problems are simply linear programs which are solvable efficiently in practice via the Simplex method. In the original branch-and-bound method, whenever an integer solution results from solving a linear relaxation, this solution is a feasible candidate for the optimal solution of the original problem. At all times, we maintain a current “best integer candidate” solution to the original problem, and then try to either (a) find a better solution, or (b) prove that no better solution is possible. These ideas are also a part of branch-and-cut. However, in branch-and-cut methods integer solutions (when found) may not immediately be feasible solution candidates to the original problem, since they may violate other constraints that we have ignored in the relaxation.

Often, the solution of a relaxed problem in a branch-and-bound (or cut) procedure will result in a solution that is non-integer. In this case, branching is necessary. Branching is the process of creating two new “children” problems from one “parent” problem; the two “children” are restricted versions of the parent. Branching accomplishes the following: the feasible set of the parent problem is partitioned into three subsets, one that is considered by one child problem, one that is considered by the other child problem, and a third that is no longer considered. Importantly, it is valid to ignore this third subset since it is infeasible for the original problem. In the case of binary integer programming (which is what we are concerned with here), branching on a fractional solution is a simple task: we identify a variable, say \(x_{kl}\), that is non-integer (e.g. \(x_{kl} = 0.5\)) in the optimal solution to the parent problem. The two children problems created include all of the constraints of the parent; child problem 1 additionally includes constraint \(x_{kl} = 0\) and child problem 2 includes \(x_{kl} = 1\) (clearly, the “ignored” part of the parent problem is the part of the feasible solution space where \(0 < x_{kl} < 1\)).

Solving a problem via branch-and-cut requires successively creating child problems, and solving them. The procedure ends when all child problems have been solved, or pruned. If we think of the structure created by iteratively generating child problems, it could be graphically depicted as a binary heap, or binary tree. Pruning halts the creation of child problems from a parent, for one of three reasons: (1) the parent problem is infeasible (pruning by infeasibility), (2) the parent problem’s optimal solution is a feasible, integer solution to the original problem (pruning by optimality), or (3) the optimal objective function value to the parent problem is greater than or equal to (for minimization problems) the current best integer, feasible objective function value (pruning by bound).
A flow chart for the branch-and-cut procedure for solving the TSP problem is given in Figure 7. The initial relaxed problem is denoted TSP(R), which is TSP(F) after removing constraints (C4) and (C5). Note that the problem of searching for and generating a subtour elimination constraint is a simple one; we need to identify a subset of nodes $S$ containing a subtour in the solution, which can be accomplished by an elementary search procedure beginning at an arbitrary node.

**An integer programming formulation for VRP**
(Laporte, Mercure, and Nobert; 1986)

Now, let’s extend the exact TSP formulation to the problem of the VRP. The formulation that we will develop does not put any special requirements on the cost matrix $c_{ij}$ for the costs between all customers and the depot. First consider a related problem known as the $m$-TSP.
Initialization: HEAP empty
Put initial problem TSP(R) on HEAP
z(u) = Infinity
X(c) = NULL

STOP
X(c) is optimal solution

Is HEAP empty?

Yes

No

Remove some problem P(i) from HEAP

Solve LP P(i)
Optimal objective z(i)
X(i) is LP solution

Prune by infeasibility

P(i) infeasible?

Yes

No

Prune by bound

z(i) >= z(u)?

Yes

No

X(i) all binary?

Yes

Search for a set of nodes S containing a subtour

No

Add subtour elimination constraint for S to P(i)

Prune by optimality

New solution:
z(u) = z(i)
X(c) = X(i)

Choose a variable X_a(i) that is non-integer.
Create two new problems, P(a) and P(b) using all constraints of P(i). In P(a), add constraint x_a=0 and in P(b) add x_a=1.
Add P(a) and P(b) to HEAP

Figure 7: Branch-and-cut for TSP