Decision support systems for logistics

DECISION SUPPORT SYSTEMS FOR LOGISTICS

by

K. M. van Hee

DECISION SUPPORT SYSTEMS FOR LOGISTICS

by

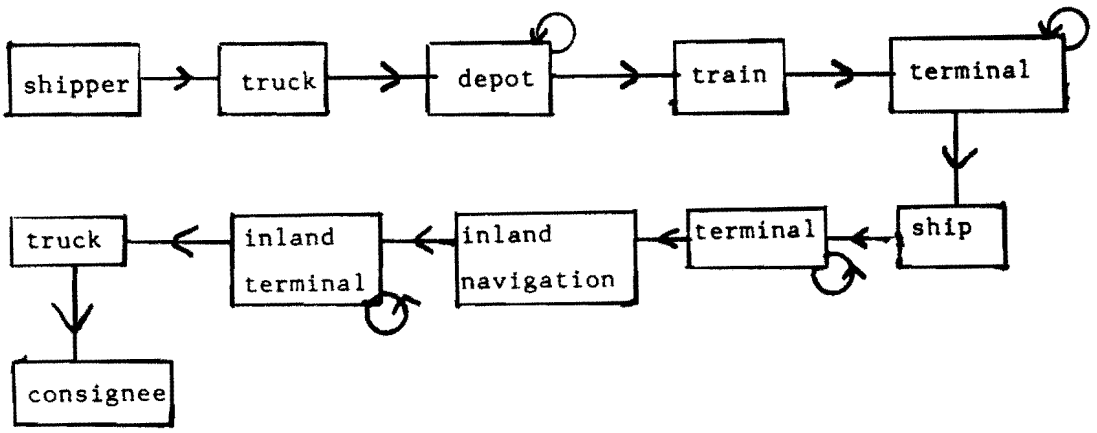
K. M. van Hee

CONTENTS

1. Introduction to logistic systems

In each economical system goods are moved from one place to another, sometimes with a break in a depot. These movements are triggered by a process of supply and demand. We call the physical movements, their monitoring and control: logistic processes. In each factory we find such processes but also between companies there are physical movements, sometimes over long distances. We call the logistic processes within companies internal logistic processes and between companies external logistic processes. The latter processes are often executed by specialized transport companies. We will emphasize in this paper the external logistic processes and therefore we will omit the adjective "external".

There are several ways to describe logistic processes. First we consider them from the point of view of the shipments. A shipment is a sequence of transport actions on a lot of goods from its origin to its destination. We call it an atomic shipment if the lot may be considered as one parcel during the whole transport. In practise a shipment may consist of several different atomic shipments. To describe the transport actions we use a graph. Each node represents a state in the transport process of a shipment and each arc represents a state-transition. The graph may have cycles. We illustrate this for an atomic shipment:



The cycles in depot and terminal indicate that the lot may be removed. There may be more ship and terminal states of the lot is unloaded and reloaded during the voyage. In the example we only considered physical transport actions, however there may also be non-physical actions like transfer of ownership of the lot. In principle each arc is triggered by some information and is producing some information. The triggering information is the control information and the produced information is an update of the status of the lot.

The cargo may be of different types:

- Bulk cargo, which may be divided into dry bulk like grains, ore or coal; liquids like oil and chemicals, and gases. Bulk cargo is not packed.
- Break bulk cargo, which may be grouped by packing-form: bags, boxes, drums, pallets or specialities like cars.
- Containers, which are packed in standard boxes, sometimes on a chassis.

The equipment for external transports are trucks, trains, airplanes, deep sea vessels, coasters and vessels for inland navigation. For short distances on termimals and in depots one uses cranes, forklift trucks and special equipment like straddle carriers and conveyer belts.

The activities of equipment may also be described by state-transition graphs. The performances of these types of equipment are very different. They are designed for efficient performing their own tasks, however due to their differences in spead and loading capacities, they cause queueing. Therefore transshipment points fulfil the role of buffers. The arrivals of means of transport at transshipment points may be considered very often as a pure random (Poisson) process. This causes large fluctuations in the need for equipment, personnel and storage capacities.

Sometimes the destination of a lot is not known at the moment of departure. Then the lot will be stored somewhere in the transport chain until the owner or a new owner determines a destination. Inventory control is therefore one of the subprocesses in logistic processes.

If we consider logistic systems as a network of transport companies then we distinguish several flows. The main flow is the cargo flow. Connected to the flow we see a financial flow of payments for services of the transport companies as well as sales transactions. Another flow we may distinguish is the transfer of responsibility for the cargo. We will be interested in the information flow. As said above, each change of status is coupled with some information. The transport companies may be divided into the following categories: shippers, shipagents, forwarders, shipowners, stevedores, wharfingers, brokers, trucking companies, depot-keepers, airline operators etc. The shipagents, forwarders and brokers are only dealing with information.

Other organisations involved in the logistic processes are customs, banks, insurance companies and trade companies. They exchange information between each other. Besides the physical activities in logistic processes there is an operations management, which includes the control of the physical activities as well as the maintenance of transport equipment and information flows, and the tactical and strategic management. The first group includes:

- Selection of the route of a lot by the shipper. (he takes into account the cost, the time and the reliability)
- Clustering of lots over carriers (ships, trucks etc.) by forwarders and shipagents.
- Arranging of lots in a carrier or in a buffer (shed, yard etc.) by shipowner, trucker etc. or terminal operator.
- Simultaneous moves of homogeneous objects, for instance empty containers or bulk cargo to match geographically distributed supply and demand. This is done by shipowners but also by shippers and brokers.
- Routing of carriers that visit several locations to bring or fetch lots to a depot.

The last group includes the planning of capacities such as

- size and composition of fleets of carriers
- volume and location of sheds, yards and berths
- manpower and terminal equipment

and policies such as

- route schemes
- reorder schedules for inventories.

For all of these managerial processes information from the physical processes are needed.

In section 2 we give a survey of the theory of information systems. In section 3 we sketch some examples of information systems for logistic processes. In section 4 we consider some combinatorial models for logistic decision problems. Section 5 is devoted to design theory for decision support systems. Finally in section 6 we consider a case of an inland navigation company where we illustrate the theory of section 6.

2. Survey of information systems theory

Information systems are developed to play a role in some other system, sometimes called object system.

We first describe formally what a system is.

A system can be defined by a 4-tuple:

$$<S, A, R, P>$$

where

S: state space, set of possible states of the system

A: action space, set of possible actions excercised on the system from its environment

R: response space, set of possible reactions of the system to its environment

P: transition mechanism, the law of motion, according to which the system walks through its state space.

We consider only discrete systems, which means that we assume that the number of state transitions in a finite time interval is finite. In the most general case the transition mechanism of a discrete system is given by a conditional probability:

$$P(S_{n+1} \in B_1 \wedge R_{n+1} \in B_2 \wedge T_{n+1} \in B_3 \mid S_n = s \wedge A_n = a \wedge T_n = t)$$

where

- S_n is the state of the system after the n-th transition
- A_n is the action after the n-th transition
- T_n is the time at which the n-th transition occurred.

The function P expresses the probability the system state belongs to set B_1, the respons to set B_2 and the transition time to set B_3 after the (n+1)-th transition given the state, action and transition time after the n-th transition. Note that a deterministic system is a special case where for some $\tilde{s}$, r, $\tilde{t}$

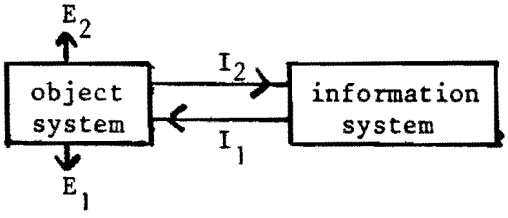
$$P(S_{n+1} = \tilde{s} \wedge R_{n+1} = r \wedge T_{n+1} = \tilde{t} \mid S_n = s \wedge A_n = a \wedge T_n = t) = 1$$

Hence for these systems we usually define the transition mechanism by a function $P : S \times A \times \mathbb{R} \to S \times R \times \mathbb{R}$ such that $P(s, a, t) = \langle \tilde{s}, r, \tilde{t} \rangle$. Often we are not interested in the transition times and then we may omit T_n resp. in the expressions above.

The walk through the state space of the system according to the excercised actions and transition mechanisme is called the process of the system.

Such a description of a system as a (probabilistic) automaton is only conceptually of interest, because in practise we cannot describe the 4 elements of a system.

Both the object system and its information system can be defined by such 4-tuples.



The object system: $\langle S_1, A_1, R_1, P_1 \rangle$

with $A_1 = E_1 \times I_1$ and $R = E_2 \times I_2$ and

the information system: $\langle S_2, A_2, R_2, P_2 \rangle$

with $A_2 = I_2$ and $R_2 = I_1$. Hence the interaction between both systems is given by the two sets I_1 and I_2 .

Note that E_1 is the set impulses form the "outside world" of the object system and E_2 its reactions to the outside world. Therefore the object system communicates both with an environment and its information system, which is expressed by the product forms for A_1 and R_1 .

An information system can be described from several view points. One way is to describe its tasks for the object system, i.e. a description of the functions it fulfils for the object system. Such a view of a system is called a functional or external specification. Another way is to describe the

internal structure of the information system, i.e. its architecture. We start with the first approach.

The functions an information system fulfils for its object system are

1. registration of relevant aspects of state transitions of its object system
2. control of the state transitions of its object system.

The registered information is stored into a database. This is the kernel of the information system. The database may be considered as a variable. A value of the variable at some moment is called a database state. The set of all possible values of the variable is called the database state space. In practise it is impossible to define this set by enumeration and therefore we define it intensionally by a database schema. There are several ways to define databases. A famous one is called the relational data model. A slightly different approach is given by so called semantic data models to which class the entity-relationship model belongs (cf. 5.3).

The information system must record relevant aspects of the states of the object system. Therefore the database state space contains a "projection" of the state space of the object system, beside other information. By the design of an information system one of the main decisions is the choice of the relevant aspects of the states of the object system. For an existing object system there are already measurements defined, so there are "natural" sources of information. It is important to get the information as pure as possible, i.e. without aggregation or other non bijective transformations. From the natural sources one may try to extend the registration. A criterion is the amount of effort it costs to register the additional information.

A complementary approach is to inventory the needs of information to control the object system. It is clear that only registration of aspects is useful if this information is used.

Very often human beings take care of a large part of the control of the object system so they are part of the information system. The way they use information

to take decisions is often very difficult to find out and therefore designers of information systems try to define the aspects to register from the object system by means of interviewing personnel that belong or will belong to the information system. A better method is to design a model for the control of the object system and derive from this model the needed aspects of the state (transitions) of the object system. In practise one will use both methods.

An other aspect of the registration-function of the information system is the way the information of the object system is obtained and manipulated. In principle there are two ways to get information from the object system:

1. after each transition a complete new state (projection) is send to the information system
2. after each transition only the changes in the state are send to the information system.

The first approach, which may be compared to making a movie of the process of the object system is sometimes used in situations where the (projected) state space of the object system is rather small, for instance where the object system is a machine. The information system is sampling the states of the object system usually at equidistant moments. This sampling is independent of the state transitions. This approach is in principle rather simple, however the amount of information to store may grow very rapidly and to such an extent that there is a need to an aggregation method to update some fixed set of variables that can be kept in the database and the source information must be deleted. One may consider this as a queue: the latest sample of the state space of the object system is the head of the queue and the oldest sample the tail. The queue has finite capacity and when it is full it will store a new sample after it has deleted the oldest one. The aggregate variables must be updated before the oldest sample is deleted.

The second approach is the usual one in large information systems where the

object system is an organisation, i.e. a company, part of a company or group of companies. Of course this approach is more efficiently, both in transmission and storage of information. In principle there are three ways to register this data in the information system:

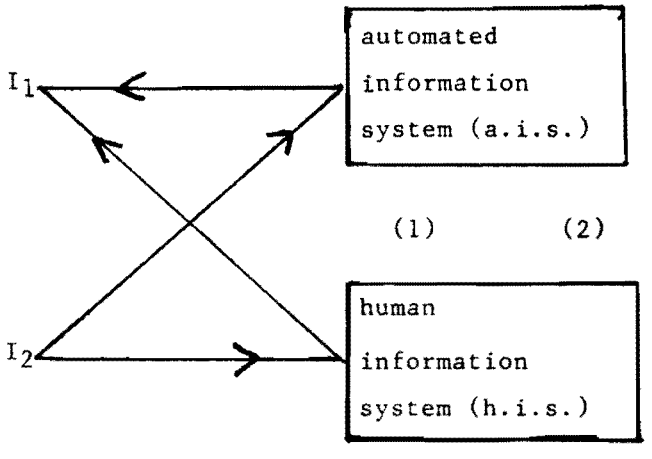
1. Update the old image of the state of the object system, i.e. update the database state. Now only the image of the actual state of the object system is stored.
2. Create a new database as a copy of the old one and the received transition information. Now the database is growing: as in the first approach. The only difference is that the transmission of information is reduced. An aggregation technique is required.
3. Keep an image of a complete old state of the object system and register only the state changes of the object system. Now the information system may reconstruct the actual state of the object system, so the system keeps the same information as with the second method, however it takes less storage capacity but more time to produce information. Because the set of updates is growing one need also a policy to reorganise the database. Usually one creates from time to time a new complete image and deletes the old one and all updates from the period before the new image of the object system state.

There is a way to organize the second method with almost the same storage needs as the third method (cf. 5.3).

A last aspect of the registration function of an information system we will consider are inconsistencies between the actual state of the object system and the most recent image in the database. One of the differences is caused by a possible timelag between the moment of a state transition in the object system and the moment of registration. Usually the registration takes place after the transition and the timelag may vary from hours to months. It sometimes happens that a transition is registered before it happens: an example

is the registration of a job before the job is finished. A good solution to these problems is to register two moments: the time of state transition and the time of the registration. Another source of difference is the occurrence of observation and transmission errors. Sometimes these errors may be discovered at the moment of registration, because one of the database constraints would be offended if the update would be accepted, however in a lot of situations erroneous information is accepted and registered. Sometimes the errors are discovered later. Because control decisions may have been executed on basis of wrong information it is necessary to keep in the data base both the erroneous and assumed correct images of the states of the object system. In accounting systems this problem is "solved" by assuming the information systems records are a correct image of the object system process upto some moment in the history and all error corrections concerning the period before this moment that arrive after it, are considered as changes of the state of the object system at the moment of the arrival of the correction. Because in accounting systems all updates concern additions and subtractions and these operations commute this approach is feasible. However in other applications this is not possible.

Before we will consider the control function of an information system in more detail we will have a look inside it.



The information system may be split up into an automated part (a.i.s.) and a human part (h.i.s.). The probabilistic behaviour of the information system as a whole is mainly due to the human part: (including the decision makers). The a.i.s. fulfils the following functions for the h.i.s.:

1. registration of information of state transitions in the object system that is not received in computer readable form,
2. reporting over the stored information, i.e. over the actual state and the history of the object system,
3. analysis of the process of the object system to obtain information for the construction of (mathematical) models for the object system,
4. decision support to personnel in the h.i.s. with the control the object system.

The first two functions of the a.i.s. are classic. The reporting side contains all kind of operational reports like invoices, instructions etc. and general data base queries. The last two functions are more advanced. To control a system one has to know the set of feasible actions and the effect each action has. In a lot of decision situations (cf. 5.2) one needs a mathematical model to describe this action-effect relationship. The analysis-function is directed towards

- identification of the model, i.e. which model-structure can be used
- determination of the parameters of the model, i.e. determination of the model-instance.

The identification function is nowadays seldom incorporated in the a.i.s., although we may expect it will be in future. In a lot of situations the determination function is incorporated. The decision support function of an a.i.s. fulfils the following tasks:

1. the decision maker is offering a proposal for a decision to the a.i.s. and the a.i.s. is calculating the consequences or effects of the decision,
2. the decision maker offers constraints and a criterion to the a.i.s. and

the system is generating optimal decisions, i.e. decisions that are feasible with respect to the constraints and with maximal criterion value, and as in 1 the effects of the decisions.

Systems or subsystems that fulfil the decision support functions are called decision support systems (d.s.s.). Decision support systems are useful in case

- the decision situation occurs rather frequently i.e. from several times a day to several times a year,
- the decision situation can be modelled and the parameters can be obtained from the registration part of the information system or from external sources.
- there are several constraints and criterions for the choice of a decision; these criterions may be conflicting and the decision maker needs to consider several alternatives.

We conclude this section with some remarks on the architecture of automated information systems. The term architecture means both the structure of a system and the art of building it. We only touch the first point here. A.i.s. consist of hardware and software. Usually a.i.s. are constructed from existing hardware and software components. The hardware components are general purpose computers (micro, mini and mainframes), periferals like terminals, printers and disc drives, and communication equipment like local- and wide-area-networks. There is a trend to specialize the computers: large computers as database machines or file servers, and also large computers for number crunching, micro- and mini-computers for local data processing. Terminals are equipped with more intelligence to do a part of the work of the computer and they are often replaced by micro computers. The wide area communication networks are becoming very important. We may distinguish two different types:

- circuit switching: between to points temporarily a connection is created

- packet switching: a message from one point to another is split up into small parcels and each parcel is individually routed through the network of nodes and at the end again assembled to the original message.

Computers that are connected to each other by a network may be coupled directly or indirectly. A direct or real-time coupling means that the application on both computers are exchanging data at run time. Direct couplings can be realized by circuit switching and packet switching both. It depends on the frequences and volumes what method is to be prefered. In general communication networks may use both methods in combination: between some nodes circuits are created and between others packets are switched.

Indirect coupling is called message switching or electronic mail. One computer is preparing a file or message and send it to the mailbox of the other computer. This mailbox is part of the network. The other computer is reading its mailbox at a moment it is ready to do this.

The software components of a.i.s. are standard components like operating systems, editors, dialog monitors and program generators (that generate code in high level programming languages like Cobol, Fortran and Pascal).

To build on a.i.s. one has to develop programs using these software components. These programs are called application software; because it makes the components applicable to the special tasks of the information system.

3. Examples of functions of information systems

The object systems for logistic systems are usually split up by the companies that act in these processes. Some companies, such as forwarders and brokers do not participate in the physcial handling of cargo. They only process information. In fact they only have an information system, controlling object systems outside their own company.

3. 1. Registration and reporting

In logistic systems we see several autonomous information systems that monitor and control only parts of the total logistic systems. Hence there is a lot of communication between these systems. In many situations the companies that participate in logistic systems have sophisticated a.i.s. for their own partial logistic systems. However the communications between these companies proceeds in primitive ways: often one computer is producing a listing that is sent by post to another company where it is manually re-entered into an a.i.s. In more "advanced" situations the sending computer is automatically generating a telex that is entered automatically into the receiving computer as a text file. Then personnel on the receiving side has only to correct or reformat the file with an editor. Better solutions are only possible if all parties in logistic processes accept standards for message exchange on the highest level of the OSI-model (cf. Tanenbaum, (1981)). Even if these standards would be available there will be a lot of work in adapting existing information systems to those standards. It would be very helpful if software tools were available with which database administrators or end-users could define communication between their database and the one from another company. Such communications are now organized by application software programmed with tools mentioned in section 2.

What is needed is a facility to define the creation of a message if the database state satisfies a certain condition. The message will contain an update

of the database of the receiving information system. The condition may be defined by a view and the message as well. The condition may be translated as: the condition does not hold if the view is empty.

We do not go into more detail on these tools here. The reason we spend some attention to it is because the registration function of a logistic information system needs it. Registration of the status of cargo and carriers is the primary function of these systems.

The company or department that is handling the cargo is controlling and recording the status changes. Other companies often need this information also. Then the communication sketched above is needed. Often several companies are interested in the same status change. An electronic mail facility is very useful in such situations because setting up direct connections with a lot of different parties simultaneously is difficult in general and seldom necessary. What is said for the cargo holds more or less also for the carriers: stevedoring companies are interested to know the departure date of ships that will arrive at their terminals because they may use this information for their planning.

Beside the monitoring of the cargo and the carriers the information systems exchange: orders, instructions and invoices for the handling of cargo or the use of carriers. These are produced by the reporting functions of the information systems.

These two classes of functions belong to the registration and reporting functions of information systems. In the rest of this section we will give examples of the analysis and decision support functions. We remark that the most existing automated information systems nowadays are concerned with the first two functions only.

From the analysis functions we consider only one example.

3. 2. Analysis of production norms for break bulk.

During a (4 hour) shift a gang is handling several types of commodities. It is not registered how many time is spent to each commody type. How to obtain production norms from the total gang-time spent on the ships and the total tonnage per commodity type per ship?

The model that may be used is a simple regression model:

Y_i: random variable indicating the total number of gang-hours spent on ship i

x_{ij}: total tonnage of commodity type j on ship i

A_{ij}: the production-time for one ton of type j on ship i, a random variable.

In the model identification phase it was decided that we could express:

$$A_{ij} = n_j (1 + S_i)$$

where

n_j: the norm for commodity type j per ton

S_i: random disturbance due to factors like the weather and the storage of the ship.

These variables satisfy

$$Y_i = \sum_j n_j x_{ij} + (\sum_j n_j x_{ij}) S_i$$

It is assumed that $E(S_i)$, the mean of S_i satisfies $E(S_i) = 0$ and the variance $\sigma^2(S_i) = c > 0$ for all ships i. If X is the matrix with entries x_{ij} then the estimate for production norms

$$\hat{n} = (X^T X)^{-1} X^T y$$

(n and y are vectors with components n_i and y_i resp. and X^T is the transpose of X).

An (biased) estimate for σ^2 is: $\sum_i \{(Y_i - \sum_j \hat{n}_j)^2 / (\sum_i x_{ij} \hat{n}_j)^2\} / k$

where k is the number of ships in the database.

If the parameter estimation is incorporated in the a.i.s. we have to avoid that $X^T X$ is singular. In practise this can only be the case if one ship has two or more commodity types on board that do not occur on any other registered ship.

We continue with two other examples of functions of information systems, namely decision support functions. The first example concerns the determination of shortest routes. It is usefull for forwarders to compute a route for a parcel but also for shipowners or trucking companies.

3. 3. Decision support

Suppose we want to compute the best routes between any pair of a finite set of nodes (subset of $\mathbb{N}$). The "distance" between two nodes i and j is the cost, the time or the probability of failure of moving directly from i to j. If there does not exist such a direct link we assume the distance to be infinite in the first two cases or 1 in the third case. It is of importance to incorporate this decision support function in an information system if the "distances" are changing from time to time.

Let $d(i,j)$ be the distance from i to j. Note that we do not assume $d(i,j) = d(j,i)$.

The algorithm is due to Floyd (cf. [Aho, Hopcroft and Ullman (1983)]) and is based on the dynamic programming technique. Let $\{1,2,\ldots,n\}$ be the set of nodes.

Let $w_0(i,j) := d(i,j)$, which is ∞ (or 1 in the third case) if there is no direct link from i to j.

Define

$w_k(i,j)$ = minimal route length to go from i to j without passing any node with node number greater than k.

Clearly: w_n is the wanted function.

It is easy to verify by induction on k that w_k satisfies the following functional equation:

$$w_k(i,j) = \min \{w_{k-1}(i,j),\ w_{k-1}(i,k) + w_{k-1}(k,j)\}$$

in the first two cases. In the last case we have to replace $+$ by multiplication.

The complexity of the algorithm is $O(n^3)$. If we are only interested in the shortest distance between two points there is another algorithm due to Dijkstra (cf. Aho, Hopcroft & Ullman 1983) with complexity $O(n^2)$.

The second example concerns the determination of the capacity of a buffer for objects, for instance containers, that arrive according to a Poisson process.

Assume that the analysis function of an information system has determined that the arrival rate of objects per time unit is λ , and that the average residence time of the object in the buffer equals μ . If two arrivals in disjoint time intervals are stochastically independent and if the probability of an arrival in an interval is $O(1)$ then we have Poisson arrivals and the probability P_n of having (in the stationary phase of the system) n objects in the buffer is: (cf. [Kleinrock (1975)])

$$P_n = \frac{\rho^n}{n!} e^{-\rho} \quad \text{where } \rho = \lambda.\mu$$

Hence we may compute

- mean utilisation ρ
- standard deviation of the utilisation : $\sqrt{\rho}$

and we may choose a buffer capacity of

$$\rho + 2\sqrt{\rho}$$

if we require the buffer can store objects in 95% of the arrivals. If we have chosen the buffer capacity to be b , we get:

$$P_n = c. \frac{\rho^n}{n!} e^{-\rho} \quad \text{where} \quad c^{-1} = \sum_{n=0}^{b} P_n$$

These examples show that building of automated information systems for logistic systems requires:

- Database theory to design the registration and report subsystem
- Statistical theory to design the analysis subsystem
- Operations research to design the decision support functions. Here we may distinguish combinatorial models as used in the shortest routes case and stochastic models as used in the last case.

4. Some combinatorial models for logistics

In this section we consider some well+known combinatorial models and their algorithms, that are useful in decision support systems for logistics.
We start with some general remarks. A combinatorial model is the translation of a decision situation into the optimization of some function f over a finite set S. So we are interested in

$$v(f,S) := \min \{f(x) \mid x \in S\}$$

and in some x^* such that $f(x^*) = v(f,S)$.
The models we will consider have very large sets S and therefore we need 'good' algorithms to determine $v(f,S)$ and x^*. A 'good' algorithm is one that determines an optimal solution in polynomial time i.e. the computation time is $O(p(n))$ if n is the size of the model-instance and p a polynomial. Unfortunately for many combinatorial models that are relevant in practise no good algorithms are known and probably they do not exist. (These are the so called NP-complete problems (cf. Garey and Johnson (1979)). For such problems one sometimes develops heuristics i.e. approximations that are often good, but not in general, that require only polynomial computation time. Another approach is to relax the problem by extending the set S such that the minimum in the relaxed problem is the same as in the original one and the computation is easier. This is sometimes done by dropping the restriction that S only contains integer points (i.e. points with integer coordinates). In this case a problem is often relaxed to a linear program. Therefore we will consider linear programs briefly.

A lineair program is an optimization problem of the following form:

find the minimum of $\sum_{j=1}^{n} c_j x_j$ over all
vectors $x = (x_1,\ldots x_n) \in S$ where S is defined
by: $S = \{ x \mid (\forall\, j \in \{1,2,\ldots,n\} : x_j \geqslant 0) \wedge (\forall\, i \in \{1,\ldots m\} : \sum_{j=1}^{n} a_{ij} x_j = b_i)\}$

An element x of S is called extremal if and only if there do not exist $y, z \in S$ and $\lambda \in (0,1)$ such that

$$x = \lambda y + (1-\lambda) z$$

The properties we need are:

a. the set S is convex,

b. the minimum is attained in an extremal point of S,

c. each extremal point x of S has at most m components that differ from zero. An extremal point is usually called a basic solution.

There are several algorithms to solve linear programs, even in polynomial time. The most used algorithms are based on the simplex algorithm (not polynomial), for which there are very fast and reliable implementations. They all search only basic solutions (cf. Gass (1969)).

Many combinatorial models can be formulated as integer linear programs, i.e. linear programs with as extra constraint on S : x_i is integer for all i .

4.1. Simultaneous transports

Consider a set of sources $I = \{1, \ldots, m\}$ and a set of sinks $J = \{1, \ldots, n\}$. They may represent sites on earth. At each source i there is a supply of cargo a_i and at each sink j there is a demand for cargo b_j . Further there is a real valued cost function c over $I \times J$, where $c(i,j)$ expresses the cost of shipping one unit of cargo from i to j . The problem is to decide on the cargo movements.

First we consider the model:

Let x_{ij} be the amount of units to be shipped from i to j . Then the problem is

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

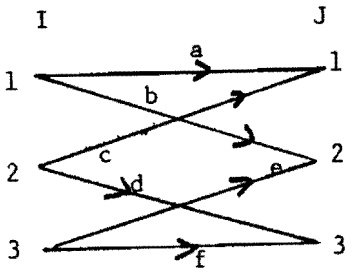
over the set S , defined by

$$S = \{x \mid (x_{ij} \geq 0 \quad \forall\, i \in I \wedge j \in J) \wedge$$

$$(\sum_j x_{ij} = a_i \;\forall\, i \in I) \wedge (\sum_i x_{ij} = b_j \;\forall\, j \in J)\}$$

This fits perfectly into the framework of a linear program. However the solution will be an integer solution. This is due to the fact that each

extremal solution of the problem correspondents to a bipartite graph without cycles. This graph is built from two sets of nodes: I and J and there is an arc between $i \in I$ and $j \in J$ if and only if $x_{ij} > 0$. To see that an extremal solution does not allow cycles consider an example:



Clearly there is a cycle (if we forget the directions). Hence we may change the solution

$x_{11} \leftarrow a - \varepsilon$; $x_{23} \leftarrow d - \varepsilon$

$x_{12} \leftarrow b + \varepsilon$; $x_{32} \leftarrow e - \varepsilon$

$x_{21} \leftarrow c + \varepsilon$; $x_{33} \leftarrow f + \varepsilon$

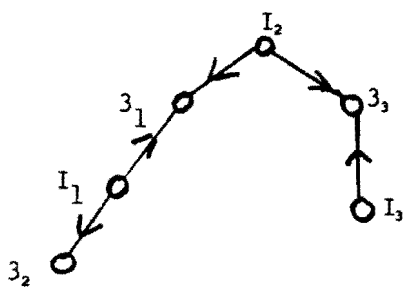
to obtain a new solution that satisfies the constraint, if

$|\varepsilon| < \frac{1}{2}\min\{a, b, \ldots, f\}$. However then

$x + \varepsilon$ and $x - \varepsilon$ are both solutions and

$x = \frac{1}{2}(x+\varepsilon) + \frac{1}{2}(x-\varepsilon)$ and is therefore not extremal.

So each extremal solution corresponds to an undirected, a cyclic graph, which is a tree. This s a spanning forest of the (bipartite) graph. Hence the values x_{ij} of a basic solution can be computed by additions and subtractions only. To verify this note that each node in a tree determines the value of the arc to its father-node given its own a or b value and the values of the arc's to its sons. So the terminal nodes determine directly their arc's and the other arc's are computed recursively. We will illustrate this by the example with x_{32} deleted.



Hence $x_{33} = a_3$

$x_{23} = b_3 - a_3$

$x_{21} = a_2 - x_{23}$

$x_{11} = b_1 - x_{21}$

$x_{12} = a_1 - x_{11}$

Hence we may use any linear programming algorithm to solve this problem with an integer solution.

The form of the matrix is $I = \{1,2\}$ $J = \{1, 2, 3\}$

x_{11}	x_{12}	x_{21}	x_{22}	x_{31}	x_{32}	
1	1					$=a_1$
		1	1			$=a_2$
				1	1	$=a_3$
1		1		1		$=b_1$
	1		1		1	$=b_2$

4. 2. Transshipment

Consider a set of locations each may have a demand or a supply of uniform objects. The decision to be made is to ship objects from one location to another to balance the differences. An important example is the transport planning for empty containers. Formally:

$$\text{minimize} \quad \sum_{i,j} c_{ij}\, x_{ij}$$

$$\text{over} \quad S = \{x \mid x_{ij} \geq 0 \wedge \sum_{k\in I} x_{ik} - \sum_{k\in I} x_{ki} = t_i \quad \forall i,j \in I\}$$

where I is the set of locations, c_{ij} the cost of the movement of one object from i to j and t_i the net stock (positive or negative). We want a transshipment such that all resulting stocks are zero. Hence we assume $\sum_{i\in I} t_i = 0$. This problem can be transformed into the transportmodel discussed above. This implies that if all t_i are integers then the x_{ij} are also integers.

We show this transformation.

Consider the transportmodel with $I = J$ defined by

$$\text{minimize} \quad \sum_{i,j} c_{ij}\, x_{ij}$$

$$\text{over} \quad S' = \{x \mid x_{ij} \geq 0 \wedge \sum_{k\in I} x_{ik} = t_i + B \wedge \sum_{k\in I} x_{ki} = B \ \forall i,j\}$$

let B be sufficiently large to ensure that $t_i + B \geq 0$ for all $i \in I$

Hence we may take $B = \sum_{i\in I} \max\{0, t_i\}$.

It is clear that $x \in S'$ implies $x \in S$ and therefore

the solution of the transport problem is at least the solution of the trans-

shipment problem. Also the opposite is true. To verify this note that in the transshipment problem we may choose x_{ii} arbitrarily and therefore we may choose $x_{ii} := B + \sum_{k \in I, k \neq i} x_{ki}$

Hence $\sum_{k \in I} x_{ki} = B$ for all i and

therefore $\sum_{k \in I} x_{ik} = t_i + B$

So any $x \in s$ implies $x \in s'$ and therefore the solution of the transshipment model is at least the solution of the transport model. So they are the same.

For transport and transshipment models there exist algorithms that are more efficient than the general linear programming algorithms. These algorithms exploit the special structure of these models (cf. Magnanti & Golden (1978)). Note that these models require a distance function as parameter. Hence first we have to solve a shortest route problem. We note that there is a large class of combinatorial optimization models that can be approximated very well by linear programming.

Models of the form

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{over } S = \{x \mid x_{ij} \in \{0,1\} \wedge \sum_{i} x_{ij} = 1 \wedge \sum_{j} a_{ij} x_{ij} \leqslant b_i\}$$

are called generalized linear assignment problems. Models of this form can be interpreted as: given m machines and n jobs, production capacity required by job j on machine i is a_{ij}, the cost of production is c_{ij} and the total capacity of machine i is b_i. It turns out that the relaxation of this problem, obtained by deleting the restriction $x_{ij} \in \{0,1\}$ has only few non-integer value x_{ij} in its solution. It holds that the number of non-integer value x_{ij} is less than or equal to m (see Benders & Van Nunen (1983)). Often m is small compared to n and therefore "very small" compared to $m \cdot n$, the number of variables.

The next combinatorial models are solved by dynamic programming techniques.

4. 3. Lot size scheduling

Suppose we have to deliver at equidistant moments in time a lot in some place over seas. We can of course ship each lot such that it is just in time for the delivery, however this might be very costly if the lots are small. Therefore it might be better to hire a depot over seas and ship large amounts from time to time to the depot. Of course now we have inventory cost, but the cost of shipping can be reduced. We assume inventory cost to be proportional to the inventory and each shipment has constant cost: K.

There are of course several variations of this model, most of them can be treated in the same way. This model formulation is due to Wagner and Whiten (cf. Wagner 1975).

Let $a_1, a_2 \ldots, a_n$ be the lots to deliver at moments 1, 2, ..., n. We make two important observations

a. if there is enough in the depot to deliver for the next moment we do not ship cargo, because we may postpone this to the following moment and so we save at least the inventory cost of the planned shipment.

b. it is only useful to ship partial sums of $\{a_1, a_2, \ldots, a_n\}$

i.e. ship for time point n : a_n or $a_n + a_{n+1}$ or $a_n + a_{n+1} + a_{n+2}$ etc.

because other quantities make extra shipments necessary and therefore only create inventory cost.

Let: $s_0 := 0$, $s_n := s_{n-1} + a_n$ for $n = 1, 2, \ldots, n$

and let $S := \{s_1, s_2, \ldots, s_n\}$. Note that S represents all the interesting inventory levels of the depot (according to b.).

Let $v(s,n)$ be the minimal cost of operation if we have at moment n a cumulative inventory of s in the depot. (i.e. the sum of the shipments to the depot is s). It is required that $s \geq s_{n-1}$ (i.e. in the former periods we delivered the demands).

According to observation a we get: $(v(s,N+1) = 0 \quad \forall s \in S$

$$v(s, n) = h.(s-s_{n-1}) + v(s, n+1), \text{ if } s \geq s_n$$

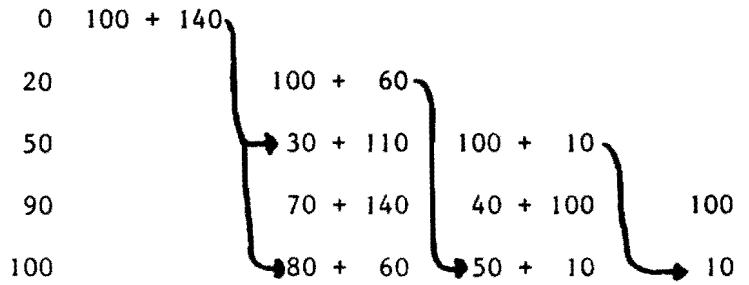
$$= K + \min_{\ell \geq n} v(s_\ell, n+1), \quad \text{if } s = s_{n-1}$$

here h represents the inventory cost per unit. We assume inventory cost are paid over each period the cargo was totally or partially available in the depot. We give a numerical example:

Let $K = 100$, $h = 1$, $N = 4$ and

$a_1 = 20$, $a_2 = 30$, $a_3 = 40$, $a_4 = 10$

$S =$



The arrows indicate the optimal decisions.

4. 4. Optimal loading (knapsack problem)

This model is a simple example of a loading problem however there is no polynominal-time algorithm known to find an optimal solution.

Suppose we have cargo units $1, 2, \ldots, n$. Each unit i needs storage capacity: a_i and has a value: w_i ($\geqslant 0$). This value can be the selling price of the cargo unit but also the penalty to pay if it is delivered too late. The decision to be made is which units will be shipped if the total shipping capacity equals b.

Formally:

$$\text{maximize} \quad \sum_{j=1}^{n} x_j \; w_j$$

$$\text{over} \quad S = \{ x \mid x_j \in \{0, 1\} \wedge \sum_{j=1}^{n} x_j \; a_j \leqslant b \}$$

A useful algorithm is based on dynamic programming:

let

$v(i, z)$ be the maximal value we may ship if the free space left equals z and we may only consider $i, i + 1, \ldots, n$ (because over the other units

we have already made a decision).

Then we are interested in: $v(1, b)$.

Note that v satisfies $v(n+1, z) = 0$ for all z

$$v(i, z) = \max \{v(i+1, z),\ w_i + v(i+1, z - a_i)\}$$

where the first term on the right side corresponds wit the decision not to ship unit i and the second term with the decision to ship it.

If we assume a_i to be integer than the parameter z may range over $\{0, 1, 2, \ldots n\}$.

The computation time is $0(n\,b)$.

4. 5. Routing one carrier

This model is known as the traveling salesman problem. There are locations 1, 2, ..., n and the distance to go from i to j is c_{ij}. The decision to be made is the order in which the locations are visited, starting from and returning in location 1.

Formally the problem can be stated as:

$$\text{minimize} \quad \sum_{i,j \in I} x_{ij}\, c_{ij} \qquad (c_{ij} = \infty \text{ if there is no direct link from } i \text{ to } j .$$

over the set:

$$s = \{\, x \mid x_{ij} \in \{0,1\} \wedge \sum_j x_{ij} = 1 \wedge \sum_i x_{ij} = 1 \wedge$$
$$\forall\, V \subset I \wedge V \neq \emptyset : \sum_{i \in V} \sum_{j \in I - V} x_{ij} \geq 1 \,\}$$

The last restriction says that no sub-tours are allowed: for all subsets V of I there must be at least one way out. If we would drop this last requirement we would have a special transportation problem! In fact this relaxation of the travelling salesman problem is often used in branch and bound algorithms. We will not describe an algorithm for this model because we consider a more general problem below. However we note that a good heuristic for the traveling salesman is obtained by starting with some route and then applying K opting (cf. Aho, Hofcroft & Ullman 1983). This technique is to

delete k connections of a route and trie to reconnect the loose ends to get a better tour.

4.6. Simultaneous routing carriers from one depot

This problem is well-known as the vehicle routing problem (cf. Christofides (1985)). It generalizes the problem mentioned in section 4.5.

There is one depot and we have a set of m carriers each with its own capacity. Further we have a set of locations $I = \{1, 2, \ldots, n\}$. At each location we have to pick up or to bring some cargo.

The distance from location i to j is: d_{ij}. The decisions to be made are the route of the carriers such that all locations are visited and the carriers return to the depot (code=o) where the criterion is to minimize the total travel distance.

There may be difficult additional constraints like so called time windows: a site may be visited only within these windows. However if we assume at all sites cargo must be delivered or at all sites it must be picked up and we omit other constraints, than we may formulate this model as:

$x_{ijk} = 1$ if carrier k visits site j directly after site i

$= 0$ otherwise

$y_{ik} = 1$ if site i is visited by carrier k

$= 0$ otherwise.

b_k = is the total capacity of carrier k

a_i = the amount of cargo at site i

minimize $\sum_{i,j \in I} c_{ij} \sum_{k=1} x_{ijk}$

over the set S where

$$S = \{\langle x,y\rangle \mid \sum_k y_{ik} = 1 \quad \text{for } i \in I$$

$$\wedge \sum_k y_{0k} = m \wedge \sum_i a_i y_{ik} \leq b_k \quad \text{for all } k$$

$$\wedge \sum_j x_{ijk} = \sum_j x_{jik} = y_{ik} \quad \text{for all } i \text{ and } k$$

$$\wedge \forall V \subset I \wedge V \neq \emptyset: \sum_{i \in V} \sum_{j \in I-V} x_{ijk} \geq 1 \quad \text{for all } k \}$$

There is of course no polynomial-time algorithm to compute the optimal

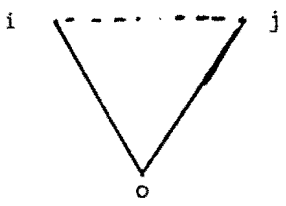
solution known. As mentioned above there are in practise several constraints that cannot be dealt with easily. There exists a simple algorithm due to Clark and Wright (cf. Magnanti & Golden (1978)) which allows all kinds of constraints. Beside that it has a rather good performance, i.e. "often" a few parcels worse than an optimal solution.

The algorithm is based on so called savings.

We define

$$S(i,j) = c(0,i) + c(0,j) - c(i,j) \qquad i,j : \text{locations}$$

These quantities give the saving of visiting i and j after each other in one tour instead of two:



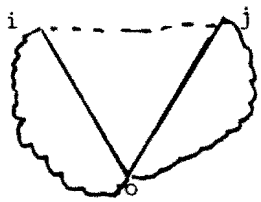
visiting i and j from 0 and returning gives:

$2\ c\ (0,i) + 2\ c\ (0,j)$

Combining gives:

$c\ (0,i) + c\ (i,j) + c\ (0,j)$

The difference is S(i,j). Now suppose we have already formed a route ending in i and another one ending in j . Let the length be respectively a and b.



Then a combination of routes may be considered:

not combining costs: a + b

combination costs: a-c(0,1) + b-c(0,j) + c(i,j

The difference is again S(i,j). Note that we must check the constraints to verify if two routes can be combined for one carrier. A sketch of the algorithm:

- create an empty list of route lists T .
- compute all savings, put them on a list L .
- while L not exhausted do
  - take the greatest element of L , let the points be (i, j)
  - look for routes in T that have i or j as one of their endpoints:

- case 1: no such routes, then create a new route in T, consisting of (i,j) and delete it from L.
- case 2: one route ends in, say i and no other ends in j; then add j to the route of i if the capacity constraint allows this. Delete the pair (i,j) from L and in case j is added to the route delete all pairs on L with i and all pairs (k,j) where k is out route of i. If the capacity constraint is forced then start a new route as in case 1.
- case 3: there are 2 routes; one ending in i, the other in j; combine these two routes if the constraints allow this. Delete (i,j) from L. In case the combination is made, delete all pairs in L which involve i or j. If the constraint is forced, create a new route.

{end}

Note that it may occur that we get more tours than we have carriers. The combinatorial optimization models presented here are classic. In a real-world decision situation it seldom occurs that one of these models gives a good description of the decision situation. Usually there are more constraints on the solution and mostly the models presented here describe only a part of the situaion. Therefore in decision aupport systems we have to integrate several models, each describing only a part of the decision situation. Further we need more "robust" models, i.e. models that can handle many constraints of different forms. Finally we note that in decision support systems stochastic models play an important role too.

5. Design theory for decision support systems

5. 1. Introduction

We use the term decision support system for a subsystem of an information system which task it is, to support decision makers. As an information system keeps track of the process of its object system a decision support system records information of possible futures of the object system. Usually the process history is recorded in more detail than the possible futures. As remarked in section 3 a decision support system may assist the decision maker by calculating:

- the effects of decisions or a decision rule (i.e. a function from state or state history to actions) generated by the decision maker
- generate optimal decisions or rules and their effects with respect to a criterion given by the decision maker.

In theory this can be done by simulating the process of the object system under various different policies, and then choosing the best one.
In practise it is only necessary to simulate some aspects of the object system, because many facets of the process of the object system are not under influence of the decisions and have no influence on the effects of decisions. Simulation as a technique for decision support has some drawbacks:

- It may require very much computing time to obtain the effects of only one decision (rule).
- It requires very detailed information of external influences in the future and also of internal mechanisms in the object system.

The first point implies that simulation is seldom a feasible technique to generate optimal solutions and the second point indicates that the data obtained by simulations may be unreliable. Note that we do not reject simulation techniques for decision support systems. Sometimes it is the only available possibility, sometimes it may be used in combination with

other mathematical models. A decision support system is based on one or several models that represent aspects of the object system. We discuss them in section 5.2. In 5.3 we consider data modelling. Here we introduce a data model and a way to implement databases for decision support efficiently. In section 5.4 we consider the functions a decision support system has to fulfil. Finally in section 5.5, we consider the design process for decision support systems.

5.2. Decision models

As the schema of a database, for registration of the process of an object system, represents the relevant aspects of the state space of the object system, so the decision model represents aspects of the whole object system that are relevant for the determination of the control. We note that the latter representation is usually less detailed and less self-evident than the first one. The decision model can be considered as a 4-tuple (cf. section 2)

$$< S, A, R, P >$$

where S represents the state space of the object system, $A = E_1 \times I_1$ where E represents the set of possible external influences, I_1 the input of the decision makers, $R = E_2 \times I_2$ the external and internal responses and P the transition mechanism of the object system. For simplicity we assume the time to be discrete, i.e. there is a time unit and all changes are considered to occur at these discrete moments only.

Sometimes the decision model does not represent the dynamics of the object system explicitely. This is the case when the decision model describes:

1. only one period for which decisions have to be taken,
2. the stationary case, in which the "behaviour" of the object system is the same in each period.

Sometimes these cases coincide because after one period the start-position

for the next period is the same as in the former in spite of the decisions made.

When we speak of decision model we must distinguish model structure, i.e. a description in terms of constants and variables or parameters, and model instance, a set of pairs: for each parameter a value. We note that parameters may have complex data structures; for instance a road network may be one parameter. We distinguish two types of parameters:

- uncontrollable parameters, partly determined by external factors, such as the arrival rate of customers, partly determined by internal factors, such as production speed of a machine,
- controllable parameters or decision variables.

Given the model structure the model is determined by the parameter sets. The uncontrollable parameters are obtained from several resources: partly from the registration database by an analysis function, partly from external sources like statistical surveys and partly by guesses from the decision makers. Therefore there will be several different model instances that describe the object system under different assumptions.

A decision rule or strategy assigns to an observed process-history of the object system an action.

The set of observable process-histories

$$H_1 = S \times (I_1 \times I_2)^*$$

(the set of all finite sequences of pairs of an action i_1 on the object system and an observation i_2 of the object system starting with the begin state of the system). A decision rule d is a function: $d \in H_1 \rightarrow I_1$. The set of all feasible decision rules is: D . The transition mechanism P may be deterministic or stochastic. Since the first one is a speciale case of the latter we only consider the latter. P assigns to each process-history a probability over $S \times R$. When the system is at some moment in a state $s \in S$,

it receives an action $a \in A$ and then it moves to some state $s' \in S$ and emits a response $r \in R$.

The set of process-histories is:

$$H_2 = S \times (A \times R \times S)^*$$

(note that H_1 is a projection of H_2).

So P is a function: $P \in H_2 \rightarrow Q(S \times R)$ where $Q(S \times R)$ is the set of probability distributions over $S \times R$. So given a decision rule d and a starting state in S the system will "move" from one state to another.

To compare different decision rules, the decision maker has a set F of evaluation functions.

An evaluation function assigns to each observed process-history a real number, i.e. for all $f \in F$: $f \in H_1 \rightarrow \mathbb{R}$

In case of a stochastic transition mechanism the value of a evaluation function is computed as the mean-value of the evaluation function over H_2. Note that the transition mechanism P induces a probability measure over H_2 and each $h_2 \in H_2$ determines a $h_1 \in H_1$. We denote this mean-value by

$$\mathbb{E}\ (f \mid d,u)$$

where d is a decision rule and u a function that assigns to each uncontrollable parameter a value.

A typical example of an evaluation function f is:

$$f(h) = \sum_{n=0}^{N-1} \beta^n\, g_1(i_n,j_n) + \beta^N g_2(j_n)$$

where $h = (s, i_1, j_1, \ldots, j_N) \in H_1$, $i_n \in I_1$, $j_n \in I_2$

Here $g_1(i_n,j_n)$ is the cost of excercising action i_n and observing j_n in stage n and $g_2(j_n)$ the final return, and β a discount factor. Hence $f(n)$ represents the present value of the cash flow.

Usually evaluation functions are "easy" to compute.

In most practical cases there are several different evaluation functions that oppose each other, i.e.

$f, g \in F$ oppose each other if there exist $d_1, d_2 \in D$

such that $\mathbf{E}(f \mid d_1, u) < \mathbf{E}(f \mid d_2, u)$ and $\mathbf{E}(g \mid d_1, u) \geq \mathbf{E}(g \mid d_2, u)$

for some parameter function u.

The decision maker is confronted with a multicriteria decision problem. This kind of problems is usually solved by chosing a linear combination of evaluation functions and consider this as criterion function which has to be optimized. Often we consider several linear combinations to define restrictions. Then we are looking for a decision rule $d^* \in D$ such that

$$\mathbf{E}\left(\sum_i a_i f_i \mid d^*, u\right) \geq \mathbf{E}\left(\sum_i a_i f_i \mid d, u\right)$$

for all $d \in D$ satisfying

$$\mathbf{E}\left(\sum_i b_{ji} f_i \mid d, u\right) \leq c_j \quad \text{where } j \in J \text{ and } J \text{ some finite set}$$

Here a_i, b_{ji} and c_j are real-valued coëfficients and i is an index over F.

The decision maker is usually "playing" with the coefficients.

We call a combination of a decision rule d, a parameter function u, a set of evaluation function (mean-) values and a set of coefficients a scenario.

Decision makers will design, compute and compare several scenario's.

We reconsider two examples.

In the simultaneous routing problem (cf. section 4) the state space is determined by

- set of carriers, each with its capacity
- set of sites, and their volume to receive or to ship
- road network with the distances

The state at some moment D the position of the carriers on the road network and their contents, and the positions of sites. The actions from the information system or decision maker are the assignments of successive locations for the carriers. They may be given in advance or during the trips. The reponse

to the information system may be a signal from each carrier if a location is reached or just one signal if the carrier has returned to the depot. The transition mechanism may be considered deterministic and is determined by parameters as the speed of each carrier and the loading times at each location. It might be more realistic to assume probability distributions as parameter values however they may be difficult to obtain. Replacing random variables by their mean-value is a standard way of modelling. For the computation of the decision rule one may use these mean-values, to evaluate the effect of a decision rule one may simulate the process using the probability distributions.

In the buffer capacity example there is only one decision to make the volume of the buffer. It is determined at the beginning of the process-history and kept fixed. The state space is described by the set of objects in the buffer and the time they stayed already in the buffer. The transition mechanisme is described by the Poisson parameter of the arrival process, the residence time distribution of the objects in the buffer. Instead of simulating the buffer process one could compute directly the (mean-)values of some important evaluation functions, such as the probability to have n objects in the buffer (provided n is not greater than the capacity). However the decision maker could be interested in other characteristics which cannot be computed analytically.

In general we may state that a combination of simulation and analytical optimization models is a good approach:

- a simulation model is used to evaluate (by means of evaluation functions) a decision rule.
- an analytical optimization model, which might be a further simplification of the object system is used to determine a decision rule.

A more sophisticated coupling between a simulation and an optimization model arises when the simulation model computes parameter values that are used in the optimization model. This occurs when we have networks of submodels each representing some aspects of the object system.

For example consider the routing problem again. Suppose that we have non-deterministic loading times at the sites.

The loading times at the sites are determined by the process at the site. This process is influenced by other carriers that need service at this site and by the number of equipment and staff available at the arrival time of the carrier. On the other hand if more carriers are sent to the same site because the capacity-demand of that site is large then the routing decision rule is influencing the handling at the site. (The routing may be arranged such that the carriers arrive at the same time or more spread over the planning period).

Often it is impossible to take into account all these aspects in one model and therefore one decomposes the total object system in a network of sub-models.

Each submodel may require parameters from one or more submodels and may deliver parameters to other submodels. In fig. 5.1 there is a strong interaction between the submodels. Of course each model may have several evaluation functions that are not considered in the diagram. With these models we may associate functions f_1 for M_1, f_2 and f_3 for M_2 and f_4 for M_3 such that given the incoming parameters these functions compute the outgoing parameters. We are looking for parameter values $p_1 \ldots p_4$ such that $p_1 = f_1(p_2, p_4)$, $p_2 = f_2(p_1)$, $p_3 = f_3(p_1)$ and $p_4 = f_4(p_3)$ Written in vector notation: we are looking for $p = (p_1, \ldots, p_4)$ such that $f(p) = p$ (where F is a vector-valued function such that $F_1(p) = f_1(p_2, p_4)$ etc.

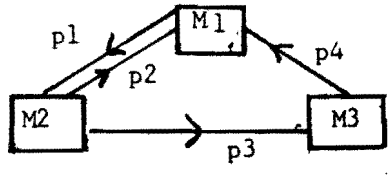


fig. 5.1

Hence we are looking for a fixed point of F. Often the computations to evaluate the function F are very time consuming and therefore we can only compute these values for a few parameter vectors. (Note that in our example of carrier routing one of these models is a complicated optimization model). In practise we "solve" this problem by starting with a parameter vector $p^{(1)}$ based on historical data and then we iterate a few times, i.e. we compute

$$p^{(n)} = F(p^{(n-1)})$$

It is seldom possible to derive properties of the function F such as a contraction property that would imply convergence. However in practise we encounter often stable systems, i.e. for relatively small n: $p^{(n)}$ and $p^{(n-1)}$ do not differ much. We will consider two special cases of these model networks:

1. Detail hierarchy. Here a hierarchy of models is considered in which models at a higher level use parameters that are obtained from lower level models by some aggregation procedure. For instance a model at some level is using a probability distribution for a service time and in a lower level model this distribution is computed with some parameters of the higher level, for instance the number of service units. It is usual that models at a lower level have a smaller time unit than models on higher levels.
2. Decision hierarchy. Here we have different decision situations. For instance at the top level we have to decide on the capacities of equipment or buffers and at the lower level we have to decide how to assign these capacities to the different tasks. To make the first decision we have to know the effects on the lower level. So we have to optimize the operations on the lowest level under the capacity conditions determined on the highest level. Another example of this situation is that at the highest level a price is fixed for some service and that in the lower level model the use of the service at this price is computed such that

at the highest level the service demand can be used to adapt the price to reach some goal.

In many cases we use optimization models to determine some decision rule and simulation models to compute the effects of these decision rules. If it turns out that the rules obtained by the optimization models are acceptable after some experimentation, then one may restrict the use of the simulation models for incidental verifications.

The approach sketched above is used frequently, usually without explicitly following the next steps:

1. Describe in as much detail as possible the object systems model in terms of the 4-tuple <S, A, R, P > and a set of evaluation functions.
2. Design for parts of this model submodels in the same terms in such a way that the evaluation functions and decision rules can be computed in acceptable time.
3. Design or select algorithms for the computations of step 2.
4. If the submodels have parameter dependencies such that they form a network of models, design an iteration algorithm to determine acceptable parameter values.

Example of step 4:

```
d := d0 ;                    {d0 is the start decision rule}
sim (d, p, r)
P0 := inf ;                  {inf is some "verly large" value}
while  ||p0 - p|| > ∈ . || P || do
 begin
P0 = p ;
optim  (p, d) ;
sim  (d, p, r) ;
end
```

Here sim is simulation algorithm that computes a parameter p and an effect r , given a decision rule r . This parameter p is used in an optimization algorithm 'optim' to determine a new decision rule d .

This approach results in a decision support system that may compute very good decision rules under the assumptions of all the submodels. In practise it frequently occurs that the environment of a decision support system, i.e. the object system is changing. In such a case there is a chance that the model assumptions do not hold anymore. Then one may adapt the models which means that the algorithm and therefore the programs have to be adapted, which may be costly (time and money) and dangerous. There are examples where the system is not adapted but where a shell is built around the system to translate the advised decision rules of the decision support system into feasible actions for the object system.

Another approach is to disregard the optimality of decision rules to gain robustness for changes in the object system. In this case the only goal is to let the system consider more alternatives than the decision maker itself could consider in the same time. In section 5.4 we return to this subject.

5.3. Databases for decision support

In this section we consider a functional datamodel to describe object systems and the parameters of decision models. The choice for a functional data model is based on the experience that it is very adequate to model real-world systems and it has a nice graphical representation. On the other hand it can be implemented like a network database which can be very efficient. Our choice is not very important because it is easy to transform a functional database schema into a relational database schema and vice versa. (cf. Tsichritsiz and Lochovsky (1984)).

In decision support systems we need to store several different scenario's. We will consider each scenario as a database state and this means that we have to store several different database states. However these states do not differ very much and therefore we are looking for an efficient implementation. We start with some formal definitions.

def. 1. A functional database schema is a 5-tuple

< C, F, D, R, V>

where:

- C is a finite set of category indices
- F is a finite set of function class indices
- D is a function, $D \in F \rightarrow C$ and it determines for each function class a category called the domain category
- R is a function, $R \in F \rightarrow C$, it determines for each function class a category, called the range category
- V is a set-valued function, with domain C . For each $c \in C$ V(c) is the set of representations of all possible objects that belong to category c .

Example

Consider the transport company with the simultaneous routing problem

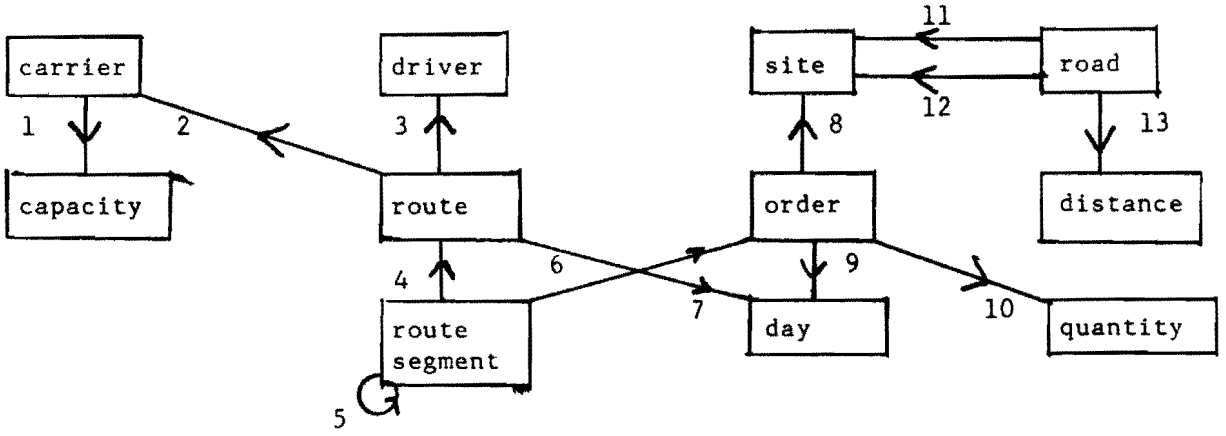


fig. 5.2

All the category indices are written in a rectangle, all function class indices are written along the directed arcs. The functions D and R are represented by the arcs: the origin is the D-value, the destination the R-value. The function V is not defined yet, however it is easy to define some coding for the objects belonging to some category, for instance capacity, distance, and quantity may be coded by positive reals, carrier, driver and site by character strings, routes by a positive integer or by the combination of the codes of carrier, driver and day if this combination uniquely determines the route. (Note that we have in the latter case a constraint that says that the value of the function with index 2 is equal to the first element of the code for the route. Constraints will be discussed below).

A functional data model is related to an entity-relationship model, however in the functional data model we do not distinguish entities, relationships and attributes: we call them all categories.
A functional data model can be considered as a semantic data model if we use verbs as names for function classes.
If we call, for instance, functions 1 and 2: "has" and "is made by" respectively, then we may read this part of the diagram as "route ... is made by carrier ... that has capacity ...".
We continue the definition of the data base state space:
def. 2. For data base schema $< C, F, D, R, V >$ we define

- $\tilde{C} := \{c \mid c$ is a set-valued function $\wedge$ dom(c) = C $\wedge$
  $\forall i \in C : c(i) \subset V(i)\}$
- $\tilde{F} = \{f \mid f$ is a function-valued function $\wedge$ dom(f) = F $\wedge$
  $\forall j \in F : f(j) \in V(D(j)) \rightarrow V(R(j))\}$
- the database state space is called S ,
  $S := \{ <c, f> \mid c \in \tilde{C} \wedge f \in \tilde{F} \wedge \forall j \in F :$
  $dom(f(j)) \subset c(D(j)) \wedge$
  $rng(f(j)) \subset c(R(j))\}$

the elements of S are called database states.

(Note that $X \rightarrow Y$ is the set of all possibly partial functions, whose range is subset of Y . Note further that dom and rng assign to a function its domain or its range).

Returning to our example we may interpret the diagram also as a representation of a database state: each rectangle represents the set of objects that belong to that category in that state and each arc represents a function between two of these sets. The function classes 11 and 12 represent the begin and end site of a road respectively, 13 the distance of the road. The function class 5 indicates the order in which the sites are visited. The other function classes are self-evident.

Now we have defined the unrestriced state space. However there may be several restrictions a correct database state has to fulfil, such as

- a function from one category to another must be total
- a function must be a surjection
- a function must be an injection
- for a category one or more total functions with this category as domain determine uniquely the objects of the category, i.e. if two objects in the category state have under these functions the same images then they are identical (This is a key-constraint).

In general we may define restrictions with expressions that represent predicates such as in our example:

$$\forall \langle c, f \rangle \in S : \forall \ell \in c \text{ (route)} :$$

$$(\Sigma\, r \in c \text{ (route segment)} : f_4(r) = \ell \Rightarrow f_{10} \cdot f_6(r)) \leq f_1 \cdot f_2(\ell))$$

(Note that f_4 means the function f(4) and $f_{10} \cdot f_6$ means function composition).

This restriction says that for each (partial) route the total load does not exceed the capacity of the carrier.

In general we define:

def. 3. Let $< C, F, D, R, V >$ be a database schema with state space S. Further let P be a predicate over S. Then the feasible state space FS is:

$$FS := \{s \in S \mid P(s)\}$$

Note that P is the conjunct of all constraints on the database states.

To implement a database according to a functional database schema there are several possibilities:

1. one may transform it to a relation database schema:
   - for each category that is the domain of some function class define a relation scheme (with the same name);
   - for each function class with this category as domain define an attribute, with name equal to the range category of the function class;
   - define one key-attribute to represent the element of the category itself; if there is a key-constraint then we may skip this attribute.
2. one may implement it as a network directly:
   - for each category we define a file with: a field to represent the objects of the category (the type is given by the function V), and for each function for which is a domain category we define a pointer to the corresponding file of the range category.
   - one may add indices like b-trees on the category files to get fast access to the categories
   - one may add per function class two pointers, one in the range-category file and one in the domain-category file to implement the inverse functions. Consider a function class with domain category A and range category B. Then we have for A and B a file with the structure of fig. 5.3.

Note that we assume an ordering in the inverse image of each element of B . This ordering may coincide with the order in which they were created.

A

key for object	pointer to B	pointer to next A with same B

B

key for object	pointer to first A pointing to it

fig. 5.3

3. one may transform the functional database schema into DBTG-schema.

We choose the second option to continue our discussion. As stated before in decision support systems we have to store several "images" of the object systems future each of which can be considered as a database state. Often these images do not differ very much, i.e. given two database states $s = <c,f>$, $\tilde{s} = <\tilde{c},f>$ their <u>difference</u>, defined by

$$d(s,\tilde{s}) := \sum_{i \in C_i} \# (c(i) \Delta \tilde{c}(i)) + \sum_{j \in F_i} \# (f(j) \Delta \tilde{f}(j)),$$

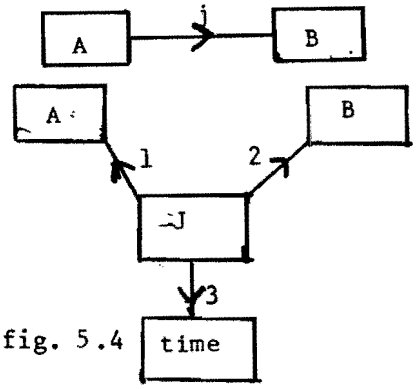
is small compared to $\sum_{c \in C_i} \# (c(i))$.

(Note that $A \Delta B = (A \setminus B) \cup (B \setminus A)$ and a function like $f(j)$ can be considered as a set of pairs).

This situation does not only arise in decision support systems but also in databases for process registration where not only the actual state of the object system but also the history of the process must be stored.

There are at least two methods to deal with this problem. First of all we may consider the different states at the level of the schema. This means that we bring "version" or "time" as category into the schema. We call the schema we obtain in this way, the <u>historical schema</u> with respect to the original one. We consider one way to create such a historical schema. Consider a part of a schema, consisting of categories A and B and a function class j from A to B . We add to the scheme for each function class j a new

category J and one for all, called Time. We define function classes as indicated in fig. 5.4.



(part of) original schema

(part of) its transformation into the historical schema

fig. 5.4

For the representation of Time we may use nonnegative integers and we have to require

$$\forall\, j_1, j_2 \in c(J):$$
$$(f_3(j_1) = f_3(j_2) \wedge f_1(j_1) = f_1(j_2)) \Rightarrow f_2(j_1) = f_2(j_2)$$

The relationship between the two schemas is expressed by:

if for the original schema in the state at time t holds

$$f_j(a) = b \quad \text{for some} \quad a \in c(A) \quad \text{and} \quad b \in c(B)$$

then we have for the state of the historical database at some time larger or equal to t :

$$\exists\, i \in c(J) : f_1(i) = a \wedge f_2(i) = b \wedge f_3(i) \not\leq t$$
$$\wedge\ (\forall i' \in c(J) : (f_1(i') = a \wedge f_2(i') = b) \Rightarrow$$
$$(f_3(i') < f_3(i) \vee f_3(i') > t))$$

Note that we do not express in this way if an object in some category is in a state or not. We consider an object "inactive" at some moment if it is not connected to other objects, i.e. is not occuring in the domain or range of some function at that moment. (There are other solutions too).

Note further that def. 2 implies that no deletion of category elements that were active at some moment are allowed. For retrieval purposes it is attractive to implement the inverse of function 1, in the example of fig. 5.4, since then we can retrieve the element of B by giving an element of a and the time and we only have to follow pointer chains.

The second method we will consider is to implement the versions only on the level of the database state space, which means that we do not introduce special categories and function classes but that we extend the state space by a time set:

def. 4. (Recall the notations of def. 2)

- $HC := \{c \mid c$ is a set-valued function $\wedge$
  $dom(c) = C \times T \wedge \forall i \in C \wedge t \in T : c(i,t) \subset V(i)\}$
- $HF := \{f \mid f$ is a function-valued function $\wedge$
  $dom(f) = F \times T \wedge (\forall \langle j,t\rangle \in F \times T :$
  $f(j,t) \in V(D(j)) \rightarrow V(R(j))\}$
- the process-space of the database PS is:
  $PS := \{\langle c,f\rangle \mid c \in HC \wedge f \in HF \wedge (\forall \langle j,t\rangle \in F \times T :$
  $dom(f(j,t)) \subset c(D(j),t) \wedge rng(f(j,t)) \subset c(R(j),t) \wedge$
  $(\forall i \in C : \forall t_1,t_2 \in T : t_1 < t_2 \rightarrow c(i,t_1) \subset c(i,t_2))\}$
- the space of histories upto t $HS(t)$ is:
  $HS(t) := \{\langle \tilde{c},\tilde{f}\rangle \mid \exists \langle c,f\rangle \in PS :$ $\tilde{c}$ and $\tilde{f}$ are the
  restrictions of c and f to the domains
  $C \times \{0, 1, \ldots, t\}$ and $F \times \{0, 1, \ldots, t\}$ resp$\}$

  (Note that we assume here $T = \mathbb{N}$).

In an implementation we have to implement elements of $HS(t)$ efficiently: For each category we define a file with a field for the key and for each function for which the category is the domain, we have a pointer to a file per function. Each of these function files has three fields: one for the time or version, one for a pointer to the range category and one to a next record in the "history chain" of the same file. This last pointer points to the record which 'contains' the time of the next change and the pointer to the new value in the range category file. Note that this construction is implemented in principle in the same way as

the one obtained by the historical schema approach, if backpointering is applied there.

Note that we store a function of time by only storing the jumps of its graph.

If a function is at some time not defined then the pointer to the range category file is nil.

We illustrate this with an example, see figs. 5.3 and 5.4:

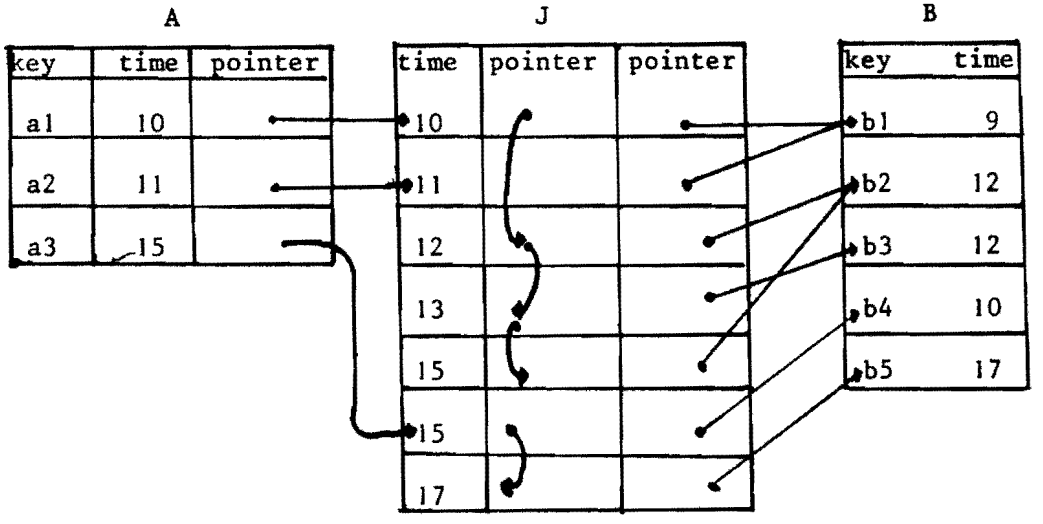


fig. 5.5

Note that we include in each category file a field for the time the record was created.

We conclude with some remarks:

- the term 'pointer' may refer to a physical pointer, indicating an address, but to a logical pointer, indicating a key value as well.
- it would be desirable if database management systems would have one of the following facilities to maintain several versions: or a facility to create from a given schema its historical schema, or to create from a given schema directly the datastructure for history tracking.
- instead of one "version" for the whole database schema one could consider given parts of the schema its own "version" administration. An advantage is

that these versions are smaller which may save space.

5.4. Functions of decision support systems

In a decision support system we have one decision model structure with several instances. Each instance is part of a scenario. We describe the structure of the scenario's by a database schema and so each scenario is represented by a database state. A scenario consists of 4 groups of data:

- uncontrollable parameter values
- decision rules
- (mean-)values of evaluation functions
- coefficients (to construct restrictions and a criterion from evaluation functions)

One group of functions of a decision support system is meant for handling these data. We call them the manipulator functions.

Manipulator functions.

These functions allow the decision maker to:

- Update manually uncontrollable parameters.
- Load automatically uncontrollable parameters from other databases;
- Update manually decision rules (including the insertion of a decision rule created by the decision maker and the changing of a decision rule generated by the system);
- Update of coefficients;
- Report data from one or several different scenario's. The report functions must include general retrieval as simple, user defined computations. It is important that several scenario's can be compared. If scenario's are only partly defined, because some evaluations are not computed or a decision rule is not even defined, the report function must allow to report over the defined parts.

  Graphical representation of reports is very desirable.

So the manipulator contains all functions to update and retrieve the problem data.

The second group of functions is concerned with the computation of evaluations.

Evaluator functions

Given the uncontrollable parameters and a decision rule the effects are calculated. Sometimes these evaluations are very simple, for instance the length of a route in a vehicle routing model, but they may be very time consuming, for instance in case of a process simulation in a queuing model. Often there are several different evaluation functions defined.

The third group of functions is concerned with the automatic generation of decision rules or actions.

Generator functions

Given the uncontrollable parameters and eventually a part of decision parameters, and a criterion function (defined by evaluation functions) the system must determine actions or decision rules that are "good" with respect to the criterion. Optimization is often intractable and therefore one may choose one of two ways:

a. An approximation of the model by a model that is tractable. The optimal actions for the approximated model are translated to actions for the original model and may loose their optimality property.
b. A brute force enumeration method is used to generate some decisions. In general we are not considering all possibilities and therefore we many not find optimal decision rules. Often heuristics are used to control the generation process, i.e. to reduce the number of

decisions to be considered.

Depth first and breadth first search methodes belong to this class.

The fourth group of functions is concerned with learning aspects.

Adaptor functions

These functions are used to derive knowledge from scenario's. The comparisons between scenario's and their realization, which is also a form of learning, is in our set-up a part of the analysis function of an information system. The knowledge that can be obtained is:

- Quality characteristics: by evaluation of a set of scenario's that are accepted by the decision makers, characteristics may be computed and may be used to update distributions of these characteristics. For instance in route planning we may compute the integral of the tonnage over de route or the ratio of the total tonnage to be shipped and the number of carriers we need to do it. Such characteristics may be used by the generator to accept or reject an alternative.
- Rules of thumb for the searching of alternatives, such as "the farest locations first". Such rules may be found in accepted scenario's and used afterwards by the generator.

Finally there is a group of functions to control the dss.

Control functions

This group consists of functions for the creating and updating of the meta data:

- database schema
- evaluation functions
- generator.

And further it contains the dialog-functions to activate the functions of the other groups.

A large class of dss for planning purposes can be formulated as a dynamic programming model. In the next section we will see an example of this formulation. Here we give a general formulation and sketch possibilities for the generator. (cf. Denardo (1982)).

$\langle S, A, C, e, T \rangle$

S : (free) database state space (cf. 6.3)

A : action space, set of possible updates

C : set of constraints over S

e : evaluation function, used as criterion

T : state transisiton mechanism

$T \in S \times A \rightarrow S$

so T describes the possible updates

The optimal value function v satisfies the following equation:

$$v(s, B) = \max \{e(s), \max \{v(T(s, a), B \setminus \{a\}) \mid \forall c \in C : c(T(s, a)) \wedge a \in B\}\}, \quad s \in S, B \subset A.$$

Hence if we stop we get e(s) and if we continue we get the best value that belongs to an updated state. Note that the evaluation function e must be defined for all states. This means for route planning that we have to evaluate unfinished routes also, for instance by a penalty.
If A is finite then the recursion terminates, however in general it is not feasible to use a straight forward dynamic programming algorithm because the size of s is too large.
Then we may base our generator on a relaxed dynamic programming model by

mapping the state and action spaces to sets with lower cardinality. Or we may use a tree search method that searches a tree that starts in (s_0, A), the starting state and the whole action space.

An example of such a tree search method is the following:

- reserve space for $k + 2\ell$ states; the k places are to store the best states so far and the 2ℓ places are to store two levels of the tree.
- Start with the root (s_0, A) and generate sons. Keep the best ℓ sons (so we need only ℓ places).
- For each ℓ states on a level i we search sons and we keep the best ℓ sons of all ℓ states stored in level i.
- Each time we find a record, i.e. a state s with value $e(s)$ higher than all the state values in the k-state places then we replace a state with lowest $e(s)$ by this state.
- As long as improvements are found the generator continues, and if no improvement is possible it stops. Then the decision maker may decide how to proceed for instance by starting in one of the "record" states or by continuation with a state with a lower value.
- In the search process rules may be used; they can be of the form: if <some condition> then update <some action>.

Generators of this kind are very robust, which means that they can be used in a wide class of decision situations so if the object system changes we do not have to change our generator but only some constraints in C or the database schema. Robustness conflicts sometimes with efficiency. However dss that only searches more alternatives than a decision maker in the same time are already helpfull. For this kind of heuristic approaches we refer to Pearl (198).

5. 5. Development of decision support systems

Traditionally decision support systems were developed by operations research specialists, who were focussed on mathematial models and optimization. Because the decision situation is often too complex to describe in one mathematical model these specialists were inclined to neglect all kind of details to fit reality into their models. Therefore decision support systems were often not used in the way they were designed for. Often decision support systems were only used as games to be played by decision makers to learn about the decision process.

The development of a decision support system has to start with a very detailed analysis of the decision situation. In this phase the analyst has to look for all the information from the registration subsystem that can be useful for the decision making and he has to define the set of possible actions and decision rules, without bothering about the way these decision (rule)s are obtained.

The structure of scenario's has to be defined in this phase. We call it the decision analysis phase. Then two different activities may be started in parallel:

- design of the manipulator functions
- development of mathematical models.

The manipulator can be designed along the usual lines of information system development. It can be tested by the decision makers even if no part of the generator is implemented.

The parallel process of construction of mathematical models is in fact the development of the analysis function of the information system. The algorithms for the models have to be tested by real-world data or simulations. If the manipulator is available at the time of testing of the models it can be used as a prototype vehicle.

If the models are accepted the generator can be implemented in the decision support system. If the system is in the phase of user-tests the development of the adaptor can be started.

In fig. 6.6 we summarize the phases:

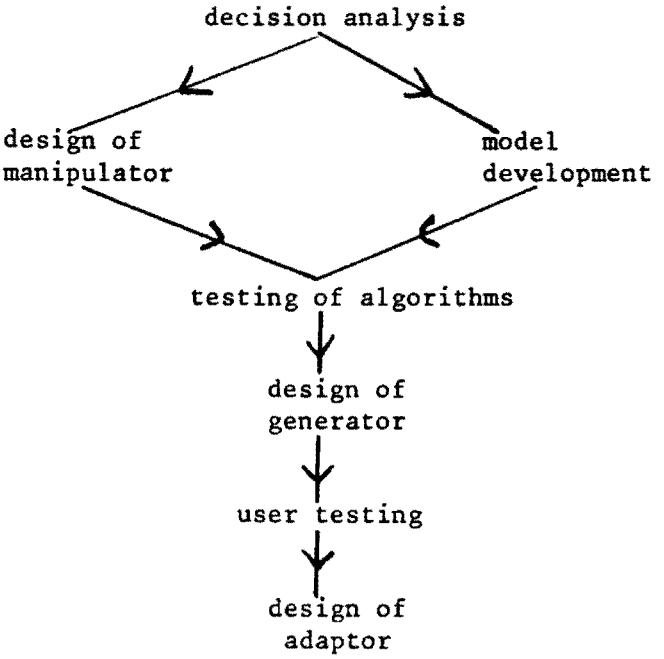


fig. 5.6

6. Case: Inland navigation companies

6.1 Company

The dss we shall discuss is a concise version of a really existing system. The dss is in the first place an instrument of the operational management not of the strategic management.

This dss has been developed for an inland navigation company that is transporting bulk cargo by means of a combination of floating carriers pushed by a so called pusher tug. The company owns ca. eighty carriers, four great pusher tugs and some special pusher tubs meant for use in the ports only. Each great pusher tug is allowed to handle at most four carriers (nowadays the official number is six).
The working area of the company covers the Rhine and its branches, the waterroutes to Antwerpen and Amsterdam and the port of Rotterdam.
For each tranport the company receives a separate order not long before the starting time. This means that the planning of the shiproutes has to be realised in a short time. The dss which we shall discuss is specially meant to this type of planning.

6.2 Information system

The information system of the company has four main functions: the registration and reporting of all relevant data related to the business process itself. The discussed decision supporting facilities are part of the control function of the information system. In this case these facilities have a strong relationship with the other functions.

The registration function stores (among others)

- not-position related data of boats and carriers i.e. technical specifications;

- position related data of boats and carriers, historical and actual both;
- already planned shiproutes;
- geographical data on waterroutes and ports;
- data on the waterlevels and streamspeeds of the waterroutes;
- contracts;
- prices for each type of activity;
- damage-messages.

The reporting function delivers (among others)
- summaries of available carriers;
- invoices (these may require complicated calculations);
- statistics of the use of equipment.

The analysis function of the information system produces
- patterns of supply of cargo;
- model of the behaviour of a ship in the river;
- model of the fuel consumption.

The control function has two goals:
- the planning of the transport capacity (boats and carriers can be hired or rented, mostly over long periods);
- the planning of the use of transport capacity: the shiproute planning.

In the next paragraph we restrict ourselves to the shiproute planning.

6.3 Route planning

The firstpart of the route planning consists of the assignments of carriers to transport orders, and subsequently to the pusher tugs. The result is

a schema of loading and discharging ports for each pusher tug. Travelling along these ports the pusher tug will pass points with time constraints, such as ports with restricted arrival and/or departure times, bridges and sluices.
So the second part of the route planning concerns the determination of the departure times and the velocities in the passing points for each ship-route.

From now on in this case we only consider the second planning problem. The goal of this planning problem is to meet the commercial commitments with as less as possible fuelcost or traveltime. The fuelcost form a substantial part of the costprice and a shorter traveltime means a better utilisation of the transportcapacity. However these goals may be conflicting. The restrictions of this optimization problem consist of the commercial commitments. Just the former restriction can not be defined hard, so the planner will try to manipulate them.
After inputting the ship, its route, the time limits of concern and the cargo, the planner may perform several optimizations with help of the dss :

- minimalization of the traveltime given the departure time in the starting point
- minimalization of the traveltime given the arrival time in the end-point;
- minimalization of the fuel consumption with a free arrival time but a given departure time;
- minimalization of the fuelconsumption with a free departure time but a given last arrival time, related to the effect of the tide on the water height of the rivers down stream;
- minimalization of the fuelconsumption with a given departure time and a last allowed arrival time.

The optimization model is a dynamic programma, see figure 1.

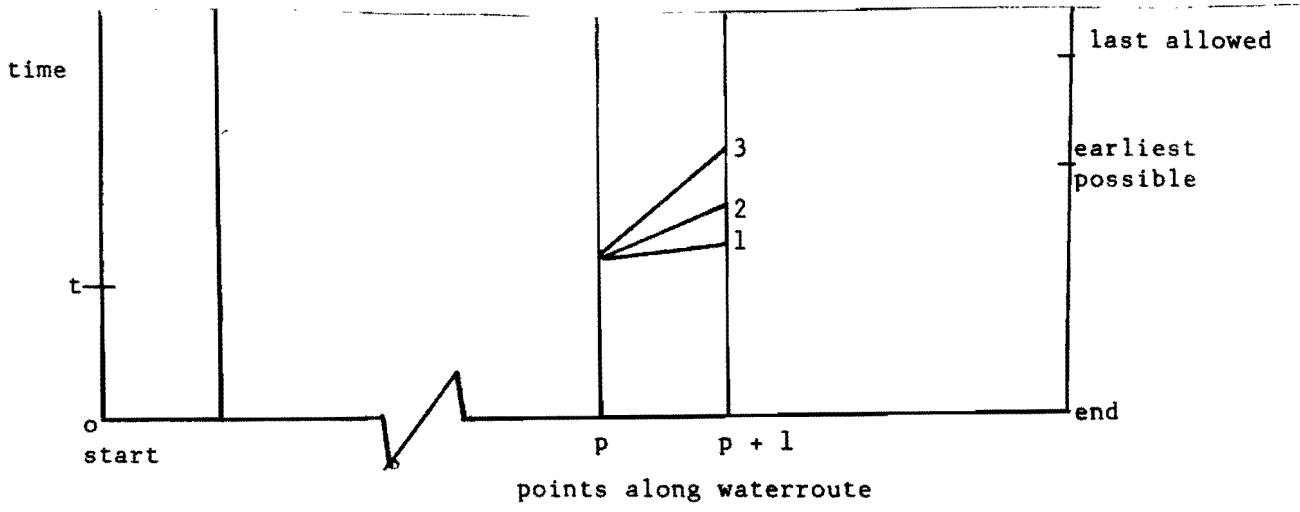


fig. 6.1

The dicision variables of the model are the departure time t in point p and the number of revolutions of the propeller per minute r , which is held constant between the subsequent points of the shiproute. With the waterbehaviour model, the velocity model and the fuelconsumption model, coming from the analysis function of the information system the fuelconsumption and traveltime can be determined. The waterlevel and stream velocity in traject from p to p+1 can be composed with a model of the water behaviour on the waterroute. Together with the choosen number of propeller revolvings per minute the speed of the boat can be determined by the velocity model and subsequently the duration of the traject from p to p+1 and given the departure time in p the arrivaltime in p+1.

From the duration and the choosen rpm the fuelconsumption can be composed by the fuel consumption model.

With a standardtechnique to solve dynamic programs each above-mentioned optimum can be determined (see [Denardo, 1982]).

In the next paragraphs we deal with the datamodel of this decision problem, the behaviour model, the optimization model and a generalization of the used solution method.

6.4 Data model

The data relevant to the dss can be clustered into four groups: waterroutes, equipment, orders and shiproutes.
Firstly, we shall explain the used notions, secondly, we shall describe the data in a functional data model and, thirdly, we shall work out some restriction belonging to the data model.

Waterroutes

The rivers and canals used by the pusher tugs form a network of waterroutes. A waterroute consists of several segments and a segment in turn is a sequence of checkpoints. In a checkpoint the waterheight and streamingspeed is known and in such a point the captain may adjust the number of revolvings per minute.
Some of the checkpoints have time-windows: for instance, ports with predetermined time slots for arrivals or departures. We call these checkpoints nodes and a segment is a sequence of checkpoints enclosed by a starting and an ending node.

Equipment

The only relevant data of the equipment to the dss are the push boats, the carriers and their restpoints and restperiods. The restpoints are always nodes. Equipment not involved in a trip is in a restpoint.

Orders

An order is an instruction to transport a certain amount of tons of a commodity from place A to place B .

Shiproute

A shiproute is a sequence of shiproute segments and each of these segments is referring to a waterroute segment. A shiproute may be cyclical.

Functional data schema:

The formal definition of the functional database schema is, sketched as follows:

C : {level, time, waterlevel, ...}

Set of category indices;

F : {1, 2, 3, ..., 41}

Set of functionclass indices;

D : {(1, waterlevel), (2, waterlevel), 3, waterlevel) (4, rpm setting) ...}

This function determines for each functionclass a category called the domain category;

R : {(1, level), (2, time), (3, checkpoint), ...}

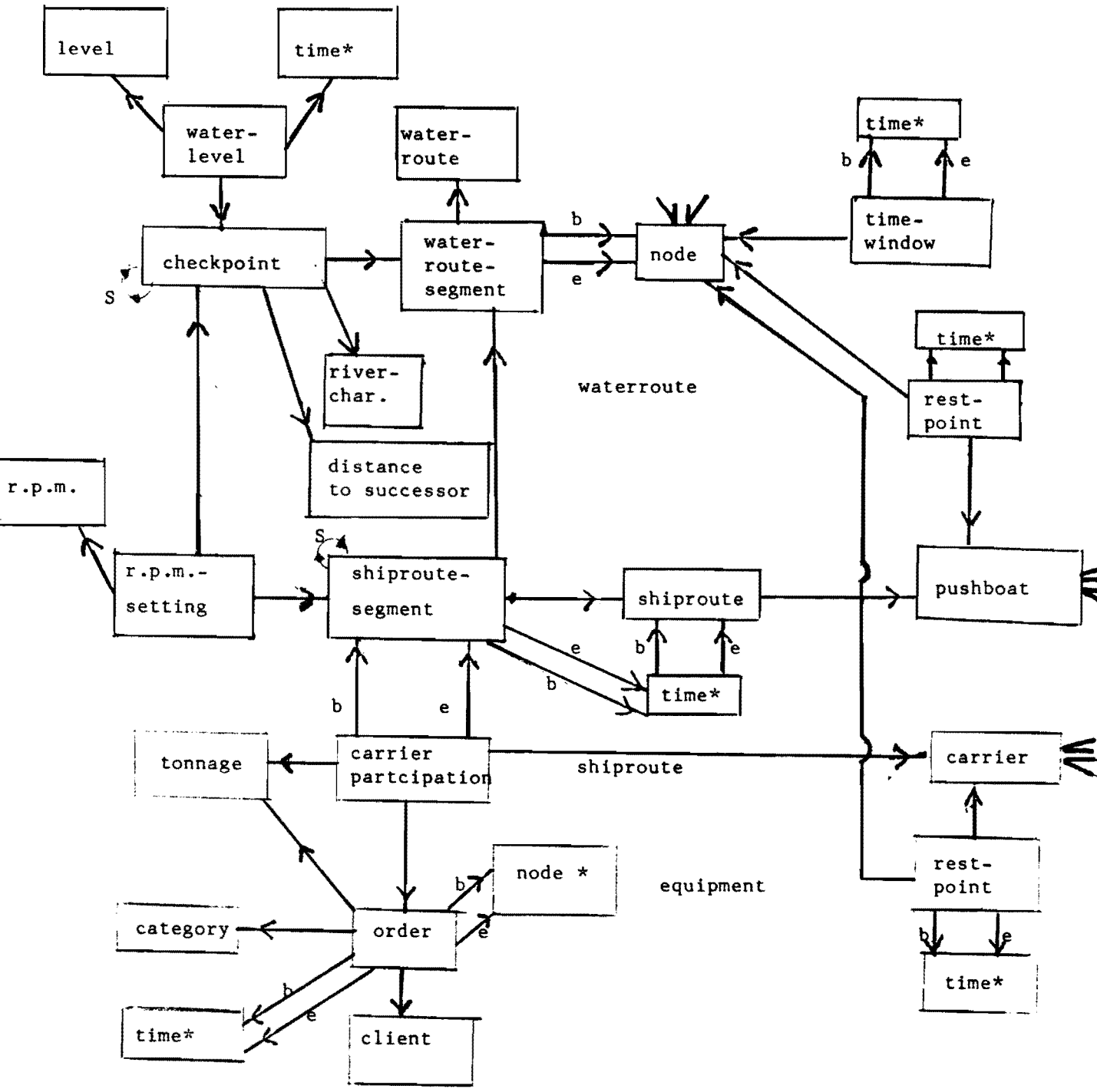
This function determines for each functionclass a category named range category;

V : {(1, {0, 1, 2, ...}), (2, {<m, h, d> | m ∈{d, 1, ..., 59}, h ∈ {0, 1, ... 23}, d ∈ {1, 2, ..., 365}), ...}

This set valued function V assigns to each category the set of representations of all possible objects that belong to this category.

For a graphical representation of the data model we refer to figure 6.2.

Data schema



order

S: successor

*: several instances

b: begin

e: end

: several atrributes

note that we did not named the functionclasses.

Some examples of restrictions in natural language

a. If a carrier or a push boat participates to a shiroute then they are not allowed to be in a restpoint.

b. The time of departure of a shiproute segment is not allowed to fall within the time-window belonging to the node of the corresponding waterroute segment.

c. The waterroute segments have to follow up each other: i.e. endnode of a segment = beginnode of the following segment.

d. The time intervals of the shiproute segments are not allowed to overlap each other, must have the same order as the corresponding shiproute segments and must suit within the total shiproute timeslots.

e. The tonnages of the carriers belonging to one order have to sum up to the tonnage of the order.

f. The distance from checkpoint to checkpoint, divided through the velocity, summed over a waterroute segment has to fall within the corresponding shiproute timeslot.

Restriction f. can be formalized as follows:

$$(\underline{A}\ srs \in \text{Shiproute segments} : f25(srs) + (\underline{S}cp \in \text{checkpoints} : I(f9(cp) = f11(srs)) \cdot f7(cp)/\text{velocity}(cp, srs) \leqslant f24(srs)))$$

Note:

Velocity (cp, srs) is a function that composes the absolute velocity of the pushing boat in checkpoint cp , belonging to shiproute segment srs . This function is based on the speed model of a push boat, which we shall discuss in next paragraph. A denotes the universal quantor, S the summation quantor; I denotes the characteristic function that assigns 1 to true and 0 to false.

With these restrictions we describe infact the feasible process-paths of the objectsystem, i.e. the relevant part of the company.

6.5 The behaviour models

In this paragraph we shall discuss three models: the water behaviour model, the velocity model and the fuelconsumption model. These models enable us to compose criterion functions in the optimization model, the total travel-time and the fuelconsumption.

Water behaviour model

The behaviour of the water in the rivers and canals has two features: water-level and streaming velocity. Predictions on the waterlevel are computed with a regression model applied on measurements of the waterlevel upstream and the volume of the rainfall.

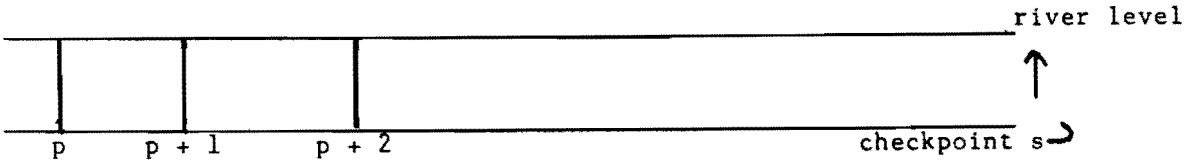


fig. 6.3

A simplified version of the regression equation of the waterlevel $W_{p,t}$ in checkpoint p on time t is:

$$W_{p,t} = \sum_{q>p} \sum_{k=1}^{5} \alpha_{q,k} \cdot W_{q,t-k}$$

where $\alpha_{q,k}$ are the regression coefficients and $W_{q,s}$ the measurements of waterheight at time s.

So, waterlevel $W_{p,t}$ is supposed to be dependent on the waterlevels upstream in the five periods preceding t. With this equation the waterlevel can be predicted in every checkpoint and time-point within the planning horizon. Note that the predictions for longer periods become worse because they are

based on less observations.

This model concerns the upstream area's of the rivers and canals. However in the downstream area of the rivers is the effect of the tide present. So, the model has to be adjusted.

Both the regression coefficients $\alpha_{q,k}$ as the measured waterlevels $W_{q,t-k}$ are issued by a government agency, charged with the management of the public waterarea's in the Netherlands.

Every day the company receives the measured waterlevels of the relevant waterroutes and stores these data in the database.

The stream velocity at some moment in checkpoint p is determined by the waterlevel wℓ at that moment and the form characteristics. (see figure 6.4) This function is tabulated.

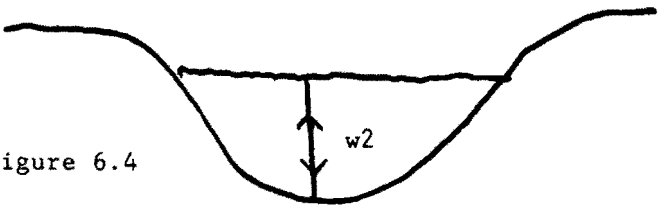


figure 6.4

Cross section in checkpoint p

The velocity model

The absolute speed of a push boat is determined by the next equation:

$$vs(p,t,r) = ss(p,t) + f(cr(p,t),\ w(p,t),\ r)$$

where vs : absolute velocity of the boat

ss : stream velocity of the river

cr : charging rate

r : revolvings per minute

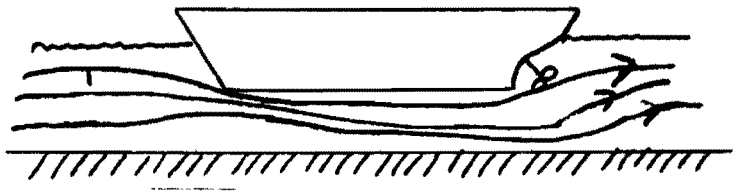
wℓ : waterlevel

p : checkpoint

t : time

The function f determines the velocity of the ship related to the river. This function has some properties in connection with the hydrodynamics of a ship travelling on a river (see figure 6.5).

fig. 6.5



The function f is increasing in $w\ell$ and r , and decreasing in cr . Intuïtively one may consider the dependency in $w\ell$ as follows: the less water there is under the ship the more difficult it is to "pump" the water under the ship backwards.

The fuelconsumption model

The fuelconsumption of a pushing boat can be composed with the next formula.

$$fc = c_1 \cdot r^{c_2}$$

where fc : fuel consumption in a time unit

r : number of revolvings in a time unit

c_1 : constant

c_2 : constant ≈ 3

Note only the last two models are produced by the analysisfunction of the informationsystem of the company. Periodically these models have to be adjusted.

6.6 The optimization model

In the optimization model we distinguish a statespace S , an actionspace A and a transition mechanism T .

We define the state as the trajectory of a shiproute until some moment i.e. a sequence <checkpoint, time, rpm> (see figure 6.6). We consider the rpm as the rpm on arrival in a checkpoint.

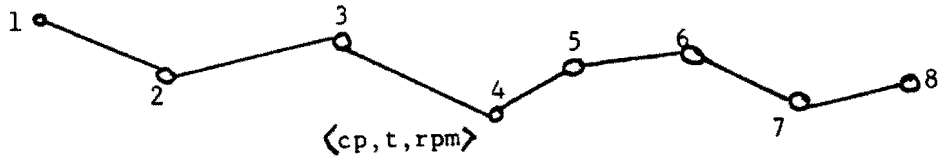


figure 6.6

The actionspace A has been defined as the cartesian product of the set of all checkpoints of the concerning shiproute and the set of all possible rpm on arrival in the next checkpoint.

We represent the transmission mechanism as a function $T : S \times A \to S$, with

$s : \langle p_1, t_1, r_1 \rangle \ldots \langle p_n, t_n, r_n \rangle$

$a : \langle p_{n+1}, r_{n+1} \rangle$

$T(s,a) : \langle s, \langle p_{n+1}, t_n + \frac{\text{distance } (p_n, p_{n+1})}{vs(p_n, t_n, r_{n+1})} , r_{n+1} \rangle\rangle$

where vs(p,t) the velocity of the boat is (see last paragraph).

Remark that only actions with p_{n+1} are allowed; a part of a shiproute has to be linked as well.

For optimization we do not need the whole states.

This optimization model is a dynamic program with as optimality equations

$$V_1(p,t) = \min_r \{c_1 r^{c_2} \cdot \frac{dist.(p, S(p))}{vs(p, t, r)} + V_1(S(p), t + \frac{dist.(p, S(p))}{vs(p, t, r)})\}$$

related to the fuelcost where S(p) is the successor of p on the route (cf. figure 6.1).

Note that in the optimality equation we only use the last checkpoint and time of the state (this is a trivial case of state space relaxation).

6.7 Concluding remarks

1. There is a considerable interaction between the dss and the registration function of the information system:
   - producing a shiproute;
   - selection of push boats and carriers;
   - record the results of the planning.
2. There is a considerable interaction between the "behaviour" models and the optimization model.
3. The dss performs only a few functions in the decisionprocess. Not present are:
   - linking shiproutes;
   - production of parameters intended to the capacity planning.
4. From the analysis of the decisionmodel it turned out which supplemental information had to be recorded in order to determine function f .

LITERATURE

Aho, A.V., J.E. Hopcroft and J.D. Ullmann

Data structures and algorithms, Addison-Wesley 1983.

Benders, J.F. and J.A.E.E. van Nunen

A property of assignmenttype mixed integer linear programming problems, Operations Research Letters 1983.

Christofides, N.

Vehicle routing problems in: The travelling sales man problem; a guided four in combinatorial problems, E.L. Lawler, J.G. Lenstra A.H.G. Rinnooy Kan, B.D. Shmoys, Wiley 1985.

Date, C.J.

An introduction to database systems, Addison-Wesley 1983, vol. II.

Denardo, E.V.

Dynamic programming: models and applications, Prentice-Hall, 1982.

Garey, M.R. and D.S. Johnson

Computers and intractability; a guide to the theory of NP-completeness. W. Freeman and Company, San Francisco 1979.

Gass, S.I.

Linear programming, methods and applications, McGraw-Hill Book Company 1969.

Kleinrock, L.

Queuing systems, vol. I & II, J. Wiley & sons 1975, 1976

Magnanti, T.L. and B.L. Golden

Transporation Planning: Network models and their implementation, in: A.C. Hax (ed.) Studies in operations management, North Holland 1978.

Pearl, J.

Heuristics: intelligent search strategies for computer problem solving, Addison Wesley 1984.

Tanenbaum, A.

Computer networks, Prentice-Hall 1981.

Tsichritzis, D.C. & F.H. Lochovsky

Database Management systems, Academic Press, 1977.