HEURISTIC ALLOCATION IN CENTRALIZED DESPATCHING OF LOG TRUCKS

by

ROBERT JAMES MCCORMACK

Thesis submitted for the degree
of Master of Science at the
Australian National University
1983

Except where otherwise acknowledged this thesis is the author's original work .

ABSTRACT

Centralized despatch systems appear to offer the opportunity for increased efficiency in log hauling operations where large fleets of trucks are hauling to few locations .

A goal of work equity as the basis for the allocation of work between trucks is proposed as one that could be acceptable to the groups typically comprising log transport systems in Australia .

Recent rapid decline in the price of computer equipment coupled with substantial increase in power and flexibility make feasible the economic development of computer aided decision environments for problems of control in industrial systems , such as those presented by log truck despatching .

The development of a computer based heuristic allocation algorithm which could provide the basis for the development of such a despatch system is described .

Testing of a 'prototype' single pass and a further developed 'three pass' heuristic allocation algorithm indicated considerable improvements in work equity could be achieved as compared to a random allocation of trips.

A generalized computer simulation model of truck fleet operations was developed to provide a testing facility for evaluation of the allocation system .

Tests of the allocation procedure using the simulation model indicated that the use of computer aided centralized despatch appeared to allow the operation of the fleet with most of the trucks close to a specified level of utilization . Thus , the operation of a smaller , more highly utilized fleet , with consequent economic advantages appears possible , without significant increase in the number of trucks exceeding target daylength .

Several important attributes of system performance were identified and methods of investigation based on the use of the simulation model testbed were developed .

Both the allocation system and the simulation model provide the basis for ongoing investigation of transport system dynamics under alternative methods of fleet management .

ACKNOWLEDGEMENTS

The material support and assistance given by members of the logging industry at Eden , through ELITT and from HARRIS DAISHOWA AUSTRALIA Pty Ltd and that of my employer , CSIRO Division of Forest Research is gratefully acknowleged for without it the field study required to support the project would have been very difficult .

The interest and encouragement of people in these groups and of my supervisors , initially Dr Ian Ferguson , and more recently Don Stodart has been no less important in sustaining the effort required and deserves equal acknowlegement . Particularly , the meticulous evaluation by Don Stodart of the work's many drafts and his patient advice are most warmly acknowledged .

Access to two important technical facilities , the Australian National University Computer Centre's UNIVAC 11/80 , and the Operations Research collections of the Canberra College of Advanced Education library made possible a project otherwise difficult in the prevailing nontechnological environment .

An unspoken debt is also owed to those within the O.R. and Computer Science community whose concern with applied problems and the achievement of a practical improvement in operations provided welcome inspiration .

The work would not have been possible without two decades of investment in education . The example and support of my parents and grandparents deserves special comment . Their individual contributions included a timeless patience with the process of learning , perspectives of unusual practicality and a steadfast belief in cultivation of an enthusiasm for enquiry and knowledge . In measure , this work is built on their foundations .

Most of all , sincere thanks to my wife , who has weathered the many hundreds of hours absences in librarys and computer centres over the several years of the project , retaining the patience to help with the final thorny task of shifting words and redirecting sentences with skilfull criticism and endless proof reading .

To Coralie and Keirin , Thanks

TABLE OF CONTENTS

CHAPTER 2 DEVELOPMENT OF A HEURISTIC TECHNIQUE FOR DESPATCHING A LOG TRUCK FLEET

CHAPTER 3 A SIMULATION MODEL FOR LOG TRUCK FLEETS

CHAPTER 4 APPLICATION OF THE SIMULATION MODEL

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

MANAGEMENT OF LOGGING TRUCK FLEETS

1.0 INTRODUCTION

Road transport of logs from the forest to the mill is a significant harvesting cost and organizing the operations to ensure efficiency should be a continuing goal .

The costs of road transport are partly due to expenditure on fuel , driver's wages , tyres , maintenance etc , but they also reflect the considerable standing costs associated with owning and using logging trucks . The following simple economic model of log truck operation illustrates these costs*.

Value of the truck and trailer	$95,000
Fixed annual charges (insurance, depreciation , interest charge on investment and registration)	$31,800
Annual running costs 80000 kms (fuel, oil,maintenance, tyres)	$32,200
Drivers wages and employment overheads	$15,000
Total annual costs	$79,000

* MACARTHUR , CSIRO personal communication .

Fixed charges comprise about 40% of the total costs and under normal levels of truck usage are incurred almost independently of truck utilization . Increased utilization , expressed as hours of operation or increased distance travelled , is therefore reflected in lower average costs per hour or per kilometre .

The major influences on utilization of logging trucks are the availability of wood to be hauled , the interaction of round trip time with the maximum working hours accepted by the drivers and , of course , the total number of trucks undertaking the haulage task . Central control of trip allocation could influence all of these factors for a particular trip and is an alternative to methods which involve a direct organizational link of the truck to a logging contractor or to a landing .

Central despatch , while common in other industries in Australia and in overseas log truck management , is seldom applied to Australian logging operations .

The despatcher becomes responsible for allocation of work to all the trucks and many of the problems associated with fluctuations in transport requirement can be theoretically overcome . For example ,

1. the round trip time problem may be overcome by assigning a truck a set of trips which better matches the total hauling time to the desired

number of working hours , the despatcher being able to choose from a much larger range of trips than those available to an individual truck driver ;

2. major problems associated with underproduction at one landing could be readily mitigated by reassigning trucks to landings where log production is over target ;

3. some amelioration of even the most difficult problems associated with seasonal or market induced production cutbacks may be possible by large scale transfer of contractors to more distant and possibly more difficult logging conditions ,thus increasing the transport requirement for a given level of log production . The possibility of the management of such large scale relocation arises because of the separation of control of the transport from the logging .

The major problems associated with the introduction of a centralized despatch system are those relating to the much higher levels of management required , with their attendant costs and responsibilities . Thus any proposal to introduce centralized despatching would require convincing evidence of its practical and economic advantages and this would require thorough investigation of the existing system and evaluation of the proposed change .

In this study , procedures are developed for the investigation and evaluation of the introduction of a centralized despatch system for a logging fleet based on a despatcher supported by a micro or minicomputer system . The haulage operation to the chip mill at Eden , New South Wales , is typical in size and structure of a number of large scale operations in Australia and is taken as a practical reference for the study .

1.1 CENTRALIZED DESPATCH SYSTEMS FOR LOGGING TRUCKS

1.1.1 The Despatcher

The central role in any despatch system would be that of the despatcher who would have the task on a continuing basis of sequential assignment of trucks to landings . The transport task would be set in relation to periodic notification of the log transport requirement . The objective in the decision making of the despatcher would be to maximize efficiency as defined by management policies . Major questions that must be answered by management in considering and defining the role of the despatcher are

1. what goals are appropriate ?
2. what are suitable criteria for choice of an efficiency factor to measure the performance of the haulage system ?
3. what procedures should be adopted by the despatcher to maximise this performance ?
4. how effectively would he operate ?

1.1.1.1 Despatching Goals

The selection and setting of goals for a despatcher requires consideration of the interests and attitudes of each of the economic groups associated with a log transport system . There are usually four groups :

1. the Company operating the mill
2. the truck owners
3. the truck drivers
4. the logging contractors .

It must be accepted that each group would seek to improve its economic position , perhaps at the expense of the other groups , at the stage of contract negotiation or even during operations .

The Company point of view is probably the most straightforward . Having accepted a contractor system , their viewpoint is that of the wood buyer and processor making direct payment for services. Long term minimization of the contract price and reliable delivery of the wood are major interests and these extend to a concern for the stability of the logging industry serving the mill and knowledge of both the actual and potential physical and economic performance of the truck fleet . The requirement for economic data is particularly relevant in contract negotiations .

The truck owners have a direct interest in both revenue and cost . Typically , the level of contract rates

are reviewed relatively infrequently and the truck owners would seek the highest rate at such negotiations . Subsequently , the truck owners would be concerned on a day to day basis with utilization of trucks , that is trip allocation , and with the costs of operation . The truck operators are in a position not unusual for contractors , that while revenue is usually earned at 'average' rates , the marginal cost of delivery is only running costs . The return from a marginal load is almost always positive , thus their interest is in maximizing utilization of their trucks .

The major interest of truck drivers is in achieving a personally desirable trade off between additional income as a result of longer working hours and the reduction in their 'own time' and perhaps the inconvenience of irregular working hours . Owner drivers have conflicting interests as owners and as drivers .

The logging contractors have a vested interest in ensuring that road haulage of wood is in relation to the production of the logging gangs , since increased stockpiles at the landing may cause inconvenience and loss of production . Since payment to the logging contractors is usually on the basis of wood delivered to the mill rather than wood delivered to the landing the interest of this group in the performance of a centralized despatch system is direct , to the point that they may be reluctant to relinquish control of the allocation of trucks to the landings .

The operational goals of a centralized despatch system must at least reconcile the interests of the four defined groups .

The diffuse and decentralized system of truck control currently operating may allow individuals or individual firms within these four groups to achieve economic gain through negotiation, organization or efficient endeavour although if viewed from a perspective of the whole transport system , a gain by one firm may be a loss by another. Replacement of such a system by one of centralized control would be accepted more readily by the separate groups if it offered each group some gain , for example increased returns or perhaps increased security of return .

The definition of despatching goals and testing them in relation to the interests of the major groups involved in log haulage to a large mill are of critical importance in considering the introduction of a centralized despatch system and are therefore a major aspect of study .

1.1.1.2 'Equity' Goals

The economic notion of equity (i.e. fairness) provides one goal which could assist acceptance of the centralized despatch system by all the major groups . If a centralized despatch system leads to increased operational efficiency by reduced lost time or reducing the incidence of incompatible trip lengths and gives greater predictability in fleet performance , then the increased

efficiency could be translated into reduced truck numbers and increased utilization for those trucks remaining . Adoption of a principle of equitable distribution of gains from improved efficiency could lead to shared economic improvement between at least the wood buyer , truck owners and truck drivers through contract price negotiations and work conditions . More specifically , a central goal of equitable distribution of transport work to all trucks is likely to be essential to gain the acceptance of the truck owners and drivers .

1.1.2 Despatch Methods

Introduction of a centralized despatch system for a truck fleet serving a major mill would require the development and testing of a practical method for despatching trucks to achieve the goals of the system . There are two aspects to this :

1) Information processing .

   It would be necessary to provide for continuous recording of the information on the log transport requirements for all landings and of the status of each truck in relation to the task assigned to it at any time and its defined cumulative performance .

2) Trip assignment .

   The assignment of a trip to a truck would be based on the recorded information and the despatching goals .

The problems of analysis of the information and consequent trip assignment would seem the more difficult . Maister (1980) reviewed a wide range of despatch systems in the American road transport industry and emphasised the importance of an experienced human despatcher for responding to unusual and changed circumstances . Despite its apparent simplicity , for fleets of realistic size the problem of despatch assignment is very time consuming when solved rigorously by computer (this is discussed further in Chapter 2) and the problem is usually left to a human despatcher . A variety of graphical and other techniques of information presentation , together with clerical and intuitive methods , are commonly used by despatchers to provide despatch schedules .

Accepting that a rigorous solution of the trip assignment problem by a computer program would be too time consuming , development of computer aided despatch systems seems to require , therefore , the programming of the computer to take over some of the information processing tasks and to provide partial or baseline solutions as a basis for revision or adaption by a human despatcher to suit the changing 'real world' conditions . Both developments would permit the decision maker to give more attention to the more complex tasks .

1.2 THE THESIS

Centralized despatch systems appear to offer the opportunity for increasing efficiency in some log hauling

operations . Methods to examine this assumption were developed . The central objective became the development of a trip assignment algorithm and the testing and evaluation of its application as a basic element in a computer aided despatch system .

Field testing of an assignment algorithm on an operational fleet is unlikely to be seen as acceptable as a first trial of a proposed system and a simulation model of a truck haulage system was seen as a prerequisite for testing and evaluating a centralized despatch system . It was recognised that the data collection effort required to develop a model with detailed predictive capacity for a specific fleet was likely to be beyond the available field study resources , it was believed that a simpler study capturing the basic operating characteristics and system dynamics would be sufficient for the development of a general truck fleet model to allow testing of the allocation procedures . Thus there were three stages to the study .

1. Development and testing of an assignment algorithm for a centralized despatch system .

2. Investigation , description and implementation of a simulation model of a synthesised truck haulage operation .

3. Testing of the performance of the assignment algorithm by application to the simulation model.

Observations of the logging operations associated with the chip mill at Eden , New South Wales , indicated definitive consideration of a centralized despatch system as a research project . The wood chip company , Harris Daishowa Australia , the New South Wales Forestry Commission and logging contractors through a local Logging Comittee , expressed interest in such an investigation . The log haulage system at Eden was therefore used as a practical reference for synthesising the truck fleet simulation model .

1.3 LOGGING OPERATIONS AT EDEN

1.3.1 Description of Operations

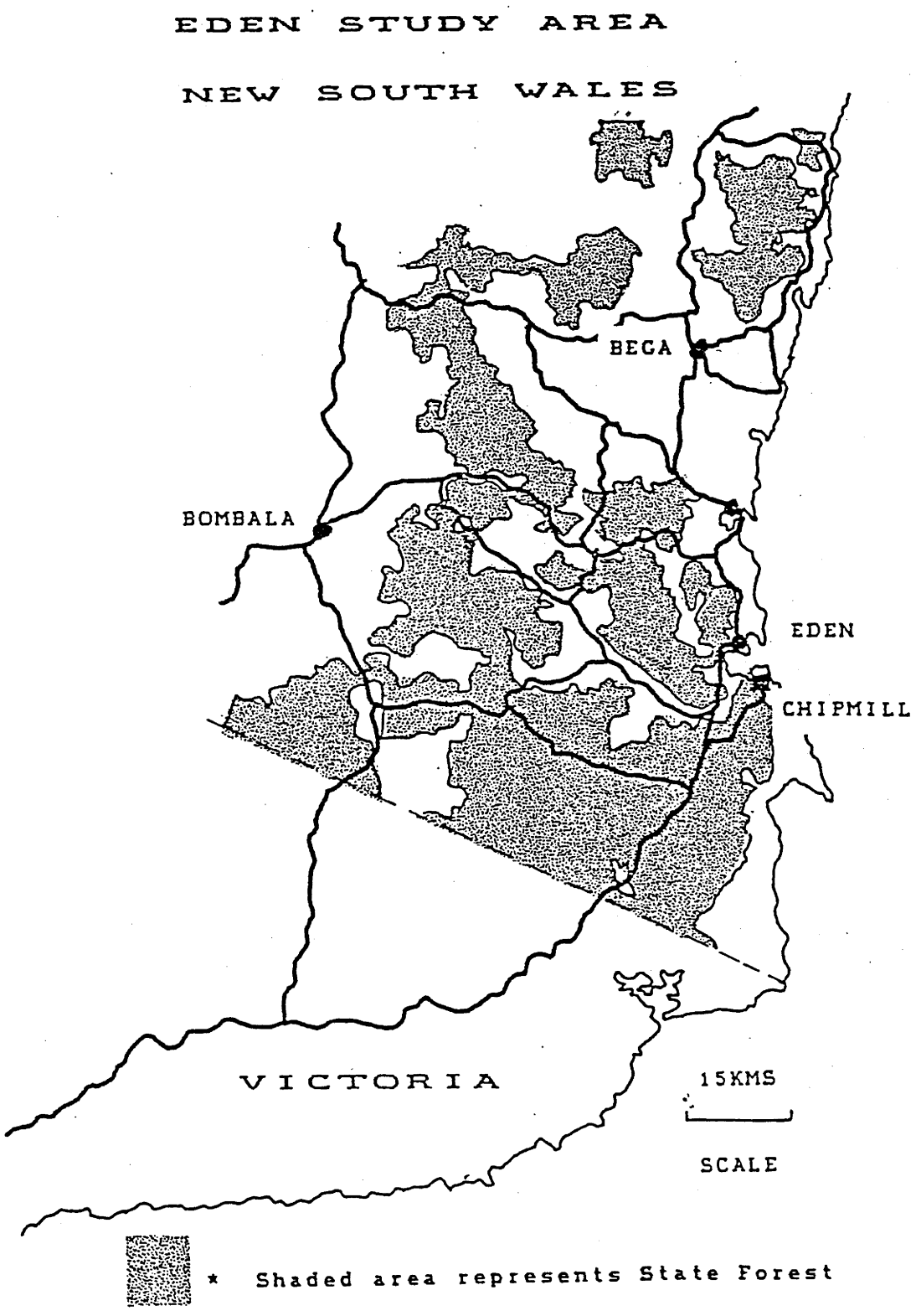
Most of the logging is on State forest land and the management of these forests is the responsibility of the New South Wales Forestry Commission . The Commission has contracted to provide for the delivery of 550,000 tonnes of logs per annum to Harris Daishowa Australia . The remainder of the pulpwood for chipping comes from privately owned forests in both New South Wales and Victoria . These operations are often at large distances from the mill . Geographic information on the pulpwood harvesting area is shown in Figure 1.1 .

The logging operations are based on a contract system. About twenty principal contractors are responsible for the harvesting and delivery of pulpwood from the forest sites allocated to the contractors by the company .

An initial investigation was conducted in December 1979 to determine the operating characteristics of the logging system and to provide a framework to formulate the structure of the simulation model . Further studies were undertaken in March and July 1980 to obtain more data on bush loading and truck travel times . At that time some thirty five logging gangs were operating at separate forest landings . The following brief descriptions of the operations are based on observations made during the field studies .

1.3.1.1 Bush Operations

Log skidding is usually done with large crawler tractors or rubber tyred skidders . Logs exceeding the maximum lengths permitted on the log trucks are crosscut on the landing . At the time of the study , most logs were debarked on the landings .


EDEN STUDY AREA
NEW SOUTH WALES
BEGA
BOMBALA
EDEN
CHIPMILL
VICTORIA
15KMS
SCALE
* Shaded area represents State Forest

Production of logs to be transported varied from day to day . There are also longer term seasonal and market induced changes in production . Short term variation may result from factors such as adverse weather , local variation of logging difficulty or skidding distance, machine breakdown or crew absenteeism and was countered to some extent by stockpiling . Most crews produced between four and six loads a day .

Stockpiling was usually restricted by the need to minimize landing size for silvicultural reasons and avoid delays when the equipment was ready to move to the next landing where the loader was required to handle logs. Stockpiles of three truck loads or less were common and even minor restrictions on production sometimes exhausted stockpiles and resulted in delays to trucks at the landing.

In the longer term production will vary with the overall characteristics of the forest and terrain as a crew moves from compartment to compartment and it must be accepted that even a small reduction in production rate could prevent a logging crew 'keeping ahead' and maintaining its stockpile with consequential delays to trucks tied to that landing .

1.3.1.2 Road Network

Almost all haulage uses some part of the national highway network and therefore loads are resticted to lim-

its imposed by the highway authorities. A large network of roads has so far been constructed or upgraded for the woodchip operations . Roads within the forest zones are high standard gravelled 'trunk' roads and 'spur' roads to provide access to each compartment . Within compartments, roading is restricted to tracks cleared by logging contractors to obtain increased efficiency by locating the landing at other than on the compartment access roads . Access difficulties in wet weather usually arise from a failure of the compartment tracks .

1.3.1.3 Trucking operations

A fleet of over 100 trucks was licenced by the Company to deliver pulpwood to the mill . An additional twenty trucks delivered chipped sawmill waste .

Sawlogs produced during the harvesting operations are delivered to sawmills in the area and constitute about 10% of the logs harvested . Sawlog haulage was excluded from this study . Chip haulage was also excluded because the trucks used cannot be readily interchanged with the pulpwood fleet . Interaction between the pulpwood fleet and the chip haulage trucks is limited and confined to queueing at the weighbridge as separate terminal facilities are used at the mill.

Trucks used to haul the pulpwood are almost exclusively 6 by 4 axle configuration and predominantly in the engine power range 200 -250 kW . A mixture of pole jink-

ers and rigid frame semi-trailers was in use . Both types usually had two axles .

Round trip times were relatively long , typically between three and four hours . Thus the choice between two and three trips a day was between about an eight hour or twelve hour day . As a consequence there was a set of landings where an additional trip may not be undertaken , although part of the working day remained , representing 'under utilization ' . There was also another set of landings where an additional trip would be undertaken , although the resulting working day would exceed that preferred by the driver .

Overtime was usually paid to the drivers , often on a regular basis .

1.3.1.4 Terminal facilities

Trucks delivering wood to the mill stopped at the gatehouse where the load was inspected and the delivery docket endorsed if the load was accepted . The trucks were then weighed on the inwards weighbridge before proceeding to the unloading apron located between the chipper infeed deck and the log storage yard .

The unloaders at the mill had sufficient capacity to lift off a full truckload of longlength pulpwood in one grab . Two unloaders were in service at the time of the study and these were backed up by several smaller front

end loaders . Normally only one unloader was in use at any one time . A jinker loading facility was provided .

Empty trucks were weighed on a separate outwards weighbridge and the completed delivery docket accepted at by the weighbridge operator on the way out of the mill .

A service centre supplying fuel and tyres was located adjacent to the mill .

1.4 ORGANIZATION OF THE LOG HAULING OPERATIONS AT EDEN

1.4.1 Organization at the time of the field studies

Log delivery was the responsibility of the logging contractors and the following groupings were based essentially on classification of these contractors .

1.4.1.1 Large multi-gang contractors

This group comprised several contractors each with a number of logging gangs and/or haulage interests . These contractors overcame , to a limited extent , the daylength / round trip problem associated with log haulage by swapping trucks around to overcome short term variation in the rate of supply of logs to landings which results from , for example , a logging crew producing at a higher rate . No ready solution was evident for utilization problems arising from a long term change in the contractors' total

haulage requirement under current fleet management arrangements .

1.4.1.2 Single gang contractors

These contractors operated one gang and owned their own trucks which were effectively tied to the one gang . The usual practice of these contractors to reduce the impact of the problems arising from short term fluctuations in log production was to stockpile . However , some also used trucking subcontractors to carry out some of the log hauling and the burden of adjusting to fluctuations in log production , particularly in the longer term , may have been placed upon the subcontractors .

1.4.1.3 Trucking subcontractors

These are mostly owner drivers and about one quarter of the truck fleet was controlled by them . Typically , they formed longer term associations with particular logging contractors and their trucks often worked in very similar ways to trucks owned by the logging contractor . Adjustment to the hauling operations consequent upon production fluctuations seemed to be undertaken by both the trucks owned by contractors and those owned by subcontractors . However when logging contractors shifted to a new area with very different round trip times , subcontractors often lost employment with one logging contractor and had to find work with another . This realignment provides the

principal means of longer term adjustment within the fleet to changes in the total haulage requirement of the contractors .

There was also a quasi market in 'spot loads' with subcontractors making special trips at the request of a logging contractor providing a facility for short term adjustment . Agreements were informal and appeared to depend on the subcontractors' network of personal contacts, but financial arrangements were facilitated by the direct payment of the subcontractors at agreed rates by the Company .

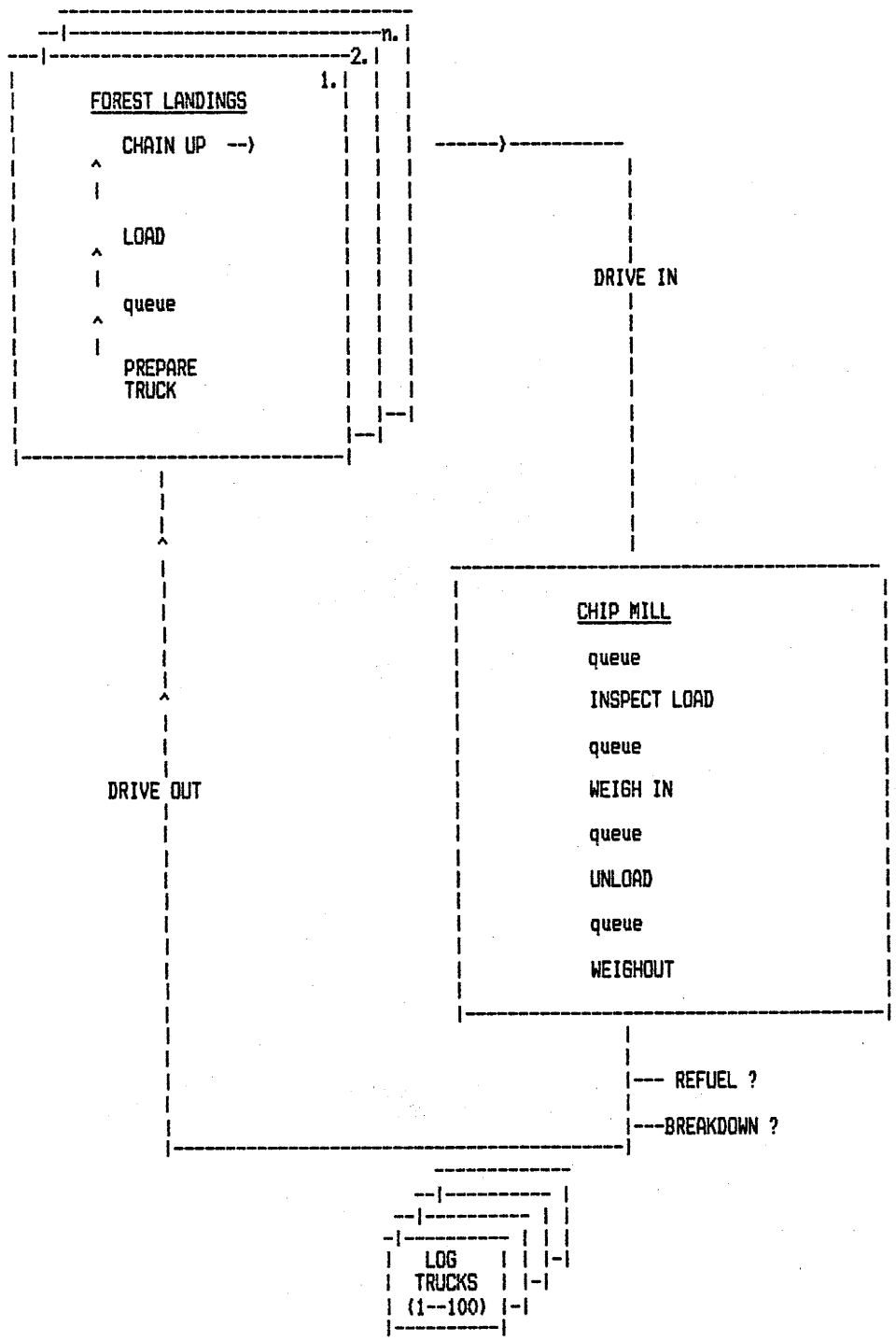
1.5 CONCEPTUAL MODEL FOR THE OPERATION OF A LOGGING TRUCK FLEET

The log haulage fleet associated with the logging operations at Eden formed the framework for conceptual description of a simulation model of the structure and performance of a log truck fleet . The model is illustrated in Figure 1.2 .

1.6 THE STUDY OUTLINE

The introduction of a centralized despatch system based on a 'Computer aided despatcher' requires a programmed truck allocation procedure which in turn requires an algorithm . The development and testing of a heuristic allocation algorithm is described in Chapter 2.

FIGURE 1.2 SCHEMATIC MODEL OF LOG TRANSPORT SYSTEM



Trials on an operational log truck fleet to evaluate the likely field performance of proposed heuristic algorithms are unlikely to be permitted where centralized despatch is not already practised because of the disruption that would be involved .

Development of a simulation model of a log truck fleet would provide for evaluation studies without such disruption . However , there are particular philosophical and technical difficulties associated with simulation models of logging systems .

These issues are discussed in Chapter 3 together with the development of a truck fleet model . The collection and development of a suitable set of input data for the model is described in Chapter 4 , together with the model's acceptance testing .

An evaluation of the performance of the heuristic allocation system using the simulation model is presented in Chapter 5 . The study is reviewed and the conclusions summarized and discussed in Chapter 6 .

CHAPTER 2

DEVELOPMENT OF A HEURISTIC TECHNIQUE FOR DESPATCHING A LOG TRUCK FLEET

2.0 SCHEDULING IN INDUSTRIAL SYSTEMS

Scheduling of work in industrial systems is a very common task and a considerable amount of operational research has been directed toward the solution of the problems in such areas as machine shops , freight despatch and timetabling for aircraft , buses , school classes and work rosters (Elion , 1978) .

In general the truck despatch problem consists of assigning a set of n trips to a usually smaller set of m tasks . It is similar to the 'n independent job , m parallel machine' problem which is widely described in the operations research literature as one of a number of 'Job Shop' problems , for example Garey , Graham and Johnson (1978) . Such problems are known as 'combinatorial' , referring to the combinatorial number of arrangements which may have to be searched to obtain the optimum solution . Miller-Merbach (1976) identifies three combinatorial types .

1. Assignment - the allocation of a set of n elements to another set of m elements .

2. Sequencing - ordering within a set of n elements.

3. Selection - selecting a set of n from within a larger set of m elements .

Truck despatching has aspects of 'selection' in the choice of a set of trips for each truck , ' assignment ' in the allocation of trips to individual trucks and 'sequencing' if the assigned trips are ordered . The despatching problems of log trucks have all three aspects and in addition some characteristics not usually associated with a conventional combinatorial problem .

1. The problem is multi-period ; the comparative performances of the trucks in a succession of periods is important .

2. The operations of the despatcher are adaptive ; a sequence of despatch schedules are required , each depending on the performance of the system in previous periods .

3. The operations of the trucks are stochastic because of the delays , breakdowns and variable travel times associated with truck operations .

While most combinatorial problems can be theoretically cast into a variety of models suitable for mathematical optimization , in most cases computation time for these models increases faster than polynomially with problem size , Miller-Merbach (op cit) . The characteristic that no computer based solution procedure can be found that runs in a time period which is a polynomially bounded function of the number of problem elements is the basis for classification of such procedures as NP complete , Lenstra (1977) . Management problems associated with transportation often fall into the NP class , Miller-Merbach (op cit , p 1) . Garey et al (op cit , p6) reported that the ' independent task , m processor problem ', closely related to the truck despatch problem , was NP complete for m > 2 and advocated the use of heuristic procedures .

Heuristic programming procedures often provide the only feasible methods for solution of such problems . Heuristics are the intuitive procedures used to formulate systematic approaches to problem solving . Heuristic programming is the implementation of these procedures and approaches as a computer programme , Miller-Merbach (op cit) .

There are several areas of concern in the published work on heuristics . Miller-Merbach (op cit) notes the lack of any unifying treatment of design methodology . This follows from the use of intuition to initiate the search for the solution . Another major area of concern

is related to the 'goodness' of the derived solutions since the solution procedures imply no guarantees with respect to approach to optimality . There are many suggestions for making comparisons of solutions as a basis for the selection of the most acceptable . Garey et al , (op cit) describe a method of determining worst case performance for scheduling algorithms which can be used as a reference for heuristic solution . Golden (1978) presents a method to estimate the likely optimum solution values for NP complete problems which can then be used to assess, by comparison , the performance level achieved with a heuristic solution .

Panwalkar and Iskander(1977) reviewed over 100 different heuristic rules . The classification system would describe the problem of scheduling trucks as 'global dynamic' ; global because information about the whole fleet is required simultaneously and dynamic because the allocation of a job can be changed in the event that it cannot be completed by the assigned truck.

Several applications of the use of heuristics to vehicle despatch have been reported . Brown and Graves (1981) incorporated heuristic methods in a computer aided real time despatch system for road tankers . In that application , an exact mathematical solution was not feasible by computation , because the available computer (although of large capacity) could not provide solutions sufficiently quickly to be useful in so called 'real time'. A heuristic algorithm was developed to provide

'prototype' solutions to the despatcher at a sufficient rate . The central design objective was to assist rather than to replace the despatcher for it was recognised that human decision making was a critical need in the complex truck despatching environment . Maister (1980 , p98) also strongly supports the need for a human despatcher to control such operations .

Brown and Graves (op cit , p29) suggested that to be effective in the control of the computer aided despatch system , the despatcher required an understanding of the computer solution procedure , and a capability programmed into the system to intervene in the solution building process . They cited as a typical example of despatcher intervention in a multi-truck despatch building process the capacity to 'fix' and remove from further consideration , specific trips on specific trucks .

The work of Brown and Graves (op cit , p22) illustrates the other role of the computer in aiding the despatcher by providing an information processing system . In association with the despatch system for the road tankers , the despatcher could store , retrieve and edit data on truck availability and type , customer requirement and proposed schedules as they are built up .

However , important questions about human capacity to use system status and problem information have been raised. Smith and Crabtree (1975) investigated aspects of scheduler decision making in the control of a simulated

job shop . Experiments involving the presence and absence of system status displays and an advanced computer based planning capacity were included in the investigation . They affirmed the limited capacity of a lone human decision maker to make effective use of either status information or the planning tool in controlling the complex sequencing problems .

Thus an important question in the rapidly developing field of 'man computer systems' is how computer processed information is best presented to facilitate human decision making . Smith and Crabtree (op cit , p224) showed the importance of simplicity in the presentation of information . Ceder and Stern (1981) selected a combination of heuristic decision rules and a computer based display system which produced schedule schematics to facilitate the intervention of experienced human schedulers in the preparation of bus timetables in forms familiar to the despatchers from their previous manual systems .

An application of minicomputers to the development of a realistic simulation of an industrial process requiring scheduling and a 'real time' decision support system was described by Buck , Deisenroth and Alford (1978) . Using an example in the scheduling and control of the soaking pit operations in a steel rolling mill , the minicomputer system provided the information base which allowed the decision maker to construct and store proposed schedules and a method of plan evaluation by means of a simulation of the prototype schedule . The decline in hardware costs associated with minicomputers and their growing capacity

make them available for a wider range of applications in 'man computer systems' .

Lucas (1981) described some modern applications of computer graphics . These developments are also important in the development of 'man computer systems' for they enable effective presentation of data to the decisionmaker as the basis for intervention in the computer programming of operations and control .

In summary , current approaches to the problems of vehicle despatch , at least for large scale or complex systems which exceed the capacity of human schedulers , appear to be restricted to the application of heuristics . Other areas of relevant work include the development of computer aided decision systems incorporating either decision algorithms , primarily heuristic , information display and storage systems , or both . Investigation of human performance in the use of these systems , suggests the importance of simplicity in information display and a capacity for intervention in the operation of decision algorithms . The reported success of 'man computer systems' in assisting scheduling operations indicates they would also be useful in despatching log truck fleets . The technology and relatively small costs of minicomputers suggests 'man computer systems' based on this technology would be feasible for system development .

2.1 A CONCEPTUAL MODEL FOR THE DESPATCH OF LOG TRUCKS

The model envisaged was a human despatcher supported by a minicomputer system .

2.1.1 The Despatcher

The despatcher would collect the information required as a basis for the allocation of trucks to the log hauling tasks . Essentially he would require , inter alia , data on a daily basis of the wood available at the individual landings and the truck availability lists.

The fixed objective is to transport wood available and the despatcher would seek to improve the efficiency of the transport operations on the basis of defined criteria. These criteria would desirably include the interests of all groups .

It has already been suggested that resistance to the introduction of a centralized despatch system must be expected from logging contractors and owner drivers who would lose their direct control of the landing operations. Also , the mill operators would have to be convinced of the possible gains from a centralized despatch system and feel confident that they could also convince the logging contractors and owner drivers of the gains before they would implement such a system .

The concept of equity for each truck was formulated as a criterion for the allocation of work by the despatcher . That each truck receive (as nearly as practical) the same amount of work would be an operational criterion that may reduce objections by the truck owners to the introduction of the centralized system . If it could be demonstrated by simulation modelling that , under a centralized despatch system , the wood becoming available at the landings was transported to the mill achieving a strict equity between trucks , then it could be argued that a centralized despatch system could also be capable of operating under modifications of the criterion which may have more appeal to the truck owners or contractors . For example , some trucks could bid to receive more work than others as a form of overtime , perhaps at 'marginal rates' , or the rates of transport from landings could be varied as specified by the logging contractor to better suit variations in the rate of bush production .

One difficulty with a centralized despatch system associated with a contractor based logging system , such as that in operation at Eden where logging crews and drivers are free to determine their working hours , is the coordination of the truck and loader drivers at the start of the working day . Another difficulty is that trucks are usually garaged at the drivers homes located in various of the small communities throughout the region , usually near the particular forest zone where the contractor is employed . In this study these difficulties were avoided by restricting the operations of the central

despatch trip allocation to the second and subsequent trips of the day . This would provide for drivers to arrange the place of the first load on each day and thus avoid dead running as far as possible . Similarly , it allows truck drivers to arrange their own start and first load time .

Thus for the purpose of model formulation in the first instance

1. The despatch allocation was prepared before the start of the day's work .

2. The despatcher would collect daily information on the volumes of wood available at each of the individual landings before some defined cut off , say 4.00 pm ; a time chosen to allow the despatcher time to prepare the next day's despatch allocation .

3. Trucks would be available for work allocation unless the despatcher was advised before the cut off time .

4. Truck drivers would be permitted to arrange their own first load for the next day , including importantly the use of the loader and notify the despatcher of the load before cut off time .

5. The daily allocation of trucks by the despatcher

would have a prime objective of ensuring that at the end of a defined period , say a month , each truck received , as nearly as practical , the same amount of work .

2.1.2 The Minicomputer System

The use of a separate minicomputer dedicated to the despatching function , rather than time sharing on a large computer system was assumed for the project . Consequently , limits on the available computer processor power were assumed .

It was envisaged the computer system would be programmed to :

1. Continuously process data collected such as the detail of loads delivered by each truck and the loads remaining at the landings and truck round trip times for specific compartments .

2. Display information such as the data collected , the currently operational or next days prototype despatch allocation , in a manner which is helpful to the despatcher and can be readily interpreted and manipulated .

3. Develop a prototype daily despatch allocation .

This study is primarily concerned with the development and testing of an algorithm to provide a prototype daily despatch allocation as the major component in a computer aided despatch system . The 'decision support' rather than 'decision replacement' aspect of the proposed system is emphasised by the modelling of the heuristic allocation procedure as occuring overnight or the day before , and thus being available for immediate modification , at the start of , and throughout the day , by the human despatcher .

Analysis , in this study , is restricted to that of the performance of the algorithm in producing a once daily despatch allocation . Analysis of this situation is considered useful in that , while the addition of 'human' capacity would undoubtedly provide improved performance, particularly through trip reallocation as a response to loader or truck breakdown, these results would provide an indication of likely minimum performance . Additionally , the computer generated allocation could reasonably be expected to provide an adequate backup (that is the computer allocation could be accepted as the actual despatch with minimal modification) , in times of staff turnover or as a new system was starting up .

Of course if successful computer based despatch allocation procedures can be developed , then there appears no practical reason why their use could not be extended to a 'real time' or 'on line' trip reallocation capability supporting the despatcher as events unfold throughout an operational day .

2.2 DEVELOPMENT AND TESTING OF A HEURISTIC

2.2.1 A Points System

A points allocation system was devised as the basis for determining priorities in the allocation of work to trucks to ensure some defined balance between the total work performed and hence the payments received by each truck. The basis of the points allocation system has a major influence on the operation of the despatch allocation system.

There are many indices that could be chosen for points allocation and of course factors could be combined into one index. Equitable opportunity for truck owners to earn profit, equitable working hours, and a ranking of the economically 'good' loads with the 'poor' all provide possible indices .

The accounting framework for the costs associated with the operation of a truck outlined in Chapter 1 ,(p1), shows that truck utilization is a very significant factor in the profitability of a road haulage operation and suggests that a point score system based on the actual trip time of completed loads would be a fair system. Trip time in its simplest form is the sum of travel time, time at the landing and time at the mill.

There are , of course , some problems associated with such a choice . The mill owners and the owners of the faster trucks might be assumed to favour such an index being based on 'average' or 'standard' trip time for each landing . An index based on 'actual' trip time would result in slower trucks being awarded more points for the same trip . The 'terminal operations' component of the actual trip time presents another problem. The time that a truck spends at the landing and mill can be considered as 'standing cost' time which is considerably less than 'travelling cost' and therefore not worthy of as many points per unit time as for travelling. Thus short trips which have a higher proportion of terminal time could be advantaged unless the points score were adjusted .

Overall the 'landing operations' are one of the significant issues associated with the acceptance by truck drivers and logging contractors of a centralized despatch system. Loading could remain the responsibility of the logging contractor who may have no direct financial incentive in ensuring the rapid loading of the trucks . Trucks loading from landings with slow loadout times would be disadvantaged by the adoption of a 'standard' trip index . A simple development of the central control system based on reliable radio reporting of landing time operations could readily allow separate points allocation on the basis of landing and travel times.

A points system derived from the total round trip time was adopted for this study . The main reasons were

that data is more readily available for total trip times, its simplicity was appropriate in the first model and modifications could be incorporated in subsequent development of the programme if testing shows them to be more appropriate.

2.2.2 The First Heuristic Allocation Procedure

Many heuristics use sorting procedures to prepare lists and then assign members of one list to another in predetermined ways. Best to worst assignment is one such way and is adopted here . In effect the best remaining trip set is assigned to the remaining truck with the worst pointscore.

The ongoing procedure adopted for the first heuristic was firstly to add the total previous pointscore of the truck at any stage (that is one point for each minute worked) and the projected points to be earned for the first trip* on the next day. The list of the projected pointscore at the end of the next days' first load for each truck was then sorted and stored . A prototype list of trip sets was made from the trips remaining for the day

* The first trip would be arranged by the truck driver to ensure the synchronization of startup time with the loader driver . The despatcher would be notified of the landing chosen to allow its inclusion in the day's despatch allocation .

by randomly assigning trips to trip sets. 'N' sets were prepared for the 'n' trucks . The pointscores associated with each trip in a set were added to give a projected points yield for each set and the list of trip sets sorted and stored . The despatch allocation was then built up by assigning the trip set with the highest point score yield to the truck with the lowest projected points score and so on, that is the 'N' sets were assigned to the 'n' trucks . The procedure is essentially best trip set to 'worst off' truck .

2.2.3 Testing the First Allocation Programme

A deterministic simulation of a twenty truck fleet working three trips a day over ten monthly periods each month of 20 days was developed and programmed to test the performance of the heuristics . This model comprised a trip generator, a points score accounting system and the heuristic allocation procedure . The daily lists of trips were drawn randomly from the selected distribution by the trip generator , and then assigned to the trucks in sets of three representing a full day's work . However , the model was deterministic in that the projected trip time was then assumed to be the actual trip time .

Two test series were conducted , Series A and Series B , with trip time distributions drawn respectively from a

uniform distribution and an empirical distribution obtained from observed round trip times at Eden . In each series , the model was tested by running the prepared computer programme twice , once with and once without the single pass heuristic . Without the heuristic , the daily three trip sets remained those initially assigned at random , and the experiment provided a benchmark for subsequent evaluation of the heuristic's performance . The same random number seed was used in both runs to generate identical trip lists.

Random allocation within a three trip day was chosen as a benchmark , firstly because it provided a procedure which was readily comprehensible . Secondly , because of the restricted range of trip times available for selection, and the assurance of three trips per simulated day , the procedure might be expected to perform creditably with respect to the equity measure proposed . Indeed , subsequent comparison with the results obtained for the benchmark simulation of the actual fleet operations , (discussed in Chapter 5), shows the random method of trip assignment to be superior .

2.2.3.1 Test Series A

For Series A tests the daily trip lists were drawn from a uniform distribution of trip times , lower limit 50 minutes , higher limit 150 minutes . After drawing the trip list for a day a scaling factor was determined by

dividing the resulting list-average trip time by 100 minutes . All trips in the daily list were then adjusted by this scaling factor to ensure that the total daily haulage task on each day of each of the ten monthly periods remained constant at 6000 minutes to assist in direct comparison of tests . This is similar to a transport manager's concern with holding the total haulage task for his fleet constant .

The results of total monthly trip times for individual trucks are presented in Figure 2.1 as comparative histograms , that is for ten monthly observations per truck for 20 trucks . The results of variation in daily total times within each month , as measured by the standard deviation of daily times within each month for each truck , are presented in Figure 2.2 as comparative histograms . The distribution statistics for these Figures are in Table 2.1 .

Figure 2.1 shows a marked collapse in the spread of the distribution resulting from the operation of the heuristic allocation algorithm . By comparison with random allocation of trips it improves very considerably the equity of the allocation of work to trucks . The standard deviations associated with these two distributions (Table 2.1) show a reduction of 87% in monthly total time variability , the principal measure of equity used in this study .

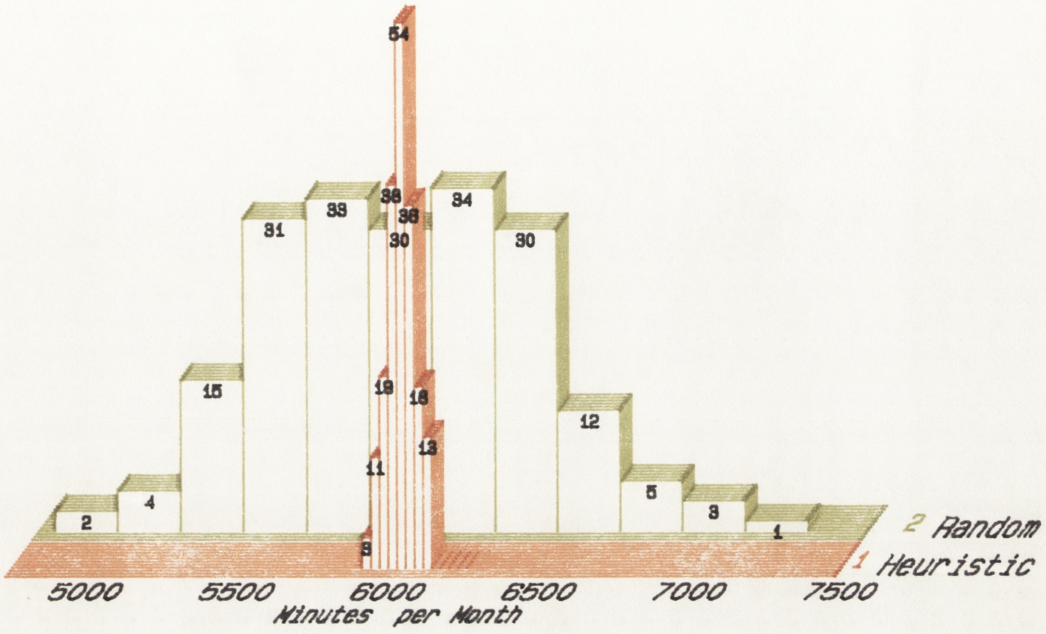


Figure 2.1 Frequency - Time Histograms of Truck Monthly Total Times Uniform Trip Distribution

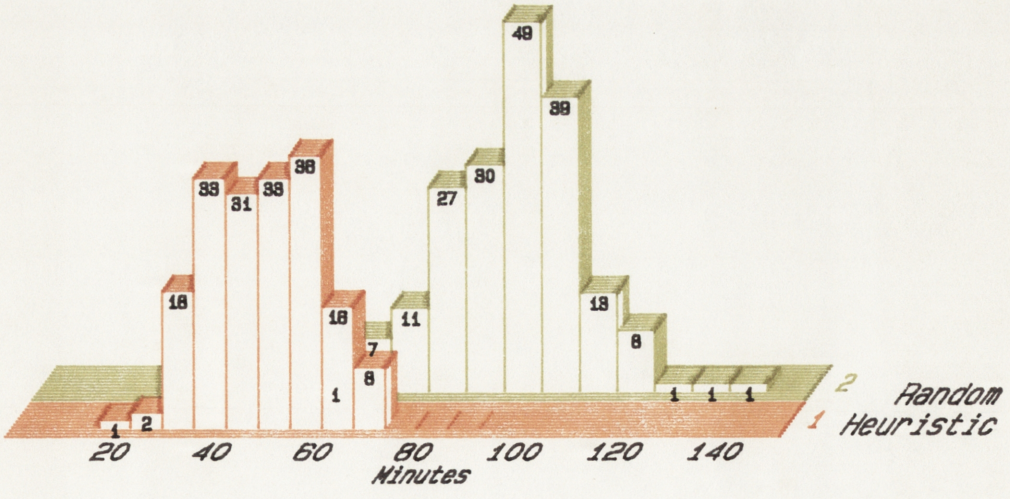


Figure 2.2 Frequency - Time Histograms of the Variation of Daily Total Time within each Month for each Truck

Figure 2.2 shows , by a shift of the distribution generated from heuristic reallocation , that the heuristic also reduces the average level of variation in daily times by about 46 % (Table 2.1) . An 'a priori' expectation that daily total time variation might have been increased by the operation of the heuristic 'over correcting' was not supported .

TABLE 2.1

DISTRIBUTION STATISTICS FOR TEST SERIES A
COMPARING THE SINGLE PASS HEURISTIC WITH
RANDOM ALLOCATION USING UNIFORM TRIP DISTRIBUTION

	RANDOM		HEURISTIC	
	MEAN (mins)	STANDARD DEVIATION (mins)	MEAN (mins)	STANDARD DEVIATION (mins)
AVERAGE MONTHLY TRIP TIME TOTAL	6000	416	6000	57
AVERAGE DAILY TRIP TIME VARIATION	94	15	52	12

2.2.3.2 Test Series B

The tests of Series A were repeated as Series B but with the trip times being drawn from a discrete distribution * derived from observations of round trip times for the Eden truck fleet (Figure 2.3) . In this test of

*The collection of data for this distribution is described in Chapter 4 .

the heuristic the daily trip lists were not scaled thus providing for some random variation in the daily total haulage task .

The results are presented , in Figure 2.4 , as comparitive histograms of the total monthly trip times , in Figure 2.5 as comparitive histograms of the average variation in daily trip times within a month for individual trucks and in Table 2.2 as the distribution statistics for the test series .

TABLE 2.2

DISTRIBUTION STATISTICS FOR TEST SERIES B COMPARING SINGLE PASS HEURISTIC WITH RANDOM ALLOCATION USING EDEN OBSERVED TRIP DISTRIBUTION

	RANDOM		HEURISTIC	
	MEAN (mins)	STANDARD DEVIATION (mins)	MEAN (mins)	STANDARD DEVIATION (mins)
AVERAGE MONTHLY TRIP TIME TOTAL	15710	850	15710	193
AVERAGE DAILY TRIP TIME VARIATION	196	31	112	23

Figures 2.4 and 2.5 demonstrate that the heuristic is again successful , firstly in improving equity in alloca-

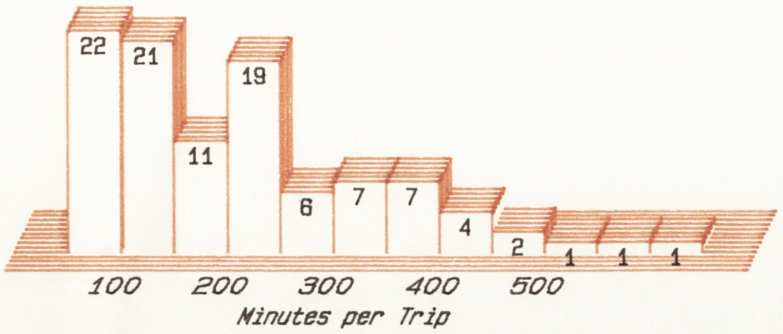


Figure 2.3 Frequency (%) Time Distribution of Round Trip Times recorded during July Study Month at Eden

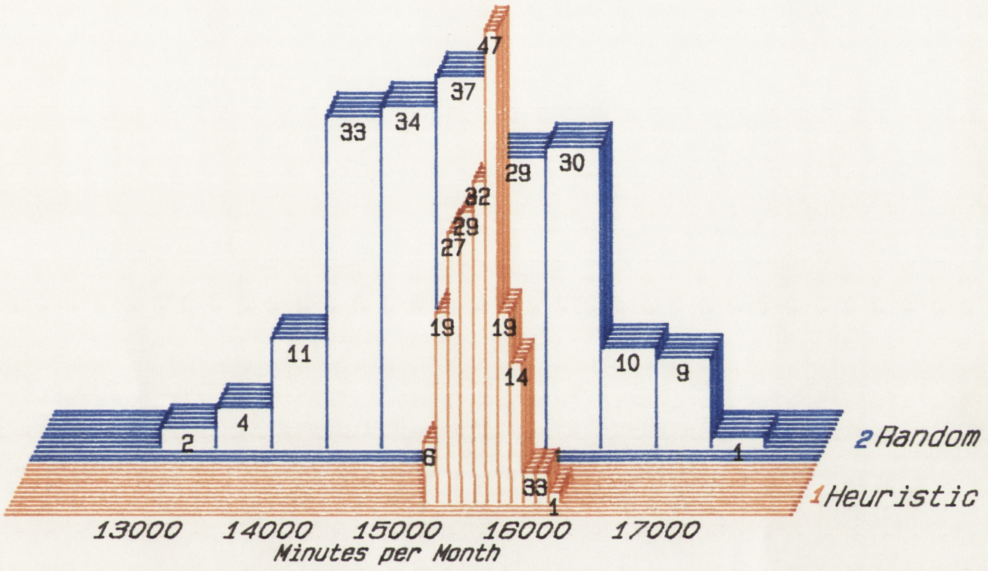


Figure 2.4 Frequency - Time Histograms of Truck Monthly Total Times Eden Trip Distribution

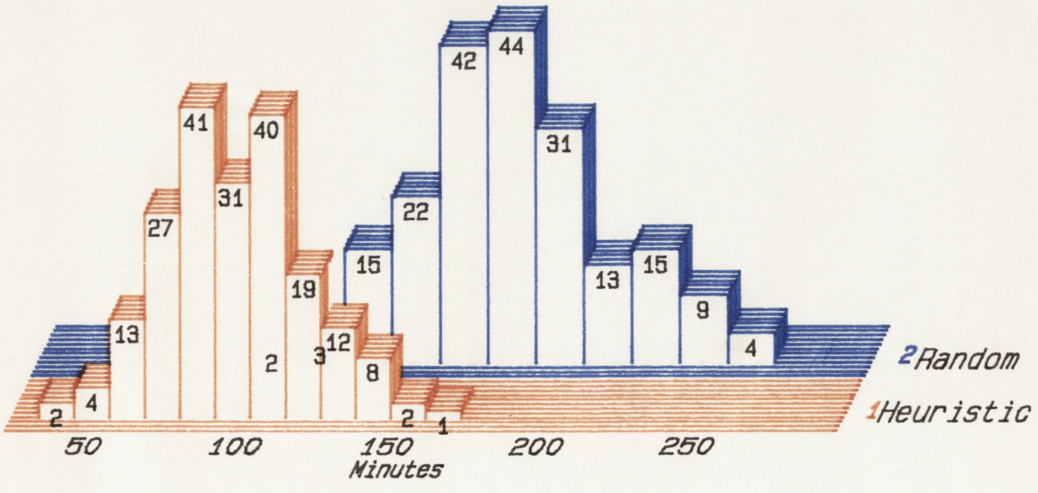


Figure 2.5 Frequency - Time Histograms of the Variation of Daily Total Time using Eden Trip Distribution

tion of work , the standard deviation associated with the the monthly totals in the heuristic test being reduced by 78% of that recorded for the random allocation , and secondly in that the variation of daily workloads of individual trucks is again reduced , by about 43 % of that of the random allocation .

In summary both series of tests provided strong evidence that application of the heuristic would be advantageous .

2.3 THE SECOND HEURISTIC ALLOCATION PROGRAMME

While the tests of the heuristic indicated that the first programme was successful for the conditions tested , two additional 'passes' were added in an attempt to improve its performance under a wider variety of conditions . They are called Pass 2 and Pass 3 respectively and follow the application of the heuristic previously described , hence called Pass 1 .

2.3.1 Pass 2 'Trip swap improvement Pass'

Pass 2 examines the swapping of trips between trucks. It is a typical 'improvement' heuristic similar in nature to that used by Barry and Robinson (1977). The objective of the improvement rule remains unchanged as minimizing the deviation by each truck from the average score after the designated trip set is completed. Pairwise compari-

sons are made of the trips allocated to the highest and lowest scoring trucks , then the next highest to the next lowest until the list is exhausted. For each pair, the longest trip allocated to the highest scoring truck is swapped if

1. it exceeds the shortest trip allocated to the lowest scoring truck and if

2. the defined maximum allowable length of the work day is not exceeded by the lower truck which is now receiving the longer trip .

The 'driver selected first trips' continue to be excluded from the paired comparisons. The remaining truck pairs are assessed in turn.

The maximum length of working day was set at twelve hours and is a necessary limit to the heuristic which could otherwise attempt to allocate all the trips to the 'poorest' truck . While the choice of twelve hours is somewhat arbitrary , such limits are acceptable in practice, implicitly in terms of what a driver is prepared to undertake and explicitly in terms of traffic regulations.

2.3.2 Pass 3 'Capacity enforcer - trip reallocation'

The third Pass is concerned directly with enforcing the rule that the defined maximum length of a working day

must not be exceeded . Although the procedures in Pass 2 check for daylength , those of Pass 1 do not and resulting trip schedules can still exceed the maximum . Pass 3 is comprised of two elements , a 'capacity enforcer' (Pass 3A) and a 'trip reallocator' (Pass 3B) . In Pass 3A the output schedule from Pass 2 is sorted again and placed in a 'linked list', a computer data handling technique which facilitates removing and reinstalling items when their points standing changes. The truck with the highest score is considered first and , if its daily work time exceeds the allowed maximum , a trip is selected for removal such that it is the smallest that could be removed to shorten the day length to a target level . In the second part of this Pass , 3B , the 'trip reallocator' , the removed trip is 'offered' to the truck with the lowest points standing. If this truck can accept it, without exceeding its own maximum day length , the trip is allocated to that truck. Otherwise the successively higher scoring trucks are considered in turn, until either the trip is allocated or the truck list is exhausted. If the truck list is exhausted then the trip is 'pushed back' to the work schedule of the next day.

When a trip is reallocated the recipient truck is removed and reinstated in the ordered list at its new level ensuring that the 'most deserving' trucks are always offered the first choice. The procedure works down the list checking in turn the working day length of each truck.

2.3.3 Summary description of Heuristic structure

The final structure adopted was

1. Pass 1 - Best to worst trip set allocator ;
2. Pass 2 - Individual trip swap improvement routine;
3. Pass 3 A,B - Capacity enforcer and subsequent trip reallocator .

2.3.4 Testing of the Second Heuristic Allocation Programme

A test programme was written to evaluate the performance of the three Pass heuristic allocation algorithm. Again two series of tests , Series C and Series D , were undertaken . The test program listings are in Appendix A.

1. Series C : Trips drawn from a uniform distribution
2. Series D : Trips were drawn from a distribution of round trip times recorded at Eden.

Testing was further expanded to investigate the effects of changes in workload and the effectiveness of the different components of the heuristic . Five different levels of total work load were considered in each test and six variations of the allocation procedure were tested.

The different heuristic combinations selected to indicate the contributions of the individual passes were

1. All three Passes active.
2. Pass 3B , trip reallocation following removal of a trip because the working day length of a truck exceeds the defined maximum, was disabled .
3. Pass 2 which includes the trip swap improvement routine and Pass 3B were disabled . Pass 3A is required to limit daylength .
4. Pass 1 the initial allocation building schedule and Pass 3B were disabled and Pass 2 was reenabled .
5. Pass 1 , Pass 2 and Pass 3B were disabled .
6. all three Passes were disabled causing random trip allocation and no daylength checking .

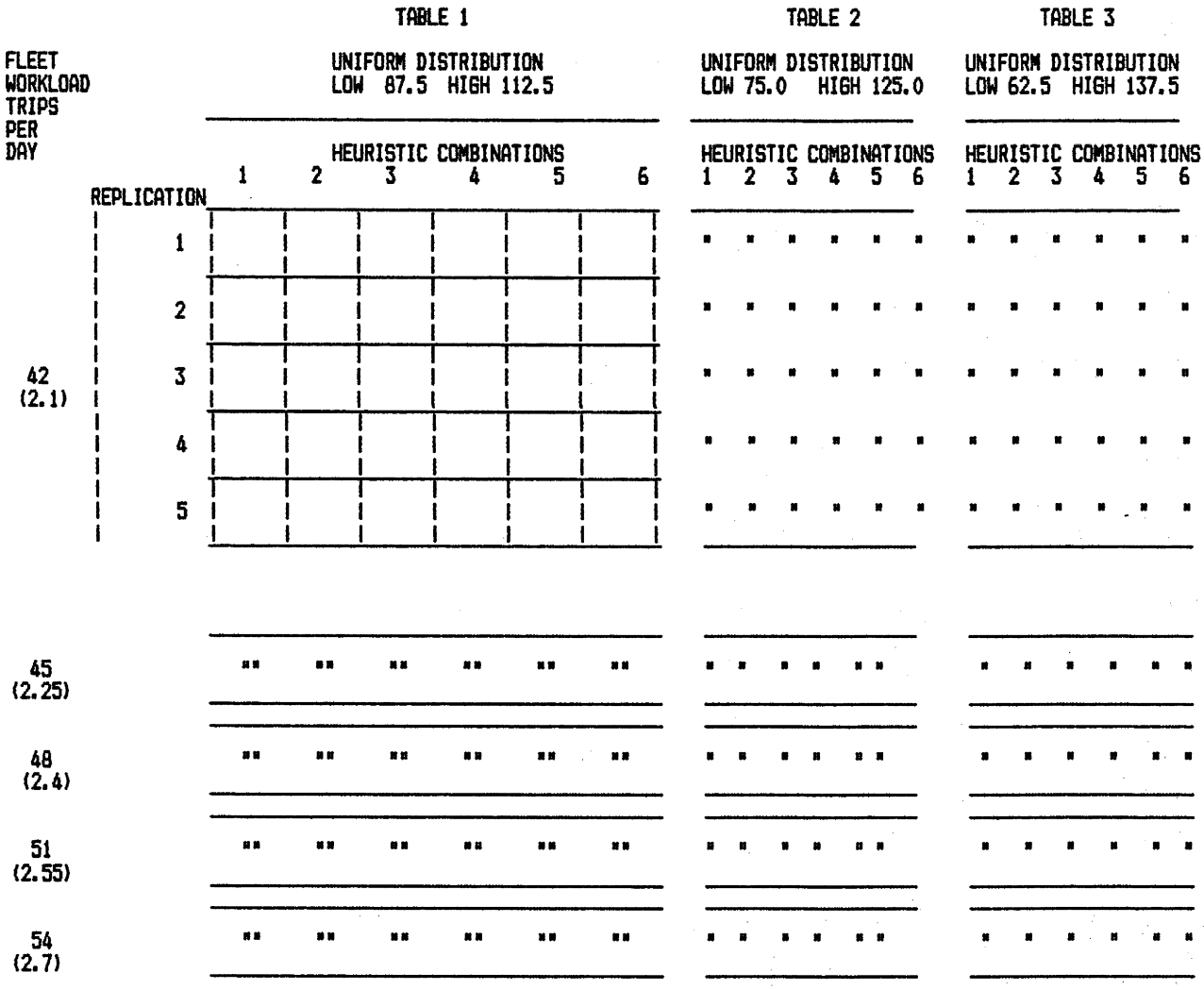
The experimental design , analysed in three two way Analysis of Variance tables ,for the series of tests is shown in Figure 2.6 . Each experiment was repeated five times .

Three statistics were collated for each cell of the experimental design .

1. The mean of total work times of each of the 20 trucks in the model fleet (average monthly total times) .
2. The standard deviation of the set of 20 individual truck monthly total work times (monthly

Figure 2.6

EXPERIMENTAL DESIGN FOR ANALYSIS OF VARIANCE



* Figures in brackets are Average Trips per Truck per Day

variation). This statistic provides the principal measure of equity as it measures dispersion of total work time in the monthly period .

3. The mean over the 20 trucks of the standard deviations calculated on the sets of 20 daily work times for each truck (daily variation).

The Minitab statistical package (Ryan et al , 1976) was used for the analysis and it is limited to two way analysis of variance .

2.3.5 Test Series C

In the test series based on the selection of trips from a uniform distribution of trip times three ranges of the distribution were used to test the sensitivity of the expanded heuristic programme to changes in the spread of trips available for allocation, namely ranges of 0.25, 0.50, 0.75 times the distribution mean , that is Uniform Distribution with Mean 100 minutes and

1. Lower limit 87.5 , Higher Limit 112.5 for the 0.25 range ,

2. Lower limit 75.0, Higher limit 125.0 for the 0.5 range , and

3. Lower limit 62.5 , Higher limit 137.5 for the 0.75 range .

2.3.5.1 Analysis of Truck Monthly Total Times

The results of the analysis of the monthly total work times is given in Tables 2.3a, 2.3b and 2.3c and presented diagramatically in Figures 2.7a , 2.7b, 2.7c . There was no statistical difference at the 95% level between the grand means for the three analysis tables for the levels of distribution range or between the six allocation treatments within any of the tables . The effects of increasing the work load seem to be carried over directly into the total work time indicating consistent performance of the allocation heuristic with increasing work load .

2.3.5.2 Analysis of Variation of Truck Monthly Total Times

The level of variation in monthly total times between trucks is the principal measure of the system's performance given the objective of attaining equity in the times worked by each truck. The result of the statistical analysis of the experimental data for the variation , represented by the standard deviation of the set of individual truck total monthly times for the whole fleet at the end of each month , is given in Tables 2.4a, 2.4b and 2.4c and are shown in Figures 2.8a, 2.8b and 2.8c. There was no statistical difference in the variation of monthly

TABLE 2.3 RESULTS OF ANALYSIS OF MONTHLY TOTAL TIMES FOR EACH TRUCK IN EACH MONTH

Table 2.3 a Results for Uniform Distribution and Range 0.25

TABLE OF MEANS

ROW						MEAN
1	7449	8002	8627	9099	9730	8582
2	7249	8002	8627	9099	9730	8582
3	7449	8002	8627	9099	9730	8582
4	7449	8002	8626	9095	9728	8580
5	7494	8051	8797	9156	9681	8636
6	7449	8002	8626	9095	9728	8580
COL MEAN						
	7464	8018	8683	9117	9713	8599

POOLED STANDARD DEVIATION = 224

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	2	50443	25211	(1
F2	FLEET WORKLOAD	4	47117188	11779297	258 ***
F1 * F2		8)	80735) +	10092	
ERROR		60)	3023400)	50390	
			(45649) +		
TOTAL		74	50271768		

ROWS are heuristic combinations + Interaction term less than Error and combined with Error for analysis
COLS are levels of workload

Table 2.3 b Results for Uniform Distribution and Range 0.5

TABLE MEANS

ROW						MEAN
1	7625	8104	8524	9336	9583	8624
2	7625	8104	8524	9336	9583	8624
3	7625	8104	8524	9336	9583	8624
4	7625	8104	8524	9336	9583	8624
5	7663	7976	8580	9054	9929	8640
6	7625	8109	8524	9333	9581	8634
COL MEANS						
	7637	8063	8543	9241	9697	8636

POOLED STANDARD DEVIATION = 201

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	2	580	290	(1
F2	FLEET WORKLOAD	4	42400304	10600076	262 ***
F1 * F2		8	735987	91998	2.68 **
ERROR		60	2419844	40331	
TOTAL		74	45556712		

ROWS are heuristic combinations COLS are levels of workload

Table 2.3 c Results for Uniform Distribution and 0.75 Range

TABLE MEANS

ROW						MEAN
1	7455	8082	8666	9126	9800	8626
2	7455	8082	8666	9126	9800	8626
3	7455	8082	8666	9126	9800	8626
4	7455	8082	8666	9126	9800	8626
5	7472	8036	8679	9273	9657	8623
6	7454	8082	8666	9122	9800	8625
COL MEANS						
	7460	8068	8670	9174	9752	8625

POOLED STANDARD DEVIATION = 228

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	2	75	37	(1
F2	FLEET WORKLOAD	4	48629140	12157285	253 ***
F1 * F2		8	150119	18765	
ERROR		60	3111879	51865	
TOTAL		74	51891212		

ROWS are heuristic combinations COLS are levels of workload

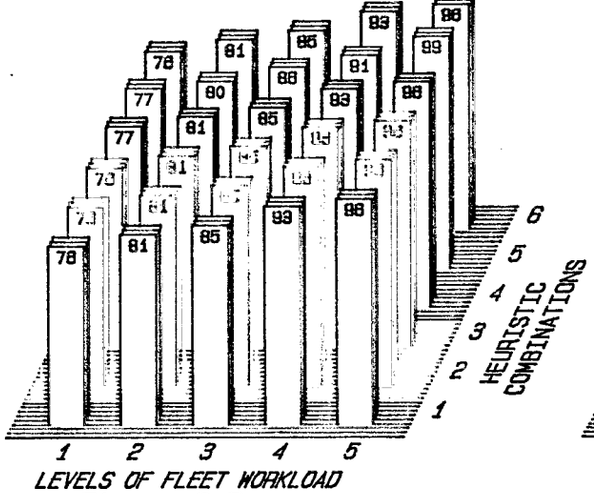


Figure 2.7 a

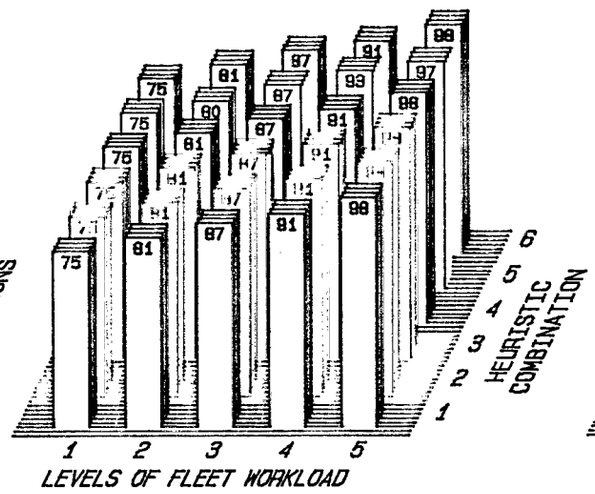


Figure 2.7 b

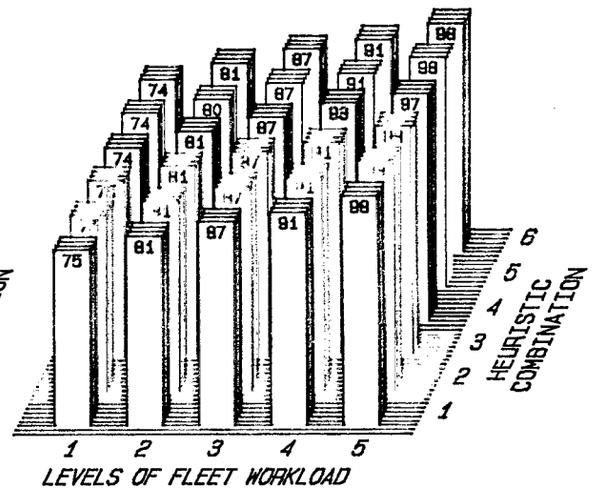


Figure 2.7 c

FIGURE 2.7 AVERAGE MONTHLY TOTAL TIMES

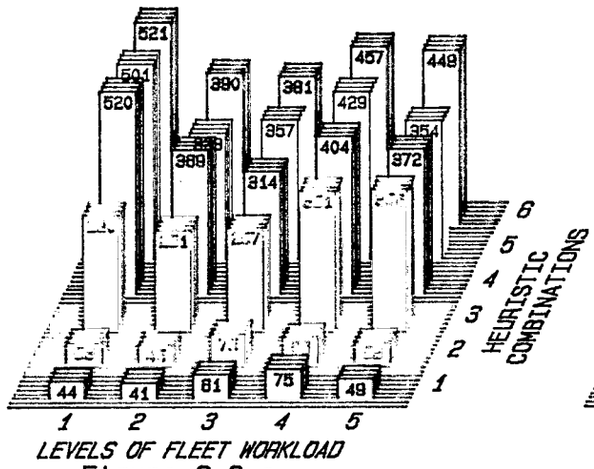


Figure 2.8 a

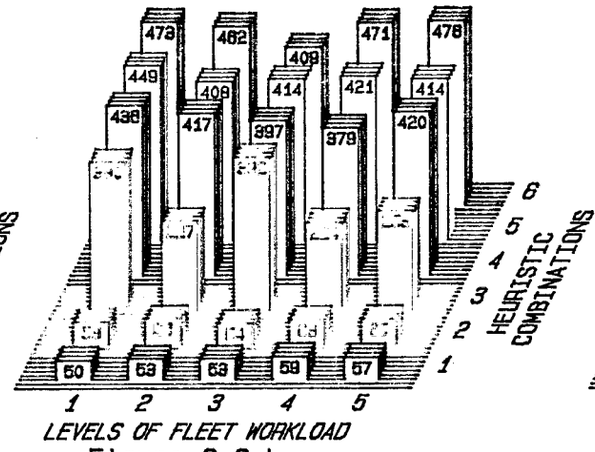


Figure 2.8 b

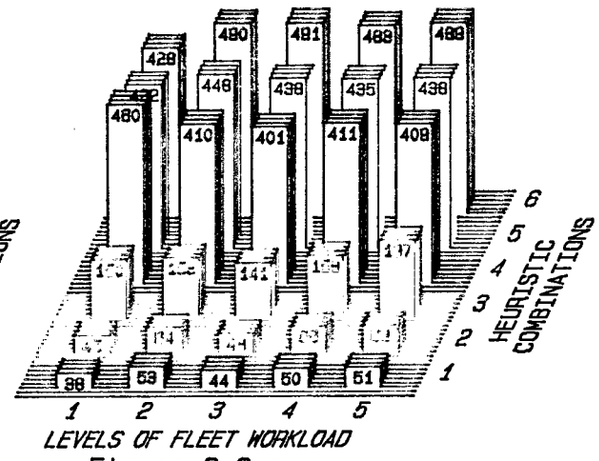


Figure 2.8 c

FIGURE 2.8 VARIATION BETWEEN INDIVIDUAL TRUCK TOTAL MONTHLY TIMES

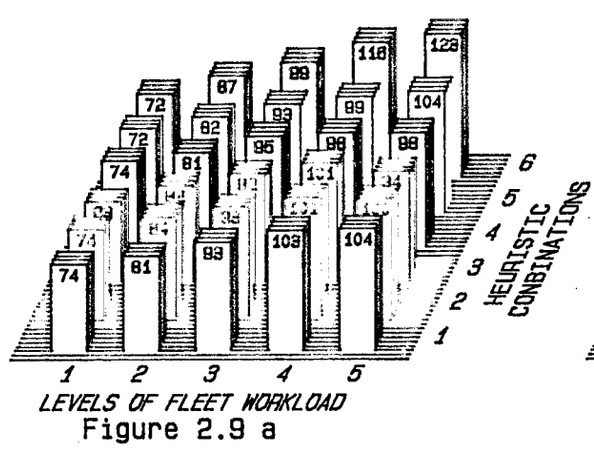


Figure 2.9 a

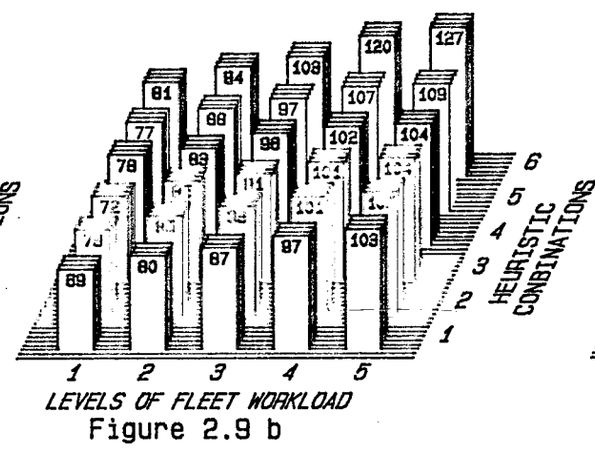


Figure 2.9 b

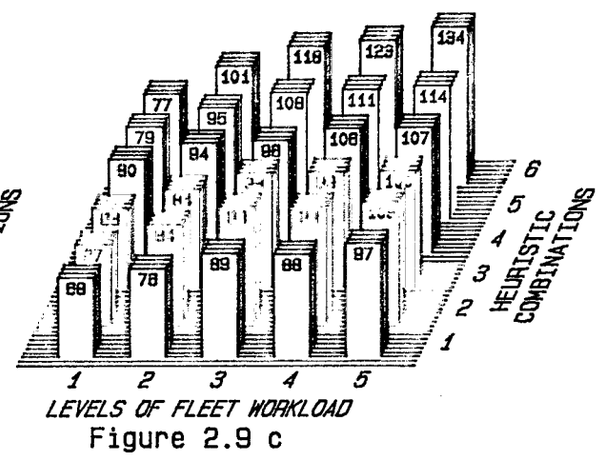


Figure 2.9 c

FIGURE 2.9 AVERAGE OF VARIATION OF DAILY TIMES WITHIN A MONTH FOR INDIVIDUAL TRUCKS

TABLE 2.4 RESULTS OF ANALYSIS OF VARIATION BETWEEN TOTAL TIMES FOR EACH TRUCK IN EACH MONTH

Table 2.4 a Results for Monthly Totals for 0.25 Range and Uniform Distribution

TABLE MEANS

ROW						MEAN
1	43	40	60	75	48	53
2	53	45	64	66	52	56
3	289	251	265	351	355	302
4	520	369	314	403	371	395
5	500	323	357	429	354	393
6	521	390	381	457	449	439
COL MEAN						
	321	236	240	297	272	273

POOLED STANDARD DEVIATION = 75

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	3827672	765534	136.1 ***
F2	FLEET WORKLOAD	4	159261	39815	7.1 ***
F1 * F2		20	178161	8908	1.58
ERROR		120	675027	5625	
TOTAL		149	4840122		

ROWS are heuristic combinations COLS are levels of workload

Table 2.4 b Results for Monthly Totals for 0.5 Range and Uniform Distribution

TABLE MEANS

ROW						MEAN
1	50	53	53	59	57	54
2	58	66	54	63	65	61
3	380	227	395	224	253	295
4	436	417	397	379	420	410
5	449	408	414	421	414	421
6	473	462	409	471	476	458
COL MEANS						
	308	272	287	269	281	283

POOLED STANDARD DEVIATION = 283

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	4186478	837296	81.4 ***
F2	FLEET WORKLOAD	4	27772	6943	(1
F1 * F2		20)	144365) +	7128	
ERROR		120)	1295880)	10799	
TOTAL		149	5654495		

ROWS are heuristic combinations + Interaction term less than Error and combined with Error for analysis

Table 2.4 c Results for Monthly Totals for 0.75 Range and Uniform Distribution

TABLE MEANS

ROW						MEAN
1	38	53	44	50	52	47
2	45	64	48	66	69	58
3	160	153	141	158	197	160
4	460	410	401	411	409	418
5	422	448	438	435	438	436
6	428	490	491	483	489	476
COL MEANS						
	257	269	260	267	276	266

POOLED STANDARD DEVIATION = 55

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	4967127	993425	324 ***
F2	FLEET WORKLOAD	4	6566	1642	(1
F1 * F2		20) +	34326)	1716	
ERROR		120)	371543)	3096	
TOTAL		149	5279562		

ROWS are heuristic combinations + Interaction term less than Error and combined with Error for analysis
COLS are levels of workload

total work times between trucks as a result of increasing the range of the uniform distribution of trip times , that is , between tables . The effect of increasing the work load is not clear. For the lower range (Table 2.4a) the treatment effect is statistically significant while it is not for the other ranges of 0.50 and 0.75 as given in Tables 2.4b and 2.4c respectively. Although the treatment is significant in the lower range Analysis of Variance (Table 2.4a) , there is no clear trend in the effect of increased workload evident in the row means .

The effect of the different heuristic combinations is highly significant statistically and provides one of the most important results of the test . The variations evident in treatment six (random selection of trips) is reduced by about 86% in treatment one (the full three Pass heuristic) . The effectiveness of the algorithm is reduced by only one per cent in treatment two (dropping the Pass 3B reallocation procedure following excess work day trip removal) . The reduction of 1% is not significant statistically. Treatment three , dropping Pass 2 , the trip swap improvement routine , is inferior to treatment one but it is still about 35% better than treatment six and the difference is statistically significant . It can be concluded that Pass 2 makes a significant contribution . Treatment four (dropping the Pass 1 allocation algorithm) is only about 12% better than treatment six and the differences are statistically different in only one of three tables (2.4c). It shows the critical role played by Pass 1 , and that it is more important than Pass

2 . Treatment five (Capacity enforcer only) provided an 8% improvement over random allocation but the differences were not statistically significant .

2.3.5.3 Analysis of Variation in Daily Total Times

The results of the statistical analysis of the experimental data for variation in daily work times , measured as the standard deviation of the set of the daily work totals for each truck for each month , is given in Tables 2.5a, 2.5b and 2.5c and shown in Figures 2.9a, 2.9b and 2.9c. Variation in daily worktime could be expected to be a factor of concern to truck drivers assuming they prefer a stable work day length . The statistics show significant effects of both work load and heuristic combination . Daily variation seems to grow proportionally with increasing fleet workload , that is , the number of trips per fleet day . This could be expected since the magnitude of the individual daily workloads is becoming larger . While the effect of the different allocation treatments is not generally clear the heuristic treatments are all significantly better than the random case (treatment six) at reducing the variation . The full algorithm (treatment one) is better than random (treatment six) by about 16% .

2.3.6 Test Series D

In the second test series the trip times were drawn from a distribution of the same shape as that obtained

TABLE 2.5 RESULTS OF ANALYSIS OF VARIATION IN DAILY TOTALS WITHIN A MONTH FOR EACH TRUCK

Table 2.5 a Results for a Uniform Distribution and Range 0.25

TABLE MEANS

ROW						MEAN
1	74	81	93	103	104	91
2	74	84	93	101	100	90
3	68	82	92	101	94	87
4	74	81	95	98	98	87
5	72	82	93	99	104	90
6	72	87	99	116	123	91
COL MEANS	72	83	94	103	104	91

POOLED STANDARD DEVIATION = 6.08

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	2190	438	4.7 ***
F2	FLEET WORKLOAD	4	22328	5582	150.9 ***
F1 * F2		20	1871	94	1.86 *
ERROR		120	4442	37	
TOTAL		149	30832		

ROWS are heuristic combinations COLS are levels of workload

Table 2.5 b Results for a Uniform Distribution and Range 0.5

TABLE MEANS

ROW						MEAN
1	70	80	87	97	103	87
2	73	85	93	101	105	91
3	72	85	91	101	104	90
4	78	83	97	107	109	92
5	77	88	97	107	109	96
6	81	94	103	120	127	105
COL MEANS	75	85	94	105	109	94

POOLED STANDARD DEVIATION = 4.7

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	4683	936	42.6 ***
F2	FLEET WORKLOAD	4	22853	5713	259.7 ***
F1 * F2		20	830	41	1.86 *
ERROR		120	2702	22	
TOTAL		149	31070		

ROWS are heuristic combinations COLS are levels of workload

Table 2.5 c Results for a Uniform Distribution and Range 0.75

TABLE MEANS

ROW						MEAN
1	68	76	89	88	97	83
2	67	84	99	99	103	91
3	68	84	98	99	100	91
4	80	94	96	106	107	97
5	79	95	108	111	114	101
6	77	101	118	123	134	110
COL MEANS	73	89	101	104	109	95

POOLED STANDARD DEVIATION = 5.3

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	11430	2286	81.6 ***
F2	FLEET WORKLOAD	4	25196	6299	225.0 ***
F1 * F2		20	2251	112	4.0 ***
ERROR		120	3386	28	
TOTAL		149	42264		

ROWS are heuristic combinations
COLS are levels of workload

from a field study at Eden (Figure 2.3 , p42) . The trips drawn from this distribution were scaled by 0.75 , 0.50 and 1.00 to provide three treatments to assess the effect of trip length on the performance of the algorithm. The total daily trips by the twenty trucks, that is the fleet work load, were again varied between 42 and 54 trips per day and the same set of allocation treatments applied. The Analysis of Variance Design adopted was the same as that used in Series C . Statistics presented are Average Monthly Time and Variation in Monthly Time , calculated as for the earlier test .

2.3.6.1 Capacity Overflow

In the case of the 1.0 scaled trips the combination of actual trip length and daily work loads in excess of 48 trips was found to be beyond the capacity of the truck fleet and a 'capacity overflow' was defined for this study as a build up of a backlog greater than the haulage task set for one day.

2.3.6.2 Analysis of Monthly Total Times

The results of the statistical analysis of the individual truck total monthly trip times associated with the levels of scaled trip lengths ,that is factors of 0.75 and 0.50 , are given in Tables 2.6a and 2.6b and Figures 2.10a and 2.10b . There was no statistical difference between the two tables at the 95% confidence level.

TABLE 2.6 RESULTS OF ANALYSIS OF MONTHLY TOTALS FOR EACH TRUCK IN EACH MONTH

Table 2.6 a Results for EDEN Distribution and 0.5 Scaled Trip Times

TABLE OF MEANS

	1	2	3	4	5	MEAN
1	5461	5768	5675	6536	7708	6205
2	5461	5768	6175	6534	7088	6205
3	5461	5768	6175	6534	7088	6205
4	5398	5823	6235	6592	7042	6229
5	5461	5768	6235	6592	7186	6228
6	5461	5768	6235	6592	7186	6228
COL MEANS						
	5450	5777	6247	6590	7082	6229

POOLED STANDARD DEVIATION =186

ANALYSIS OF VARIANCE

	DUE TO	DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	28232	5648	(1
F2	FLEET WORKLOAD	4	50063224	12515806	407.2 ***
	F1 * F2	20 } +	145783 }	7289 }	
	ERROR	140 }	4303350 }	34654 }	
	TOTAL	149	54394808		

ROWS are heuristic combinations + Interaction term less than Error and combined with Error for analysis
COLS are levels of workload

Table 2.6 b Results for EDEN Distribution and 0.75 Scaled Trip Times

TABLE MEANS

ROW						MEAN
1	8228	8780	9306	10144	10374	9387
2	8228	8780	9306	10144	10374	9387
3	8228	8780	9306	10144	10374	9387
4	8268	8691	9412	10061	10885	9423
5	8228	8780	9306	10144	10374	9387
6	8294	8780	9306	10051	10422	9373
COL MEANS						
	8228	8763	9323	10050	10422	9363

POOLED STANDARD DEVIATION 220

ANALYSIS OF VARIANCE

	DUE TO	DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	180220	36044	(1
F2	FLEET WORKLOAD	4	93275984	23318996	495.1 ***
	F1 * F2	20}	800888}	40044} +	
	ERROR	120}	5793226}	48277}	
	TOTAL	149	54394808		

ROWS are heuristic combinations + Interaction term less than Error and combined with Error for analysis
COLS are levels of workload

Table 2.6 c Results for EDEN Distribution and 1.0 Scaled Trip Times

TABLE MEANS
(MINS)

LEVEL OF WORKLOAD	HEURISTIC COMBINATIONS 1	2	3	4	5	6	
42 TRIPS	10820	10833	10834	10806	10743	10868	
45 TRIPS	11563	11582	11586	11632	11183	11637	
48 TRIPS	12313	12279	12378	12236	11490	12502	**OVERFLOW
51 TRIPS	12555	12609	12669	12437	11594	13193	**OVERFLOW
54 TRIPS	12643	12747	12437	12521	11602	14088	**OVERFLOW

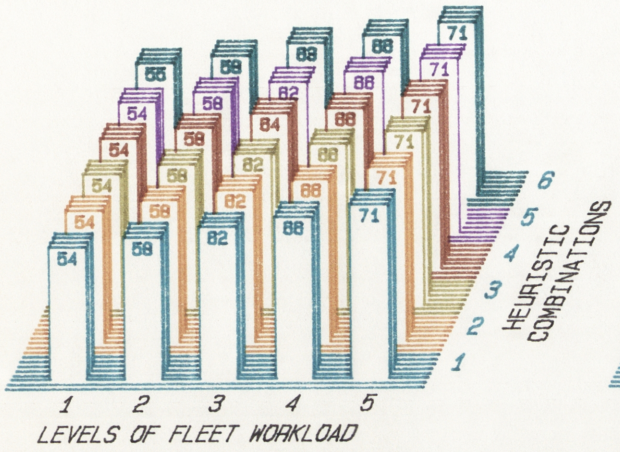


Figure 2.10 a

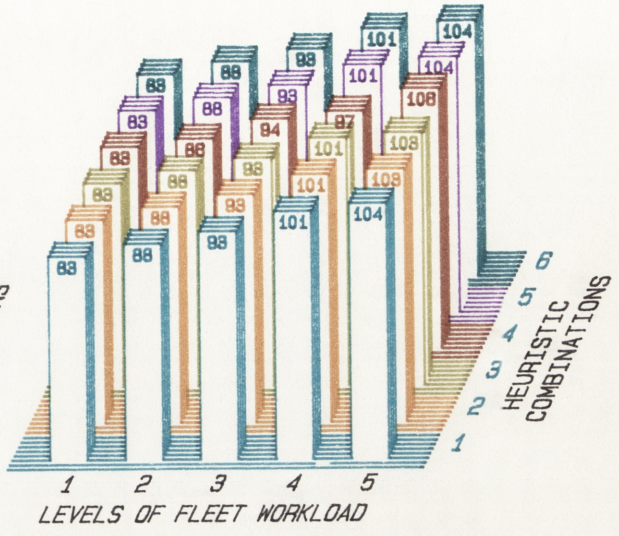


Figure 2.10 b

FIGURE 2.10 AVERAGE MONTHLY TOTAL TIMES

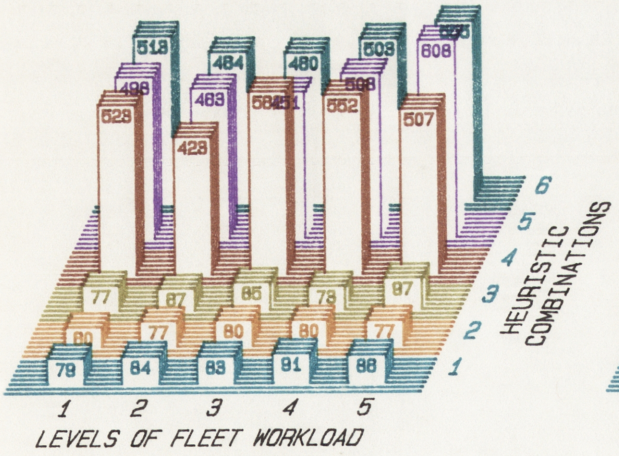


Figure 2.11 a

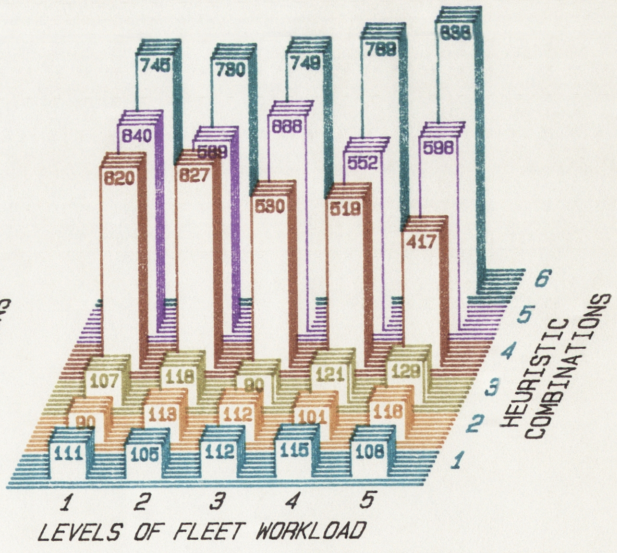


Figure 2.11 b

FIGURE 2.11 VARIATION BETWEEN INDIVIDUAL TRUCK TOTAL MONTHLY TIMES

The results for the 1.0 scaled trip lengths , Table 2.6c , provided some important insights into the behaviour of the heuristic allocation procedure as daily capacity was approached .

The first concerns the likely levels of fleet capacity . The theoretical maximum daily capacity for trip times under the model's assumptions is 14,400 minutes, that is 20 trucks for 12 hours. Indications of practical daily capacity without the heuristic are given by treatment five , the Pass 3B only treatment which is random trip initial allocation with no trip reallocation , and simply enforces the maximum daylength . Practical capacity could be expected to be much less because the trucks need to complete a whole number of trips (the daylength / round trip problem outlined earlier) ,and appears to be about 80% of the theoretical maximum at the 45 trips per day loading , just below the onset of capacity overflow . Treatment one , the full three Pass heuristic system , allowed about 4% more work time without overflow . Thus , under conditions of a defined maximum allowable daylength and , for fleets working at maximum throughput, the full three pass heuristic trip reallocation procedures appear to increase capacity by about 4 % .

Secondly , the heuristic procedures appear to increase throughput as the trip demand rises into capacity overflow . Fleet throughput (minutes worked per day) approached 90% of the theoretical maximum capacity under the condition of overcapacity , although , of course not

all loads available were being shifted . The main reason is that as the backlog builds up into capacity overflow the range of trip lengths available to the heuristic algorithm of Pass 3 has a wider range of trip times available to select from to 'fill the holes' in the allocation. This appears to be an important characteristic of the algorithm under conditions of transient excess demand , in that it works more efficiently under stress and therefore backlogs are cleared more quickly . This behavioural characteristic is a form of damping and is widely recognised as contributing to stability in many systems . Of course , sustained excess demand results in continuous capacity overflow and some trips not being hauled .

2.3.6.3 Analysis of Variation of Monthly Total Times

The variation of the monthly totals of trip times between trucks is given in Tables 2.7a and 2.7b and Figures 2.11a and 2.11b . Performance in minimizing this variation represents the primary measure of fleet equity . There was a 30% increase in grand mean variation between the two tables corresponding to scaling of trip length by factors of 0.50 and 0.75 , but this appears to be a result of the consequent 50% increase in the magnitude of individual trip times and monthly totals .

The effect of the allocation treatments is highly significant and readily seen in Figure 2.11b .

TABLE 2.7 RESULTS OF ANALYSIS OF VARIATION BETWEEN MONTHLY TOTALS FOR EACH TRUCK FOR EACH MONTH

Table 2.7 a Results for EDEN Distribution and 0.5 Scaled Trip Times

TABLE MEANS

ROW						MEAN
1	79	84	82	91	86	84
2	60	77	80	80	77	75
3	77	67	85	73	97	80
4	532	432	562	552	507	513
5	498	463	451	508	608	505
6	513	464	460	503	555	499
COL MEANS						
	291	263	287	301	322	293

POOLED STANDARD DEVIATION = 59

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	6817999	1363600	386.9 ***
F2	FLEET WORKLOAD	4	55217	13804	3.92 **
F1 * F2		20	117439	5872	1.66 *
ERROR		120	422875	3524	
TOTAL		149	7413529		

ROWS are heuristic combinations COLS are levels of workload

Table 2.7 b Results for EDEN Distribution and 0.75 Scaled Trip Times

TABLE OF MEANS

ROW						MEAN
1	111	105	112	115	108	110
2	90	113	112	101	116	106
3	107	118	90	121	129	113
4	620	627	530	519	417	542
5	640	589	666	552	596	608
6	745	730	749	789	832	770
COL MEANS						
	385	380	376	366	367	375

POOLED STANDARD DEVIATION = 80

ANALYSIS OF VARIANCE

DUE TO		DF	SS	MS	F
F1	SCHEDULING TREATMENTS	5	11240699	2248140	351.2 ***
F2	FLEET WORKLOAD	4	8293	2073	(1
F1 * F2		20	224518	11226	1.75 *
ERROR		120	768254	6402	
TOTAL		149	12241764		

ROWS are heuristic combinations COLS are levels of workload

Treatments one , two, and three which all include Pass 1 are about 80% better than the random, unrestricted daylength combination represented by treatment six . Treatments four and five are not statistically different from the random allocation (treatment six) for the set of shorter trip lengths (0.5 scaled) but perform about 20% better for the longer (0.75 scaled) trip lengths . The increasing effectiveness of treatment four , which is based on Pass 2 and the daylength enforcing Pass 3A , as the workload rises , is seen in Figure 2.11b .

The effect of treatments associated with the number of trips per day (fleet workload) , showed contradictory trends between the trip lengths corresponding to the two scaling factors of 0.50 and 0.75 , and appears to be a result of the changing behaviour of treatment four noted above . Table 2.7a (scaling factor 0.5) shows that the workload treatment effects are statistically different and increased with the trip length . Table 2.7b (scaling factor 0.75) shows that variation decreased with increased daily workload although the treatment effects are not statistically significant . The interaction term is significant in both tables , and is most obvious in Figure 2.11b where the effect of the maximum daylength constraint in treatments four and five is contrasted with the steady climb in treatment six (no heuristic , no daylength limit) and again the improved performance of treatment four is noted .

Analysis of the data for the unscaled trip length , although not analysed statistically because of the incomplete table due to capacity overflow , showed the five active treatments (one to five) perform about 50 % better than the random treatment six , until the onset of capacity overflow . However there is an improvement in the relative performance of treatment two and three over treatment one , which can be attributed to their increasing role in conditions of backlog , confirming the result of the earlier test series .

2.4 REVIEW AND SUMMARY

Truck despatch problems appear to belong to a very difficult class of problems (identified as NP complete) for which generally practical mathmatical programming solution procedures have not been devised . In addition , a variety of human and technical interactions experienced in the management of truck fleets have led workers to the conclusion that human control of despatching is essential. However , systems for computer aided decision making were identified as having potential in the acknowleged complexity of real time despatch . Programmed procedures would be required for such systems to provide for despatch evaluation and to create baseline or backup despatch allocations . The implementation , evaluation and demonstration of an appropriate programmed procedure was adopted as the major goal of this study . Techniques of heuristic programming were identified as the most appropriate for the project .

Development of the heuristic proceeded in two stages. The first stage algorithm was based on a ' best to worst ' approach and subsequent testing suggested excellent performance by reducing the variation in simulated monthly totals of worktime by about 80 % in comparison to those from workdays with three randomly assigned trips . Minimizing this variation was accepted as the principal objective in the development of the procedures , in accordance with a goal of work allocation equity .

The second stage additions were intended to improve , or at least maintain performance under a wider range of conditions , that is 'increase robustness' . They consisted of an 'improvement ' heuristic followed by a 'capacity enforcer ' and an associated 'reallocation' heuristic .

Testing of the fully developed heuristics under two types of trip distribution and varying 'allocation intensity' , that is , total trip work load divided by number of trucks , showed major reductions in the variations of monthly total pointscore between trucks could still be achieved even under a wide range of work load conditions . The initially developed Pass 1 heuristic was shown to make the greatest contribution to system performance . If the system were to be developed to provide real time support during the day , being used by the despatcher as required, as external events forced changes to the initial despatch, the trip swap improvement routines of Pass 2 could serve as a model . Some important behavioural characteristics

of the heuristic were identified , including an ability to improve throughput under conditions of capacity overflow , aiding system stability . At the onset of capacity overflow the full three pass heuristic appeared to allow about 4% greater throughput than was the case with random allocation and an enforced daylength .

The next step involved the testing of the heuristic in the 'stochastic' environment more closely representing the log truck fleet environment . The development of a discrete event simulation model to provide this testing environment is described in Chapters 3 and 4 . The objective of the testing would be to obtain further information on the performance of the algorithm , in particular the effects of variation in fleet workload . This work is described in Chapter 5 .

CHAPTER 3

A SIMULATION MODEL FOR LOG TRUCK FLEETS

3.0 INTRODUCTION

In conjunction with the growth in 'systems' thinking there has been widespread recognition of the complexity of the interactions in many modern industrial production systems . In contrast to many other numerically oriented techniques , simulation modelling can often cope with this complexity , and is now used commonly as an operational research technique . Major factors in the evolution of simulation modelling have been :

1. access to powerful computer processing facilities,

2. the development of computer languages which facilitate efficient implementation , and

3. a theoretical foundation for the application of the modelling approach .

Zeigler (1976) suggests a framework of five basic elements for a systematic approach to simulation modelling:

1. the real system : the source of the observations,

2. the experimental frame : the limited set of circumstances under which the real system is observed and within which any interpretation can be presumed valid ,

3. the base model : the modeller's view of how the system functions . The model attempts to account for the input-output behaviour of the observed real world system . Shannon (1981) terms this the conceptual model . Zeigler (op cit) states that the base model can never be fully known or quantified ,

4. the lumped model : the simpler abstraction of the base model which is capable of implementation,

5. the computer model :the logistics of the implementation of the lumped model as a computer program . Shannon(op cit) adds a sixth stage to Zeigler's framework , that of

6. experimentation or use of the model .

The modelling approach adopted in this study was based on Zeigler's framework and is taken up in Section 3.2 . However , some general considerations in the development of 'Discrete Event Simulation Models' are discussed first .

3.1 DISCRETE EVENT SIMULATION MODELLING

3.1.1 Model complexity

Reduction of the experimenter's conceptual or 'base model' to a 'lumped model' capable of implementation is recognised as a creative process of design rather than solely logical deduction and is usually subject to two opposing pressures . It is often desirable to keep at a high level the complexity of the model description the better to capture the 'richness' of the structure of the observed system . However , this is usually tempered by a limited capacity to obtain adequate data as input to a more refined level , by increased difficulties of verification of an improved model or doubts that the experimenter can control or comprehend the more detailed information produced . There is usually a compromise .

3.1.2 Computer implementation

Zeigler(op cit) , in common with most authors from the operations research community , is careful to distin-

guish between the model definition process ; that is , formulation of the lumped model , and the computer programming stage . He stresses the relative importance of the definition of the model .

Recent contributions from the computer science/software engineering fields stress the contribution of improved and sophisticated computer languages to the narrowing or even removing of Zeigler's distinction . Watts(1978) argues that suitable modern languages provide an improved medium for the modeller to design and implement a model , most notably through the provision of language constructs such as 'entity' , 'event' and 'process' which parallel the cognitive processes of model design .

Osborne(1977) identifies three attributes of a language suitable for simulation modelling .

1. An automatic method of event sequencing , that is an orderly method of passing control from one program segment to the next .

2. A convenient method for generating and referencing entity data structures.

3. Facilities for collecting references to entity structures into sets .

Kreutzer(1983) provides a description of the development of these language constructs and a taxonomy of the resultant simulation languages as illustrated in Figure 3.1 . He notes that "raising the semantic level of a language provides an increase in simplicity , conceptual congruence , ease of learning , reduced error rates and speeds model development" (Kreutzer , op cit , p74) . However he also notes that such improved facility usually "has to be paid for by a decrease in flexibility , generality of concepts , and processing and storage efficiencies" (Kreutzer , op cit , p74) .

FIGURE 3.1

TAXONOMY OF SOFTWARE LEVELS

Level 1	Modelling with general purpose programming languages	[FORTRAN , PL/1 , PASCAL ...]
Level 2	Modelling with procedure oriented simulation software	[GASP , SIMPAS , SIMPL/1 , SIMSCRIPT , ECSL, SIMULA ...]
Level 3	Modelling with descriptive scenarios	[GPSS , INS , DEMOS , ...]
Level 4	Modelling with model generators and declarative systems (a field for specialized software packages)	[DRAFT,CAPS..] and [Warehouse planning: -- SIMWAP] [Computer system simulation : SCERT, SAM , CASE ,...]

Source : Kreutzer , op cit , p 74 .

The level two languages of Kreutzer(op cit) , for example SIMSCRIPT and SIMULA , combine the specialist simulation attributes which aid in model implementation with the power and flexibility of a general purpose language .

Indeed, as Stanton (1977) notes , large scale simulation modelling is among the more demanding of computer programming applications . This has provided continuing stimulus to computer scientists to develop better language constructs and programming methodologies and resulted in improved language features particularly those identified by Osborne (op cit) , which are also of major importance to many other programming applications . Stroustrup (1983) provides an example of the continuing language development with his inclusion of constructs of the 'CLASS' type in modern languages .

3.1.3 A Programming Language for the Model

Programming requirements for this thesis included the efficient implementation of a large scale simulation model as well as the heuristic scheduling algorithm described in Chapter 2 . The requirement of the simulation model favoured the choice of a language from Kreutzers higher semantic levels (Figure 3.1) . The heuristic algorithm favoured the choice of a lower level general programming language . The choice for this study was SIMULA 67 , one of the higher level languages appropriate to a large scale simulation model and familiar to the author .

3.1.4 SIMULA 67

SIMULA 67 well illustrates the continuing innovations of high level languages . Primarily an ALGOL derived high level language it has the powerful addition of the 'class' concept , Dahl and Nygaard , (1966) . The language is thus readily capable of extension and necessary list handling and simulation features are provided as system class language 'extensions' . User defined classes such as Class Truck or Class Chipmill which were developed for this study are accessible as language extensions .

SIMULA has become widely used as a base language to further develop specialist simulation environments , for example DEMOS ,(Birtwistle , 1981) and DISCO , (Helsgaun, 1980) . In this study it aided the implementation of the complex truck allocation routines , described in Chapter 2 , through the provision of effective list processing and set handling constructs . An important attribute of the block structure of SIMULA is the capacity to implement 'top down programming' techniques with the development of the program proceeding as the development of a series of layers , each with increasing detail , but with each program module separated from others except where linkage is explicitly implemented . This feature was an important contribution to the efficient development of correct programs . It also allowed simultaneous descriptions of both the 'lumped model' and the 'computer model' using the program 'procedure skeleton' combined with a descriptive choice of procedure and class names .

3.1.5 Verification and validation of simulation models

Verification and validation are acknowledged as major issues in the use of simulation methods for problem solving (Zeigler , op cit) . Verification is the process of checking the correctness of translation of a designer's formal model into a functioning computer model . It seeks to answer the question ' does the program function in the way the programmer intended ? ' . Validation is the broader task of checking the behaviour of the model and answering the question 'does the model generated behaviour describe the modeller's view of the real systems (his base model) adequately for the purpose of its intended usage ' ? . Validation has also been described as a process of generating confidence in the model (Sargent, 1979) .

Zeigler (op cit) noted several states of validation. A model is said to be 'structurally' valid if it embodies the same physical components and interactions as the target system . A stronger level of validity is that the model is ' replicative' , that is the model is capable of reproducing the 'behavioural' data used in its development. Finally , models may be 'predictively' valid if they prove capable of adequately predicting system behaviour for sets of input data beyond those used in its development.

Sargent (op cit) noted that confidence in model performance depends on each of the phases of definition ,

implementation , verification and validation . He suggested that the extent of effort expended in each of these phases depends on the relative importance (and worth) of improved confidence in the model and the costs associated with the testing required .

Shannon (1981) detailed a wide range of tests and testing viewpoints which may be used in validation . He also emphasised the role of sound software engineering techniques in contributing to validation as well as program implementation .

3.1.6 Pseudo Random Number Generation

Stochastic simulation techniques are based on the use of random numbers and their efficient generation is a basic requirement of the computer system or programming language used . Leeming (1981) , in a review of a number of simulation languages , noted that the generator used in a SIMULA compiler failed a randomness test .

The generator used in the UNIVAC SIMULA compiler is a multiplicative congruential one , Simula Programmers Reference (1982) . The randomness of this generator was tested using a runs test , claimed to be one of the most discriminatory randomness tests (Maisel and Gnugnoli, 1972, p142). They reported that the for two tailed 'runs' test in a sample of 100 , the occurrence of more than 61 , or fewer than 40 runs supported the rejection of a null hypothesis that the generator was random at the 95%

significance level . A run is the occurrence of two consecutive numbers either both greater than or both below the median .

One hundred sets of one hundred numbers were generated by the Univac Simula compiler and numbers of 'runs' were counted . There were only three occurrences in the one hundred tests when the numbers of runs were either greater than 61 or less than 40 providing no evidence supporting the hypothesis that the number sets were not random . The pseudo random number generator used in the UNIVAC SIMULA compiler was thus judged to be satisfactory.

3.2 FORMULATION OF THE SIMULATION MODEL

The description of the development of the simulation model for a log truck fleet is presented in terms of Zeigler's five basic elements , namely

1. the real system
2. the experimental frame
3. the base model
4. the lumped model
5. the computer program .

The truck fleet as it was operating at Eden during the period December 1979 to July 1980 was used as a reference for the conceptual description of a model . The

model description is not generalised to encompass all log truck fleets , for example , in the defined model all trucks deliver to the one mill . While the description is most appropriate to the orientation of this study , examination of centralized despatch of log trucks serving one mill , it is nevertheless also appropriate to other log truck fleets . As conceived , the model could be readily applied to the centralized despatch of log truck fleets serving several mills if there is rapid communication between the separate mills or trucks and the despatcher , for example , by radio network .

3.2.1 The Real System

The real system is a truck fleet and its associated loading and unloading facilities . The log truck fleet at Eden was used as a reference for this study .

3.2.2 The Experimental Frame

The specific activities under study are those of each truck in the fleet , their responses to assignments of particular sets of trips and to loading and unloading operations . Thus , aspects of queueing at both the forest landing and the mill are within the experimental frame .

The experimental work envisaged was to test the effect on truck operation of variation in travel time ,

landing and unloading operations and trip assignment , and to aggregate the individual truck operations to ascertain likely system performance .

3.2.3 The Base Model

Log truck fleets can be viewed as socio - technical economic systems . It is clear that , for example , social interactions of the drivers , the attitudes of the individual drivers in regard to starting and finishing times , the technical performance of trucks and the economic needs of the drivers , the owners , the logging contractors and the company purchasing the wood can all influence a trip decision and the overall performance of the fleet .

At Eden , the trucks were generally tied to particular logging crews and drivers could assess in advance the requirement for transport during the current day and usually the next . This allowed them to plan start times , extend turnaround time at the landings or extend the working day length depending on , inter alia , the log volumes ready for hauling from the landing . Extended turnaround times at the landings were often observed during the studies , particularly at mealbreak time .

Thus control of the fleet rested with many decision makers in the real system . Two critical areas of decision making affecting the allocation of trips , emerged during the field studies of the reference fleet .

1. The driver's choice of daily starting time .

2. The decisions by the logging crews in respect of wood deliveries to the landings , the loading times of each truck at the landing and the number of loads to be hauled for the day.

3.2.4 The Lumped Model

The final design adopted is illustrated in Figure 3.2. The travel and load sequences are implemented as separate elements . Loading is included as a separate element to enable the model to be used in experiments to study the likely effects of different loading stategies . Mill operations are modelled as a one server queue with fixed delays rather than as a four stage process of sequential servers .

3.2.5 Landing Operations

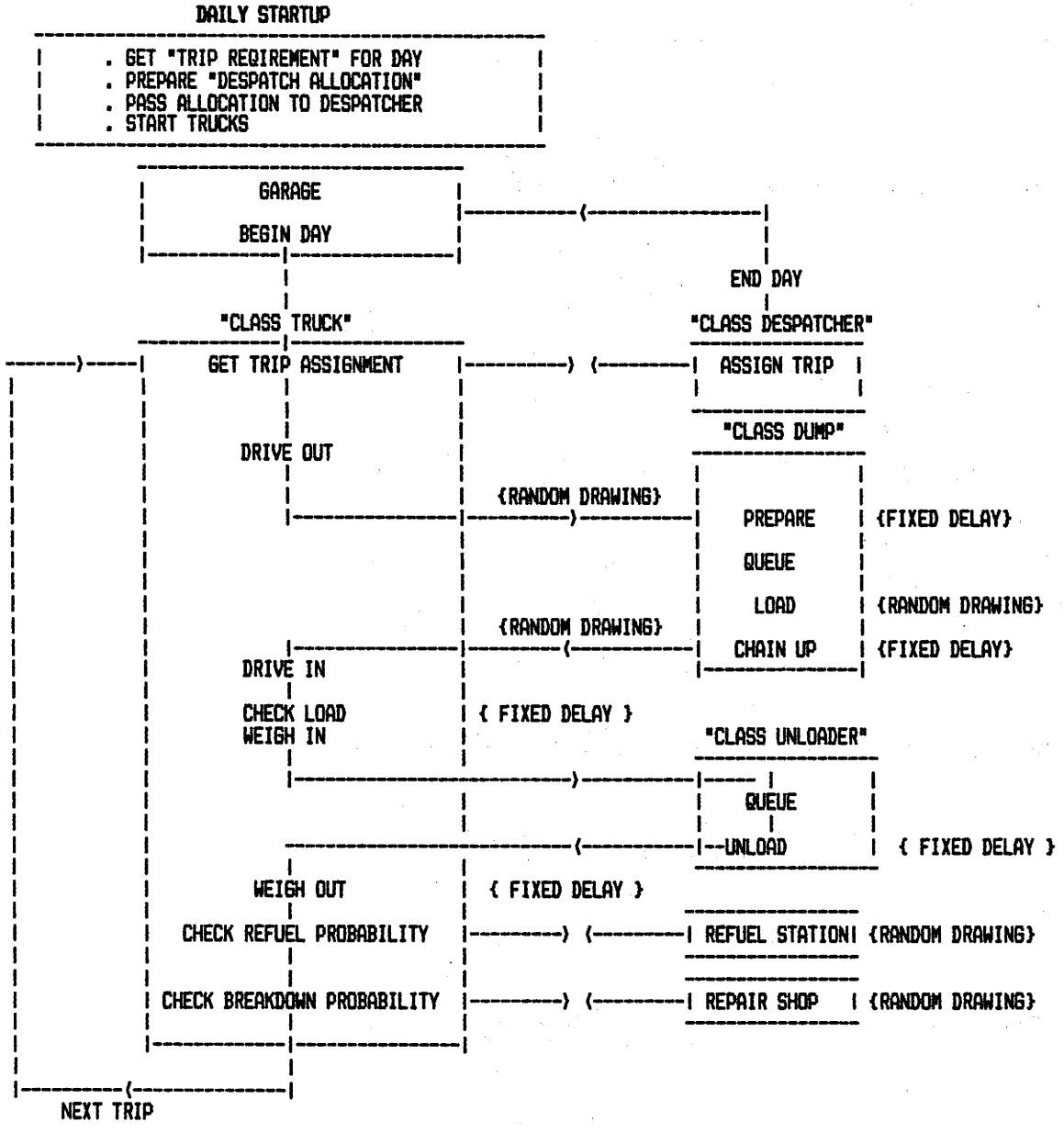
Landing operations are modelled as : a constant total time for preparation of the truck , queueing time for the loader , if any , loading times which are drawn from a discrete distribution supplied by the model user , and a constant time to chain up .

3.2.6 Travel Time

Driving times are drawn randomly from a generalized distribution based on the median values of round trip times . The median value was chosen as a more reliable

FIGURE 3.2

SCHEMATIC REPRESENTATION OF MODEL STRUCTURE



measure of trip time in small samples with extreme values. Total travel time is divided on an arbitrary ratio of one to two into travel out empty and travel in full .

3.2.7 Mill Terminal Times

The terminal process incorporated into the design model is a fixed delay for the inwards procedures , an unloader queue , a stochastic time for unloading and another fixed delay after unloading .

3.2.8 Refueling

Refueling is set to occur at a specified probability, tested after trucks left the terminal . The duration of the refueling activity , when it occurred , is drawn from a specified distribution .

3.2.9 Breakdown delays

Breakdowns on the road , although often overcome by the driver to enable delivery of the load on the day , may require a subsequent visit to the garage , for example , for tyre repair or replacement or checking of mechanical repairs carried out on the road . Modelling of breakdowns by a delay at the end of the trip was therefore adopted . There is a complete lack of data on frequency of occurrence and duration of breakdowns of log trucks for the reference fleet . This deficiency presented difficulty in the development of the design model . Given that some breakdowns are of quite long duration and that all break-

downs affect the despatching operations in a disruptive way , it was decided to incorporate breakdowns in the model rather than omit provision for them , but in doing so it was recognised that the procedures would be arbitrary .

Breakdown delays were set to occur at a specified probability , tested once each trip , immediately after unloading and any refueling . If a breakdown event occurs, the duration is selected from a specified distribution . When simulated repair time exceeds the time remaining in the simulated day any trips due for allocation to the truck are 'pushed back' for reallocation on the next day . A working day of 10 hours was adopted for the garage .

3.2.10 Trip Assignment

Trip assignment was modelled as a central despatcher working to a supplied despatch allocation of a full day's work with an assignment time delay of 0.01 minutes , effectively instantaneous . The system is modelled assuming the production of the full day's computer generated despatch allocation prior to the the start of the work day, to support the concept of decision aiding rather than decision replacement for the despatcher . The design allows the allocation table to be provided by either a 'direct assignment' module working with a prior fixed trip assignment recorded on computer disc file , or the heuristic allocation system to be added later .

There are two significant problems of control in trip assignment :

1. choosing and enforcing the end of the working day.

2. modelling the 'startup' time of each truck on each day .

Two working rules were formulated for control of these two aspects .

1. A trip was not started if its expected duration would cause the driver to exceed a defined maximum working day length . Expected duration was calculated from the median round trip times with a fixed allowance for unloading .

2. Trip assignment began at 5.00 a.m. daily . Trucks were started immediately if the projected time of return to the mill fell after mill opening time . Otherwise a start time was calculated by first drawing a target return time for the truck , drawn from a uniform distribution of the time between 7.00 a.m. and 7.45 a.m. A target start time was then calculated by subtracting the expected round trip time from the return time .

Target working day length is set at a specified maximum . Where an additional scheduled trip would cause the work day length to be exceeded the trip is 'pushed back' for reassignment on the next day .

Daily startup using the above rules differs in a number of ways from the real systems . The model assumes all trips start from the mill whereas in practice trucks are usually garaged at the homes of drivers or owners . Trucks may even be loaded on the previous evening and remain so overnight .

3.3 IMPLEMENTATION OF THE LUMPED MODEL

A basic objective in the design of the lumped model was to provide a general purpose simulation model as a base for experimentation . The model was programmed in three 'layers' using the prefix class capabilities of SIMULA 67 and comprises classes 'Basic' , 'Dataset' and 'Chipmill' .

The first , or outer layer , class 'Basic' Appendix B , contains the file opening and handling , data storage and the statistical procedures .

The second layer , class 'Dataset' , Appendix C , implements the procedures associated with maintaining and updating the daily despatch allocation tables . A relatively simple module transfers trip assignments for the day from lists on a computer file to the daily despatch

allocation table . This module is discussed below as 'direct allocation' , and was replaced with a much larger program element 'class Schedule' which implemented more complex heuristic algorithms for centralized despatching experiments .

The main layer , 'class Chipmill' , Appendix D , implements the fleet simulation model . Figure 3.2 presents the structure of this class . The model caters for up to 200 trucks , 50 'dumps'* with associated loaders, one unloader and one despatcher .

3.3.1 The Model Output

The primary model output is a set of tables describing on a daily basis the worktime , the total elapsed time, the breakdown time and the distance travelled together with an activity map which shows the fleet status at fixed time intervals . Some examples are provided in Appendix E. Secondary output , available optionally with each run , comprises a complete event dump which records each element transition for each truck and a trip summary which details the start time , time elements and compartment assignment for each trip , (also in Appendix E) .

* The term 'Dump' is used in the Eden area to describe a forest landing .

3.4 REVIEW

The adoption of a sound framework for model development and the use of modern computer science based approaches to program implementation were both identified as important to project success.

The objective was to produce a flexible and structurally simple model which would encourage confidence in its use . Almost all numerical quantities used to define the operation of the model , that is , the time distributions and constants and the various probabilities governing event selection , are readily changed or replaced to suit specific real world data .

In this study data for the operation , verification and validation of the model were derived from that collected in field studies of the Eden log truck fleet .

The collection and assessment of these data and the selection and use in the model of formulations based on these or other sources are discussed in Chapter 4 . The implementation of a lumped model requires that the model be verified and validated to ensure reliable output . Verification and validation are of course intertwined with the construction phase as well as final testing ,and are similarly discussed in Chapter 4 .

CHAPTER 4

APPLICATION OF THE SIMULATION MODEL

4.0 INTRODUCTION

Data required as input to the model includes

1. terminal times at the mill
2. refuel and tyre change time
3. breakdown frequency and duration
4. travel times to and from landings
5. terminal times at the landing .

The field studies of the log truck fleet at Eden (described in Chapter 1) were the primary source of data . The field data collection and analysis and adaption to the simulation model are described in this Chapter .

4.1 DATA INPUTS

4.1.1 Terminal Times

4.1.1.1 The Field Studies

Data were collected at the weighbridge site over seven days in December 1979 . Wood was accepted at the mill between 7.00am and about 10.30pm although few pulpwood loads were delivered after 8.00pm . The data were collected on

3-12-79 Monday	11:00 - 17:00
4-12-79 Tuesday	07:00 - 18:00
5-12-79 Wednesday	07:00 - 16:00
6-12-79 Thursday	07:00 - 18:00
7-12-79 Friday	07:00 - 14:00
10-12-79 Monday	11:00 - 17:00
11-12-79 Tuesday	07:00 - 18:00

The time of arrival at the mill was recorded together with the time of arrival and departure of trucks from each of the inward and outward weighbridges . The recorded times enabled calculation of queuing time , the weighbridge handling times and the mill unloading time from and back to the weighbridge .

Studies were conducted from the gatehouse on Wednesday 12th and Thursday 13th when use of the refuel/ tyre repair facility was recorded while still monitoring truck transit times from gate in to gate out .

The pattern of arrival times for the trucks at the mill is shown in Figure 4.1. A lull immediately following early arrivals is clearly evident . The drivers intimated that many of the early arrivals had been loaded the previous evening .

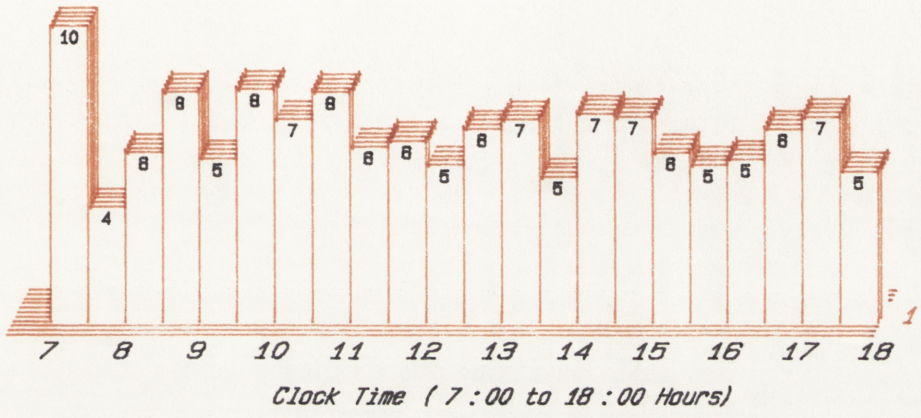


Figure 4.1 Half Hourly Truck Arrival Rate
Results of Weighbridge Study

Figure 4.2 Frequency (%) - Time Histogram of
Turn Around Time on Forest Landings

Results from the study of the weighbridge are given in Table 4.1 , as the mean times in minutes and the standard deviations , for waiting times at the gate and the weighbridge handling times for both weigh in and weigh out. Some effects of system disturbance are evident in these data . The longer weigh in times on Monday the third arose as a result of loader breakdown , with drivers seeking an explanation from the weighbridge operator for the abnormally long queue that faced them off the weighbridge. A breakdown in the outward weighbridge on Tuesday the 11-12-79 required both weigh in and weigh out to be conducted on the one bridge . Trucks were held in the gatehouse queue to allow free access to the bridge by both traffic streams , resulting in longer gatehouse times and increased outbound queuing .

TABLE 4.1

WAITING TIME AT GATE AND WEIGHBRIDGE HANDLING TIMES

DAY	WAITING TIME AT GATE (mins)(s.d.)	WEIGH IN (mins)(s.d.)	WEIGH OUT (mins)(s.d.)	NUMBER OF LOADS
3-12-79	2.05 (.47)	1.33 (.36)	1.16 (.25)	56
4-12-79	2.31 (.52)	0.74 (.21)	0.92 (.30)	124
5-12-79	2.47 (.80)	0.81 (.19)	1.34 (.28)	119
6-12-79	1.98 (.42)	0.82 (.18)	1.12 (.20)	134
7-12-79	1.85 (.53)	0.82 (.22)	1.30 (.31)	126
10-12-79	1.73 (.42)	0.84 (.19)	1.24 (.26)	48
11-12-79	3.11 (.72)	0.97 (.24)	2.11 (.46)	13
STUDY AVERAGE	2.29	0.87	1.34	

Unloading times were obtained from a three hour time study . A longer period or periods would have been desirable but the loader operators were reluctant to accept more than a short study . The observer recorded the departure of a truck from the weighbridge , the time of moving onto and off the apron and the operations of the unloader . Loads were unchained by the driver before the truck was called onto the apron , usually while in the unloader queue . During chipping , the unloader maintained log feed to the chipper deck as well as unloading the trucks and as many loads as possible were unloaded directly on to the chipper deck . Other loads were placed in the stockpile area . Trucks were signalled by the loader operator to move off the apron as soon as the load was lifted clear . The next truck was signalled to move on to the apron when the previous load had been either stacked or dumped and the loader repositioned for unloading .

The study results given in Table 4.2 cover the unloading of 48 trucks . The average active cycle time for the unloader was 3.2 minutes , that is a truck could move on to the apron for unloading every 3.2 minutes . Eighteen loads were stockpiled and the remaining 30 were chipped directly . The average unloading time (weigh in to weigh out) per truck was much longer (18 minutes) due to queueing . The average queueing time was about 10 minutes , that is a queue length of about 3 trucks . Unloading to stockpile was faster than unloading to the chipper infeed . The unloading rate during the study was

equivalent to a rate of about 17 trucks per hour . It is apparent , both from the daily arrival pattern (Figure 4.1) and from observations made during the seven days of the weighbridge study that the unloading system , as it then operated , was close to or at its maximum capacity .

TABLE 4.2

RESULTS OF LOADER TIME STUDY

ELEMENT	TIME (mins)(s.d.)	NUMBER OF OBSERVATIONS
QUEUING TO UNLOAD	10.38 (6.2)	42
TRUCK ON TO APRON	0.36 (.20)	48
UNLOAD TRUCK	0.70 (.34)	48
TRAVEL TO STACK	0.62 (.34)	18
STACK LOAD	0.19 (.06)	18
TRAVEL TO CHIPPER	0.58 (.13)	30
WAIT TO DUMP LOAD	2.26 (1.6)	9
DUMP LOAD	0.60 (.26)	30
TRAVEL TO TRUCK	0.39 (.15)	48
INTEREFERENCE AND DELAY	2.69 (1.56)	5
LOADER IDLE	4.27 (3.59)	10
AVERAGE ACTIVE CYCLE	3.42 (1.58)	48

Any disruption was quickly reflected in an increase in queue length . It was also apparent that the management of the combined truck unloading-chipper feeding operation is a complex problem , affected by the rate of

chipping required , the current location of the stockpile relative to the apron , the skill of the loader operator and the success of the particular operating strategies employed .

That the unloading time (queue and unload) varies over time (day to day) is evident from the data on total time in the mill presented in Table 4.3 . The data are from the seven days of the weighbridge study , the two days of the gatehouse study and the two sets of tachograph data . The first , Set A , was collected from one truck over several months prior to this study . The second , Set B , was collected from five trucks during the July study . The unloader breakdown of 3/12/79 is also evident in the 'mill time' figures . Likely causes of the large variation are changes in the chipping load and the skill level of various operators .

Taken together the weighbridge data (Table 4.1) and data for unloading operations (Table 4.2) provide the basis for assessment of a multi-stage queueing process involving four sequential servers . The combined active service time is eight minutes . Unloading is the slowest process and determines the steady state processing time of the system at about 17 trucks per hour (3.2 minutes per truck) . For example , in the case of two trucks arriving at the gate together , the second truck would usually have some queueing at the loader and the first truck would depart outward from the weighbridge before the return of the second truck .

TABLE 4.3

AVERAGE TOTAL TIME AT MILL

STUDY	TIME (mins)	NUMBER OF OBSERVATIONS
MILL TIME STUDY		
1	47.7	56
2	15.3	124
3	22.9	119
4	14.6	134
5	20.9	126
6	18.0	48
7	28.1	131
8	21.1	132
9	25.9	121
10	16.8	97
AVERAGE OF MILL TIME STUDY	28	1087
TACHOGRAPH SET (A)	27	116
TACHOGRAPH SET (B)	27	161

4.1.1.2 Adaption of the Field Studies to the Simulation Model

Examination of the terminal operations showed a set of four sequential serving processes , inspection at the gate , weighing in , unloading and weighing out . Examination of the mean and variation data for these processes shows they were 'dominated' by unloading , that is , the relatively small variation of processing times for each of the servers means that stochastic variation in the two quicker preceding processes would often be absorbed in unloader queuing . These two processes , that is inspection and weighing in , were modelled as an equivalent fixed delay , based on their observed mean value of three minutes . Similarly , the short weighout time was modelled as a fixed delay of one minute . The low absolute magnitude of the times for these processes means that they have little effect on the comparatively long round trips . The terminal model is then two fixed length delays , a queue for unloader , a stochastic unloading time , followed by a fixed length delay .

Thus the queueing behaviour of the terminal model depends on the truck arrival pattern , and the distribution of unloading times . The relatively short time study indicated an unloading time of 3.2 minutes per truck . Insufficient information was collected for a detailed analysis of the distribution of unloading times and a normal distribution with mean of three minutes and a standard deviation of one minute was selected for application in

the model . Integer values were used for the main time index to reduce computer processor demand and storage requirement . A simple formulation may better serve the objective of gaining user confidence in the overall model.

4.1.2 Refuel and Tyre Repair

4.1.2.1 Field studies

Data on the frequency and period of use of the refuel and tyre facility located one hundred metres from the gate house were obtained on two days of the field study . It was not possible for the observer to distinguish clearly between the two activities of refuel and tyre replacement as both were conducted on the same apron . For this study, both are regarded as a similar stoppage for replacement or replenishment of consumables . The visits recorded totalled 183 while in the same period 349 loads were delivered to the mill , that is , trucks stopped at the facility about once every two trips . The average delay was 11.5 minutes . It was apparent that the delay at the facility also included time associated with the drivers' talking . Such 'social' delays are discussed below in connection with landing times .

4.1.2.2 Application of the field studies to the Simulation Model

Little data were available to support the development of a refuel model based on either distance travelled or engine hours in relation to fuel consumption and tank capacity . While it is accepted that such development would improve the realism of this aspect of the model , it was not feasible in this study and its contribution to the overall objectives slight .

Again , for simplicity , a normal distribution was prescribed with a mean of 10 minutes and a standard deviation of three minutes . A more detailed study and analysis would provide a better understanding and model of the refuel process . However, its numerical impact on the model would be relatively small .

4.1.3 Bush Loading

4.1.3.1 Field studies

Data on the loading time of trucks at the landings were obtained from two separate field studies and from tachograph charts . Loading time at landings was divided into three elements

1. prepare to load , secure load
2. loading time
3. nonproductive and idle time .

The data for the 58 observed loadings is summarized in Table 4.4 . Marked variations in the loading times are evident on the thirteen landings studied . A range of both machine size and operator skill and the size and organization of the landing appeared to contribute to this variation . However , observations also suggested that the major proportion of the variation in the turnaround time was from unproductive and idle time . Common causes included lack of wood ready to be hauled and the driver sharing a meal break with the logging crew .

TABLE 4.4

RESULTS OF LOADER STUDIES

LANDING NUMBER	PREPARE TO LOAD AND SECURE (mins)	LOAD (mins)	DELAY (mins)	TOTAL (mins)	NUMBER OF LOADS
1	8.5	21.7	8.0	36.8	6
2	7.2	33.7	7.3	48.2	6
3	7.3	24.7	18.7	51.9	3
4	5.3	27.0	3.6	36.3	3
5	6.2	14.4	7.8	32.8	5
6	7.3	14.8	15.8	37.8	4
7	8.5	22.0	84.1	114.5	2
8	7.0	12.0	6.6	25.6	3
9	7.2	20.4	4.1	31.8	7
10	9.0	21.0	10.0	40.0	3
11	3.6	28.2	23.6	54.5	5
12	4.6	19.0	0.6	24.2	5
13	5.2	19.7	9.7	29.7	6
	6.6	21.7	11.2	39.5	58

Distributions of turnaround times from the three data sources are presented in Figure 4.2 . Data from tachograph Set A were drawn predominantly from one landing and the turn around times are much longer than those from both Tachograph Set B and the main landing study . There are insufficient observations for more detailed statistical analysis .

4.1.3.2 Application of the Field Studies to the Simulation Model

Table 4.4 and Figure 4.2 show considerable variablity in turnaround times at the landing . The Eden loading operations would , in fact , require significant research for the reliable prediction of landing times . Indeed , prediction may only be possible in a general way due to the influence of human and social factors on landing delays in a relatively 'unstressed' transport system . Research into landings or trucks under 'production stress' would provide more insight into technical capacity .

The landing operations were modelled in terms of the productive elements observed , i.e. prepare to load , load after queueing if necessary ,and secure the load . Again the numerical representation selected relied on rounded numbers as an added indication to model users and clients that the accuracy of the input data and the assumptions about data distribution are suspect . However, the planned use of the simulation model is for a

series of comparative runs with different allocation procedures , but using the same data for terminal times , refuel , landing times and travel times and it was assumed that 'synthesised data' were adequate for comparisons .

4.1.4 Travel Time

4.1.4.1 Field Studies of Travel Time

Data from tachographs provided the only direct information on travel time . The tachographs recorded engine revolutions , road speed and distance and vibration on a circular clock driven chart . A knowledge of the road network and truck operations allows the travel characteristics of trucks over some specific route segments to be obtained . Four segments repeatedly used were identified . The three major road classes in the network were represented ; main bitumen highway , gravelled secondary road and earth compartment access track . The data given in Table 4.5 indicates the different speeds attained on these different road classes . An important feature of the data is the low variation in the travel times .

Data for round trip time to each compartment were collected over the month of July 1980 . All truck entry and exit times for the month were recorded on the truck delivery docket by the weighbridge staff . These data combine travel time, landing time and any delay or idle time occuring during the trip . About 4000 loads were

delivered during the month . The data were punched on to computer cards and programs written to process the resulting data files . About 80 compartments were visited by log trucks in the month , 69 provided more than ten loads and 23 more than 50 . The travel time for the first trip of each day for each truck was not available from this recording method as the daily start time was unknown but about 1600 roundtrips were identified .

TABLE 4.5

RESULTS OF TACHOGRAPH STUDY

SURFACE	DIRECTION	MEAN TIME	STANDARD DEVIATION	NUMBER OF LOADS
Bitumen	Unloaded	24.7	1.31	63
Bitumen	Loaded	31.8	1.59	83
Bitumen	Unloaded	12.9	0.79	233
Bitumen	Loaded	16.6	0.68	299
Gravel	Loaded	19.3	1.92	89
Earth	Unloaded	6.2	1.50	89
Earth	Loaded	7.8	1.27	88

4.1.4.2 Application of the field studies to the Simulation Model

Data for the roundtrip times were grouped into separate files , one for each compartment visited . Basic statistical information was calculated for all compartments with more than three trips , including the number of trips , the mean and median , the 25th and 75th percentile

values and a 'mid mean' statistic calculated as the arithmetic mean of all observations between the 25th and 75th percentile . This latter statistic was calculated to provide a better measure of the centre of the distributions in the presence of extreme tail values and as an alternative to the median . Later analysis showed the distributions to be skewed and the median was accepted as the more appropriate measure . Appendix F presents a summary of these data .

The data for each of the compartments was highly variable and in the light of the loading studies and the low variation of travel times along selected sections of the road (as found from the tachograph charts) it was concluded that the major source of variation was in the landing turnaround time .

The impact of this variation in trip time could be significant in a study which has equity between trucks as a criteria for evaluation of system performance . Preliminary analysis of the compartment data revealed that few had sufficient recorded round trip times to obtain a satisfactory distribution of these times . A method of projecting distributions for the remaining compartments was needed . The method developed required fitting individual distributions of a common type to the trip time data for each compartment with adequate observations and deriving regressions to fit the parameters of the common distribution to the observed median round trip time . Median estimates for round trip time were available from

the field study for all compartments and the regressions could thus be used to project individual distributions of trip times for each of the 80 compartments .

The MLP statistical package ,Ryan et al (1979) , was used to fit a variety of continuous distributions , including the normal , lognormal and the three parameter lognormal to the trip times for individual compartments with sufficient observations . The goodness of fit was measured using the chi squared statistic provided by the package . However , there are difficulties associated with the derivation of this statistic where class interval populations are low , MAISEL and GNUGNOLI(1972 , p85) and class intervals were chosen to ensure a minimum of five observations per cell . The three parameter lognormal produced the lowest relative chi squared values for almost all data sets and was selected as the model for the estimation of a single common distribution to provide round trip time estimates for the simulation model . The table of results is presented in Appendix F .

Parameters of the distribution were obtained by separately regressing the origin shifts , means and variances obtained from the MLP package , on the median trip times for the thirty six data sets . Regressions were fitted using the GLIM package ,Royal Statistical Society (1977) . The results are presented in Table 4.6 . The 'a' parameter , the origin shift , provides the location of the whole distribution along the X axis , the 'm' parameter provides the central measure of the displaced

distribution and the 's' parameter measures its dispersion, ('m' and 's' being logarithms of the mean and the variance parameter of a 'normal' distribution) .

TABLE 4.6

RESULTS OF REGRESSION ANALYSIS FOR TRIP TIME PARAMETERS

a =	3.22	+ .7452 *	MEDIAN	R*2	0.77
	(17.17)	(0.069)			
m =	2.92	+ .0034 *	MEDIAN	R*2	0.17
	(0.319)	(0.0012)			
s =	.7179	+ .00045 *	MEDIAN	R*2	0.05
	(.1364)	0.00055)			

Note : Figures in brackets are Standard Errors

It was hypothesised from the field observations that the majority of the variation in round trip times was due to delays on the landing . If this were so then the actual size and shape of the trip time distributions as measured by 'm' and 's' should be independent of actual travel time since they would be dependent on landing operations . However , the 'a' parameter should be strongly dependent on the level of trip time since it is a location parameter , representing the minimum trip time intercept .

The hypothesis is essentially supported by the regressions shown in Table 4.6 . In the 'a' regression the median provides a good fit while in the 'm' and 's' regressions both levels of fit associated with the regressions are low although the 'm' equation is statistically significant . The 'm' and 's' slope coefficients are

numerically small and the standard errors associated with the median parameter estimates are high indicating an insignificant contribution to fit and to the estimator by that parameter .

There are several important limitations to the application of predictions based on a derived generalized distribution . Most importantly , the analyses were conducted without regard to truck engine power or driver habit . Thus , while the particular objective of the model's development may be met , i.e. a consistent generation of continuous distributions of trip times for comparative evaluation in the use of the simulation model, a demand for accurate travel prediction may not . For example , the real world selection of a comparatively powerful truck to visit a compartment whose median trip time had been based on trip times generated by a slower truck will cause erroneous prediction . Secondly , it seems likely that variations in landing times are the major contributors to the estimation of the 's' or variance parameter . It was observed during the study that some landings appeared to be more efficient than others . Comparing Landings 5,8,9,12 and 13 with 3,7,and 11 in Table 4.4 shows that the true variation of landing times may be different from the 'average' level estimated by regression . Again , while the generated estimates may be satisfactory for an experimental program which uses the model in a comparative way the particular distribution generated for an individual landing may be inaccurate .

Nonetheless , to obtain a specific round trip time distribution for a compartment , its recorded median time was used in conjunction with the regression equations in Table 4.6 to provide the three parameters of the displaced intercept lognormal distribution . Specific times were then recorded by random drawing .

4.1.5 Breakdown Frequency and Duration

4.1.5.1 Field Studies

The collection of data on the frequency of occurrence and duration of breakdowns was not feasible in connection with this study . Suitable records of breakdowns over a long period were not kept by the truck owners and breakdowns during the short study period were not recorded . Some component of truck breakdown time is already present in the round trip time , since all breakdowns that are overcome on the road with the load still delivered on the same day would be unreported , and appear as an unduly long roundtrip time .

4.1.5.2 Simulation of the frequency and durations of breakdowns

Breakdowns have a very wide range of duration . Discussions with the drivers during the study suggested that as many as five days a year may be lost in repair ,

service or modification to truck or trailer . A breakdown probability of 0.03 was selected , and for duration , a normal distribution mean 200 minutes and standard deviation 200 minutes . At an average delivery level of two trips a day , the number of days lost would average about four to five .

4.2 VERIFICATION AND VALIDATION OF THE SIMULATION MODEL

4.2.1 Verification of the Simulation Model

Verification of the simulation model was intertwined with the model construction phase . Final checks were a detailed examination of the model output , including the various forms of output files and run logs (detailed in Chapter 3 , Section 3.1) . In particular , the data produced by the secondary output system of the model , which had been included in development to aid debugging and verification , was used . It produced detailed data on the occurrence of each simulated load assignment and time elements for each completed load . Examples are provided in Appendix E . As a formal verification the assignment of all trips for two simulated days were checked in detail against the daily demand input , simulated breakdowns checked for garage entry and exit , and the consistency of behaviour of the reported loader , unloader and travel times was examined .

A variety of status and warning messages and data were included in the VDU activity monitoring display , which is produced while the simulation is in operation . A number of abnormal conditions were detected from this display during program development when a very detailed 10 minute reporting procedure was used . The principal model output is a user defined report on truck fleet status at fixed intervals throughout the simulated day . Examples are provided in Appendix F . Both a warning and a system status display of this type could readily be developed to assist a despatcher .

It was concluded that the functioning computer model correctly translated the 'lumped model' .

4.2.2 Validation of the Model

Validation is acknowledged to be difficult for many stochastic models. Zeigler(op cit) makes the point that a model should only be validated against its experimental frame and the objectives set during its construction .

4.2.2.1 A Validation Experiment

There were several difficulties in validating the model . No data were available which were independent of those used to derive the travel distributions used in the model. Another problem was that complete data on the actual performance for the month were not available

because actual daily startup times were unknown. An estimate of startup time for each truck for each day was synthesised by deducting the observed median trip time for the compartment of the first load from the recorded time of the first arrival on that day (obtained as described previously from the time of arrival with the first load) .

A validation experiment was conducted to gain some insight into the model's level of performance by running the model on the July study data. Thus the model attempted to replicate the performance of the reference fleet for the study month .

The data produced by the model on individual truck monthly total trip times were compared with estimates obtained by adding the median value for the first trip to the rest of day performance recorded in the July study. The two resulting distributions are presented in Figure 4.3 . Since trucks in the reference fleet worked widely differing numbers of days and trips , the monthly total trip times were divided by the numbers of trips delivered to derive an average trip time .

Results from this comparison are presented in Figure 4.4 in the form of an error distribution of the difference between the average time per trip for each truck predicted by the model and that derived from the July observations .

The simulation model consistently underestimated the total work time , on a per trip basis by about 17 minutes.

Investigation of the modelled mill terminal performance revealed an average mill time of 12 minutes compared to an average unload time from all observed data of 28 minutes . Figure 4.5 presents the frequency distribution for mill times for the validation run and the field study data . The underestimation could be rectified by more accurate modelling of the terminal times . However , since the terminal times model does not detract from the usefulness of the simulation model in a comparative role associated with the experimentation , it remained unchanged . The loader study was necessarily short and the observed average unloader time of 18 minutes obtained in December 1979, was possibly not representative of the operations in July, 1980 , the period of the validation experiment .

The underlying philosophy of model design was for a simple open structure allowing the structural validity to be readily determined . The validation run provided assurance that the model was structurally valid . In other experimental applications , the parameters determining each of the principal processes (load , unload, travel etc) could be readily modified as required .

It was concluded that the simulation model adequately described the system modelled and that comparative performance of the simulation model under experimental input would produce valid results .

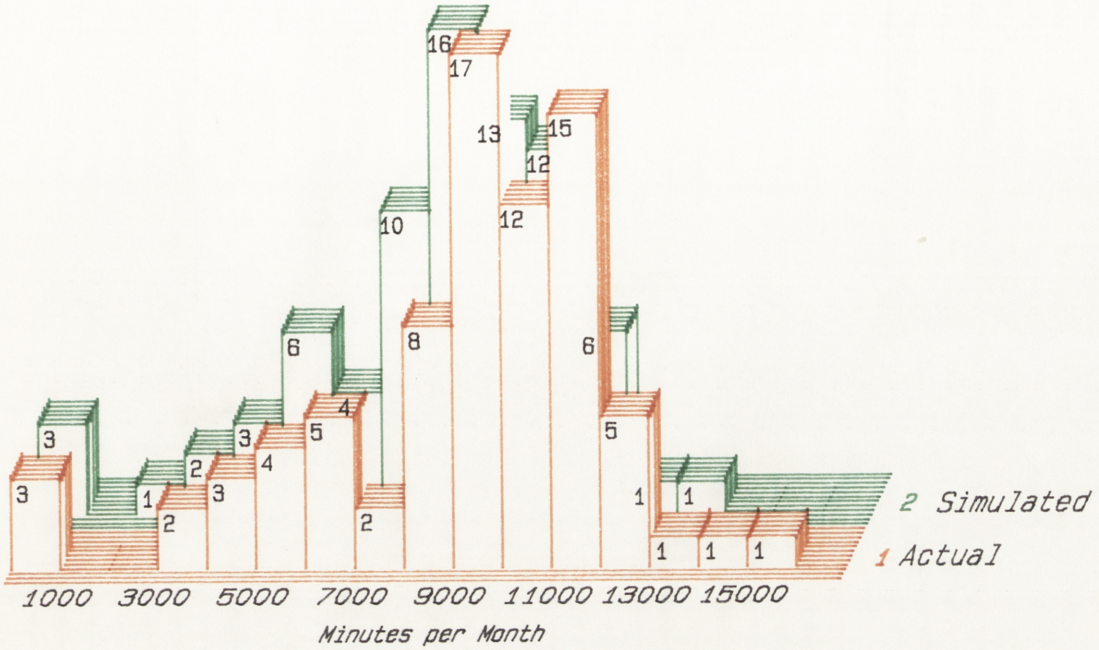


Figure 4.3 Frequency (%) - Time Histograms of Truck Monthly Total Times comparing Estimated Actual with Simulated

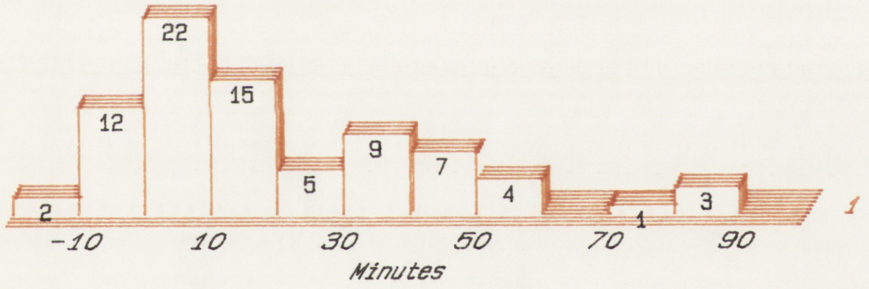


Figure 4.4 Frequency (%) Time Histogram of Differences in Average Trip Time per Truck between Actual and Simulated

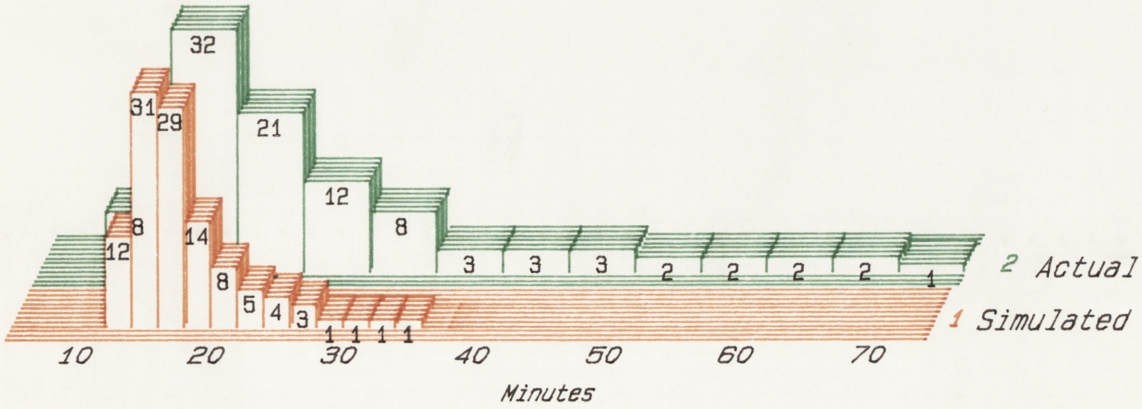


Figure 4.5 Frequency (%) - Time Histograms of Mill Unloading Times comparing Actual with Simulated

4.3 SUMMARY AND REVIEW

The major objective of this section of the work was to develop a flexible simulation model of truck fleet operations as a major tool for experimentation . A large SIMULA program implementing such a model was successfully developed and verified . A number of limited studies were undertaken to obtain reference data on truck operations to aid implementation of a more realistic model . While a number of difficulties were experienced in attempting to obtain definitive data as base input for the model , the programmed simulation model appears well suited to use in comparative experiments and was flexible , in that the mathematical relationships in the input data can be readily modified .

CHAPTER 5

APPLICATION OF THE HEURISTIC ALLOCATION PROCEDURES TO THE SIMULATION MODEL OF A TRUCK FLEET

5.0 INTRODUCTION

To evaluate the likely effects of the introduction of a centralized despatch system to a truck fleet , the heuristic allocation procedures were tested with the simulation model . Testing within the framework of the model allows evaluation of the heuristic procedures in two important respects , firstly in its capacity to cope with the stochastic variation introduced in the simulated performance of the fleet and secondly with respect to variation in the pattern of daily demand throughout the test period .

5.1 DATA REQUIREMENTS

The simulation model enables generation on a stochastic basis of round trip times for particular landings . The heuristic allocation algorithm was tested on the basis that wood was already available at the

landing . In practice this would correspond to trips being considered for assignment only if the availability of wood were notified on the previous day . With wood assumed to be available , the additional information required to test jointly the allocation algorithm and fleet simulation model are the landings from which the wood is to be hauled, the number of loads to be delivered daily and the trucks available.

The use of 'real' data in the testing of the allocation procedures on the simulation model assists acceptance of the proposal for a centralized despatch system and the additional inputs needed were derived from the complete delivery schedule recorded in the July study period and used in the model validation (Figure 4.1 ,p90) .

A data set for a smaller full time haulage fleet was synthesised by extracting from the July data all trucks which missed delivery on more than three days in the month and all trucks with monthly work totals less than 40% of the average. The workload for several of the trucks removed on this basis could be linked together to provide data for additional 'phantom' trucks because the patterns of work days were not overlapping. The resultant data set (combining observed and synthesised 20 day trip sets) covered a full time fleet of 80 trucks .

There were several reasons for extracting trucks from the 106 recorded and then combining some of those extracted to give a fleet number based on full time operation. A basic assumption in the development of the scheduling system was that trucks were available for full time work and the fundamental purpose of the heuristic is to allocate the work evenly . Thus comparison with a fleet including part time trucks and exhibiting a wide range of monthly average truck total trip times would be unfair and provide little insight into the operations of the heuristic . There is , however , no apparent reason precluding subsequent development of more complicated 'equity' procedures capable of accepting a range of target utilization levels rather than the 'equality' level accepted here . The objective here is to demonstrate clearly whether or not centralized despatching could be advantageous and simplicity is helpful in this respect .

A further difficulty in using the observed data arose due to a major mechanical failure in the chip mill in mid July which closed the plant for a number of days. There had been some warning of a closure and the operations of the truck fleet were abnormal before and after the closure. Data for the day prior to and the two days following the closure were therefore not included in the synthesised data set . The daily number of deliveries adopted is shown in Figure 5.1 . Total deliveries were 2984 loads over a period of 20 days.

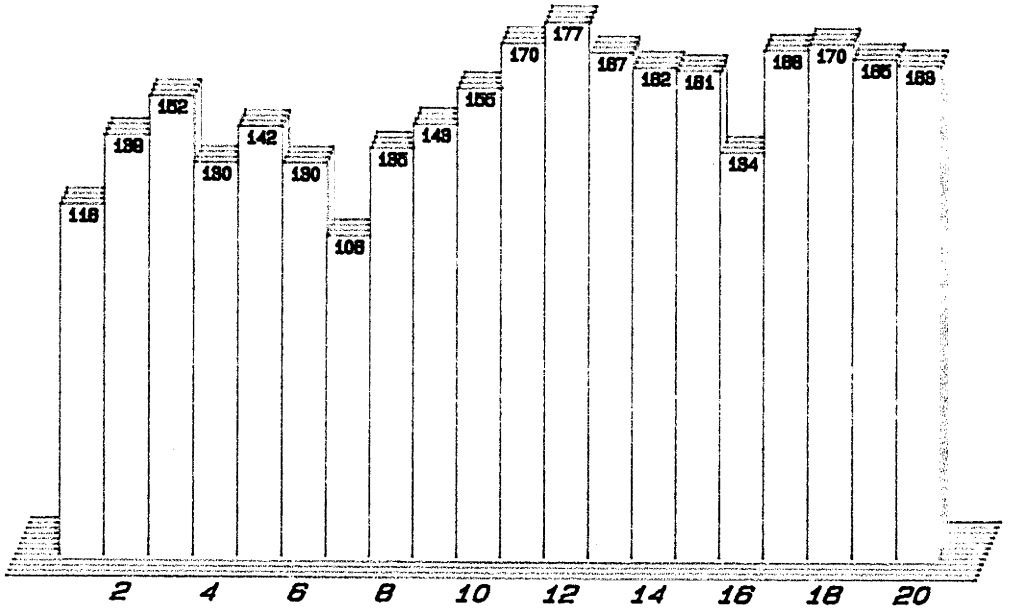


Figure 5.1 Daily Total Deliveries for the 80 Truck Synthesised Data Set over 20 Days

5.2 TESTING OF THE HEURISTIC ALLOCATION ALGORITHM

The testing of the allocation / truck fleet simulation system using the 20 day synthesised compartment and load requirement schedules was conducted in three phases.

1. A benchmark simulation without heuristic trip reassignment .

2. Simulation using the heuristic allocation algorithm for reallocation of trips between trucks with a variety of fleet sizes .

3. A more detailed study of the capacity of the fleet under a steady load pattern.

5.2.1 Evaluation of the performance of the fleet under 'observed' allocation - a benchmark

The simulation model (described in Chapter 4), was run using the exact pattern of load deliveries recorded in the synthesised data set to assign work to trucks without any trip reallocation . With regard to trip assignment , the modelled sequence was :-

. Before the start of each simulated day a 'trip requirement' list was produced for each truck . The list was compiled as , firstly , any unstarted trips from previous days and secondly , the sequence of new trips for the current day recorded in the synthesised data set for that truck . The resulting 'trip requirement' list was then simply adopted as the 'despatch allocation table' for the day . The effect of this procedure was to ensure that the simulated individual trucks completed the specific list of trips recorded in the systhesised data set . No reallocation of trips between trucks was undertaken .

. Trips were then assigned to trucks by the 'despatcher module' in accordance with

the despatch allocation table at the beginning of each truck's trip cycle (Figure 3.2 , p 80) , that is , at the beginning of the day and as each simulated truck returned from the weighbridge .

. A trip was not assigned to a truck when the remaining work time for the day was insufficient in relation to the expected trip time calculated as the sum of the median round trip time and an allowance for mill time .

This benchmark simulation run represented the likely performance of a full time fleet of 80 trucks if the work allocation between trucks was as observed in the July study month . The particular results obtained are, of course still the product of the stochastic variations introduced by the simulation model .

The distribution of total trip times for each truck in the 80 truck benchmark run is shown in the front row of Figure 5.2. The standard deviation of the total monthly trip times for individual trucks (the principal measure of 'equity') is reduced by 70% over that of the 106 truck fleet simulated for the validation run (Chapter 4) . The reason for the improvement is the elimination of the part time trucks .

5.2.2 Evaluation of the heuristic allocation model

The major programming work required for these evaluations involved the embedding of the heuristic trip reallocation program code as described in Chapter 2 within that of the fleet simulation program . The SIMULA code for the modified Class Dataset is given in Appendix G . Since the program code was copied directly from the earlier program , verification consisted only of checking the correctness of the links to the data areas used for input and output by the module . The modelled sequence used in trip assignment then becomes:-

- The 'trip requirement list' is prepared in the same way as before , however the prepared list is now passed to the heuristic reassignment program module where the algorithms described in Chapter 2 are applied .
- The resulting 'daily despatch allocation table' is made available to the despatcher module for trip assignment to trucks as before .

The other modules of the simulation program remained the same . The same round trip , breakdown and unloading times were used as in the benchmark to ensure a valid comparison and evaluation of the simulated effects of heuristic allocation .

The programmed allocation procedures can now be

more clearly seen as the critical innovation in the development of a computer aiding system . Its role in an operational system was simulated as the preparation of a prototype despatch table , fitting in between the assembly of the daily 'trip requirement' list and final modification and acceptance of the 'despatch allocation table' by a despatcher .

The simulation runs sought to answer two major questions :

How effective was the heuristic ?

What would be the effect on system performance and equity of a reduction in the total fleet numbers ?

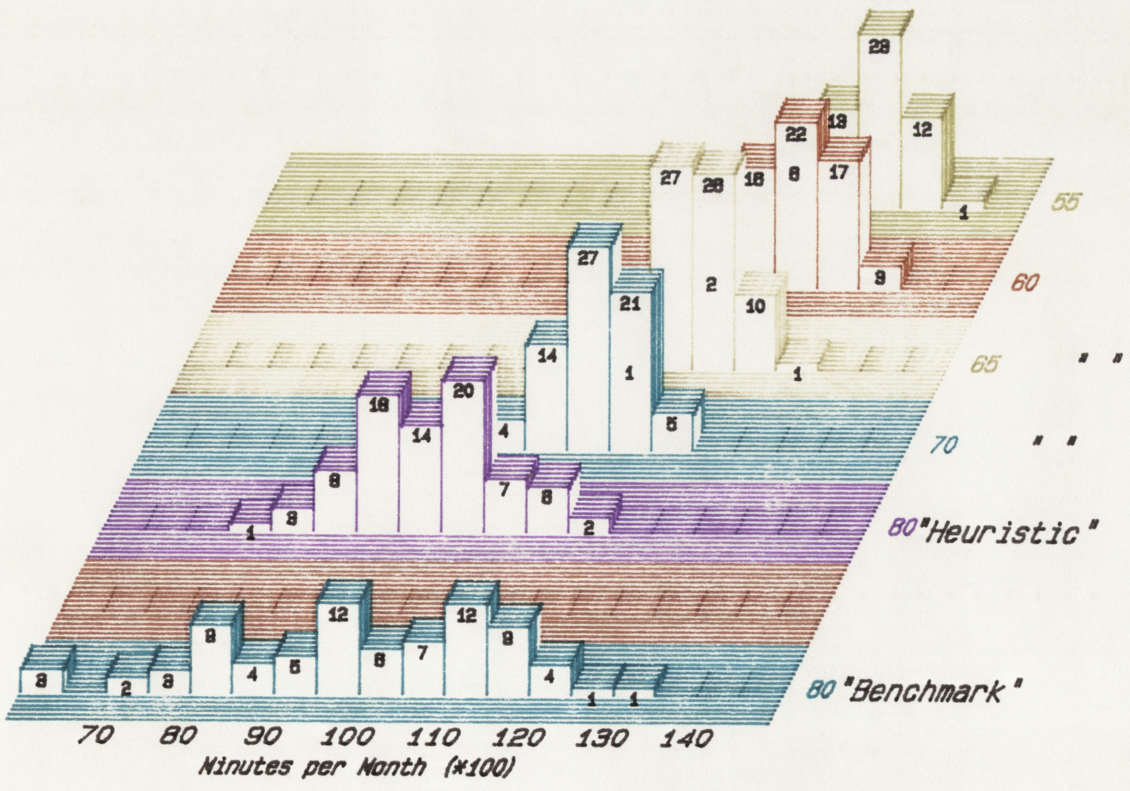


Figure 5.2 Frequency (%) - Time Histograms of Truck Average Monthly Total Time

5.2.3 Fleet size simulations

The performance of fleet sizes of 80, 70, 65, 60 and 55 trucks were simulated with the same synthesised daily trip requirement pattern used for the truck benchmark . The statistics of the results of these runs are given in Table 5.1 and the distributions of monthly total times are shown in Figure 5.2 .

TABLE 5.1

STATISTICS OF DIRECT AND HEURISTIC ALLOCATION ON SIMULATED FLEETS

TRUCK NUMBERS		MEAN MONTHLY TOTAL TRIP TIME (mins)	STANDARD DEVIATION (mins)	RANGE (10-90 percentile) ABSOLUTE (mins)	RELATIVE TO MEAN %
80	Direct**	10426	1703	4025	40
80	Heuristic	10340	856	1982	19
70	Heuristic	11818	467	1337	11
65	Heuristic	12622	387	1124	9
60	Heuristic	13286	477	1496	11
55	Heuristic	13683	423	1356	10

** Benchmark simulation run , direct allocation

5.2.3.1 Truck Total Trip Times for One Month

The level of variation in the truck total trip times for the month , as measured by the standard devia-

tion - the principal measure of work equity used in this study - was reduced between 50% and 80% by the heuristic allocation algorithm compared with that for the benchmark . Although not quite as great as the reduction achieved in the deterministic tests comparing the heuristic with random allocation , described in Chapter 2 , the reduction is still very large . Another measure of the spread of the distribution is provided by the range statistic , calculated here as the 10 to 90 percentile range . At the 65 truck level , 90 % of the trucks were within 4.5 % of the mean , compared to a spread of 20 % above and below the mean for the benchmark fleet of 80 trucks , a substantial improvement in the equitable distribution of work between trucks .

5.2.3.2 Backlog

The dynamic response of the transport system to variation in daily demand is an important aspect of system performance . Adoption of the synthesised data set, based as closely as practicable on the recorded sequence for July 1980 , provides an opportunity to obtain insight into system behaviour under heuristic trip reallocation with dynamic daily demand variation . The simulation of a variety of fleet sizes provides information on the effects of increasing overall workload .

New daily demand over the 20 days can readily be divided into two periods (Figure 5.1) . A low ,

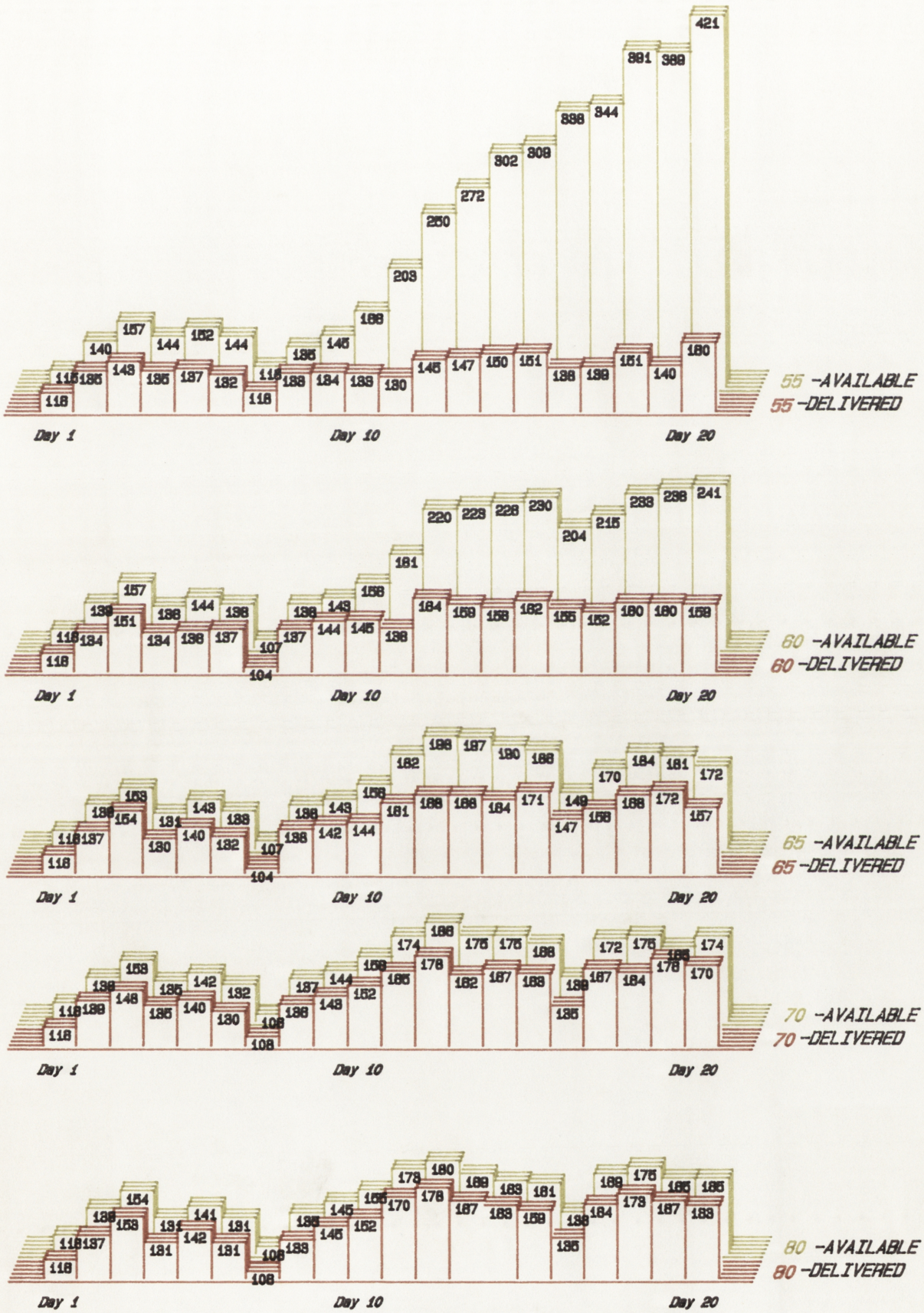


FIG 5.3 Trips Available and Delivered Daily for Five Simulated Fleet Sizes

variable level of demand characterised the first ten days, (about 135 trips per day) , and a higher , more constant demand the second period (164 trips per day).

In Figure 5.3 * each pair of walls represents a simulated fleet size with the histogram of the front wall representing the deliveries achieved and the rear wall of each pair the total trips to be allocated each day , that is the backlog to the start of the day and the new daily demand . Statistical data associated with these histograms is presented in Table 5.2 .

TABLE 5.2

AVERAGE DELIVERY AND BACKLOG PERFORMANCE OF DIFFERING FLEET SIZES

NUMBER OF TRUCKS IN FLEET	TOTAL DELIVERY (trips)	AVERAGE DELIVERY (*)	AVERAGE BACKLOG (trips)
80 Direct**	2984	1.88	1
80 Heuristic	2984	1.88	1
70	2982	2.12	5
65	2968	2.28	10
60	2902	2.33	34
55	2779	2.51	60

* - trips per truck per day
** Benchmark simulation run , direct allocation

A fleet size of 55 trucks is clearly unable to maintain deliveries . The backlog grew to 260 trips or

over 110 truck days in the second ten day period . The 60 truck simulation generated a backlog of 78 trips or 34 truck days in the same period while that for the 65 truck fleet was only 15 trips or 7 truck days . The 70 truck fleet simulation was essentially free of backlog . These results indicate that a fleet of about 65 trucks is required to maintain delivery with this pattern of demand without significant backlog while as few as 60 trucks could cope if the first period of lower demand were to reoccur again immediatly allowing the backlog to be cleared . The effect of the heuristic algorithm in smoothing out wood delivery is evident .

The large backlog built up by the 55 truck fleet , mostly in the last 10 day period , is about two days of production whereas there was only a half a day of production accumulated by the 60 truck fleet. Knowledge of longer term variations in total haulage demand and maximum practical levels of stockpiles on the landing become important to the transport manager in determining minimum feasible fleet size . Acceptance of high backlog levels would allow relatively long periods for demand to be 'averaged out' and a smaller fleet .

* 100 trip units have been subtracted from all histograms in Figure 5.3 , exaggerating the vertical scale , the to better illustrate the variations in demand and backlog and facilitate comparison of treatments on one sheet .

Three statistical measures are proposed to describe performance in a more general way .

1. 'Allocation Pressure' which provides a measure of the workload facing the allocation system and is independent of fleet size . It is calculated as either the average number of trips available for allocation per truck per day , or more generally , as the expected average working time required to complete all available trips as a proportion of the theoretically available time (maximum working day) . The two calculations are readily linked in the this case since the maximum working day (720 minutes) and the average trip length (275 minutes) are both known . The number of trips available for allocation per truck per day is used below .

2. 'Throughput' calculated as average delivery per truck per day , or more generally , and in the sense used in Chapter 2 , as average achieved work time per truck per day as a proportion of work time available - a measure of performance independent of fleet size .

3. 'Daily Backlog' generated , which is the difference between the allocation pressure facing the trucks at the start of the day and the daily throughput . It can similarly be

expressed as trips per truck per day or a proportion of daily work time .

Calculations of point values of both allocation pressure and throughput for the daily data (Figure 5.3) show wide variation between a minimum allocation pressure of 1.35 trips (on day 7 for the 80 truck fleet) and a maximum of 7.65 trips (on day 20 for the 55 truck fleet) . Throughput ranges from 1.33 trips (51% capacity) to around 2.6 trips (about 100 % capacity) for the periods of heavy backlog .

These three statistical measures are seen as providing useful measures of fleet performance which could aid its management . Allocation pressure is the major factor subject to operational control through management of fleet numbers and transport requirement .

5.2.4 Experiments on Allocation Pressure

5.2.4.1 Selection of data

Three distributions of the daily total trip times by individual trucks were compiled by pooling from the range of simulation data , output generated by the six fleet size experiments . Allocation pressures of 1.6, 2.5, and 3.6 trips per day per truck were selected . The daily total trip times of each truck in the fleet

for each of three days during which the fleet experienced the particular allocation pressure , were extracted from the simulation output and combined into a representative distribution of daily total trip times . Thus nine days of data resulted in the three distributions of Figure 5.4 with statistical summary in Table 5.3 . The particular allocation pressures were selected on the basis that about 2.5 trips per day per truck appeared to be the maximum steady state capacity (discussed below) of the simulated system , and convenient numbers of observations at 1.6 and 3.6 trips per day per truck were available providing reference points respectively about 36 % below and 44 % above this capacity .

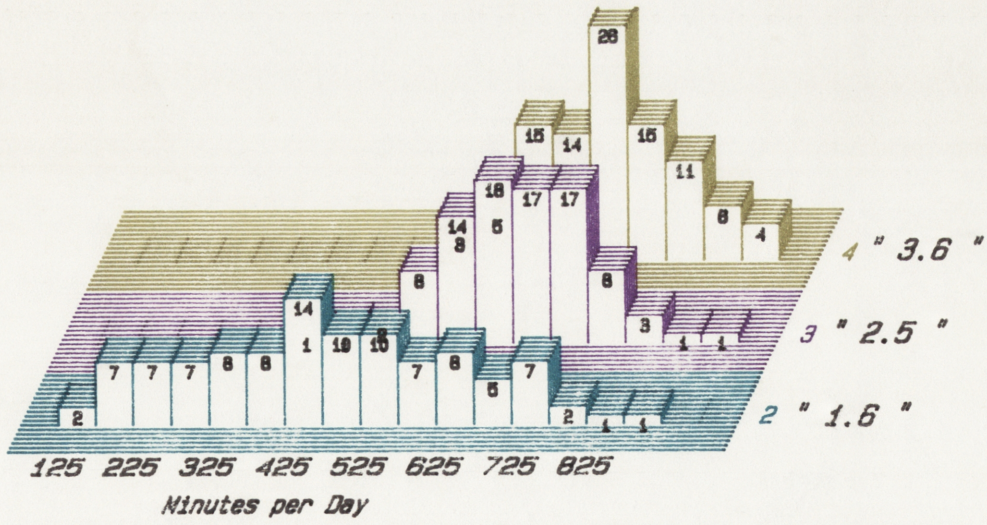


Figure 5.4 Frequency (%) Time Distributions of Daily Total Trip Time per Truck for Three levels of "Allocation Pressure"

TABLE 5.3

PERFORMANCE AT THREE LEVELS OF ALLOCATION PRESSURE

ALLOCATION PRESSURE (trips per day)	AVERAGE TOTAL WORKDAY (mins)	THROUGHPUT % capacity	BACKLOG % capacity	EXCEEDED MAXDAYLENGTH (% of total delivery)
1.6	450(175)*	63	0	8
2.5	658(103)	91	0.1	23
3.6	720(110)	100	40	50

*Standard deviations in parenthesis

The standard deviations of the distributions of daily total trip time per truck as percentages of the appropriate distribution means were 39%, 16% and 15% respectively . The effect of the heuristic in reducing variation in daily totals is apparent. Minimal backlogs were produced by 1.6 and 2.5 trip allocation pressures and throughput rose to 91% of theoretical capacity . 100% throughput was only obtained at the expense of very high levels of backlog . For this trip distribution and maximum daylength , sustained operation at allocation pressures in excess of 2.5 trips would likely produce steadily increasing backlogs indicating that the likely maximum level of throughput which can be sustained is about 91% . Further investigation of steady state behaviour is described below .

Another performance characteristic of concern is the number of trucks exceeding the desired maximum daylength . Even at the 2.5 trip level 23% of the trucks'

workdays exceeded 12 hours . Exceeding the expected work daylength occurs because the simulated despatcher is programmed to check the 'expected' trip time against time remaining ('MAXDAYLENGTH' set at 12 hours minus elapsed time) . Expected trip time was calculated as the recorded median round trip time for the compartment about to be allocated to the particular truck . The actual round trip time is determined by randomly selecting a trip time from the general distribution and adding breakdown and other elements of the trip model .

The data for the 3.6 allocation pressure provide an indication of the correct functioning of this component of the model since , under the conditions of considerable excess demand , the model was able to use 100 % of capacity and the numbers of trips exceeding the median round trip time was equal to the anticipated 50 % . The probabilities of exceeding the 12 hour limit calculated from the six fleet size experiments, that is , the 80 truck benchmark run and the 80,70, 65, 60 and 55 heuristically allocated runs , are given in Table 5.4 .

These simulations incorporate the dynamic variation in allocation pressure generated by the variable new daily demand and the previous days' backlog . The probability rises more steeply as the number of trips delivered per truck per day (throughput) increases . These proportions could be reduced by a modification of the heuristic setting a lower target maximum day length such as used in Pass 3 to ensure that only shorter daily work schedules were prepared .

TABLE 5.4

PROBABILITY OF EXCEEDING TARGET MAXIMUM WORKDAY

NUMBER OF TRUCKS IN FLEET	AVERAGE DELIVERY (*)	NUMBER OF TRIPS >MAXDAY	PROBABILITY %
80 Direct*	1.88	184	11
80	1.88	167	10
70	2.12	184	13
65	2.28	283	21
60	2.33	382	31
55	2.51	422	38

* Trips per truck per day
** Benchmark simulation run , direct allocation

5.2.5 Experiments on steady state capacity

Evidence obtained in the deterministic test (Chapter 2) suggested that with higher backlogs the heuristic algorithms worked more effectively to improve throughput. This aspect was further investigated by running a series of simulations with successively increased constant levels of daily total trip demand for the fleet . Trips at the rate of 120, 125, 130 , 135 and 140 trips per day were simulated in each trial for a fleet of 55 trucks . The overall sequence of trips synthesised from the study data was used but with the day of assignment ignored by simply taking the days alloca-

tion as the next recorded 120, 125 etc trips from the list . The results describing throughput and backlogs are given in Table 5.5 and presented in Figure 5.5 .

The simulation of 140 trips per day could not be completed because the backlog level continued to climb and ultimately exceeded the programmed capacity of the simulation model .

TABLE 5.5

STEADY STATE PERFORMANCE

FLEET SIZE (trucks)	AVERAGE DELIVERY THROUGHPUT (trips)	(% capacity)	AVERAGE BACKLOG (trips for fleet)	(% capacity)
120	2.17	83	6	4
125	2.26	86	8	6
130	2.35	90	15	10
135	2.41	92	43	30

Some backlog was evident , even at the 120 trip level and steady state capacity was exceeded at 140 trips . The level of backlog appeared to build up and stabilize at a higher level for each succeeding test up to 140 trips . The earlier observation of increasing

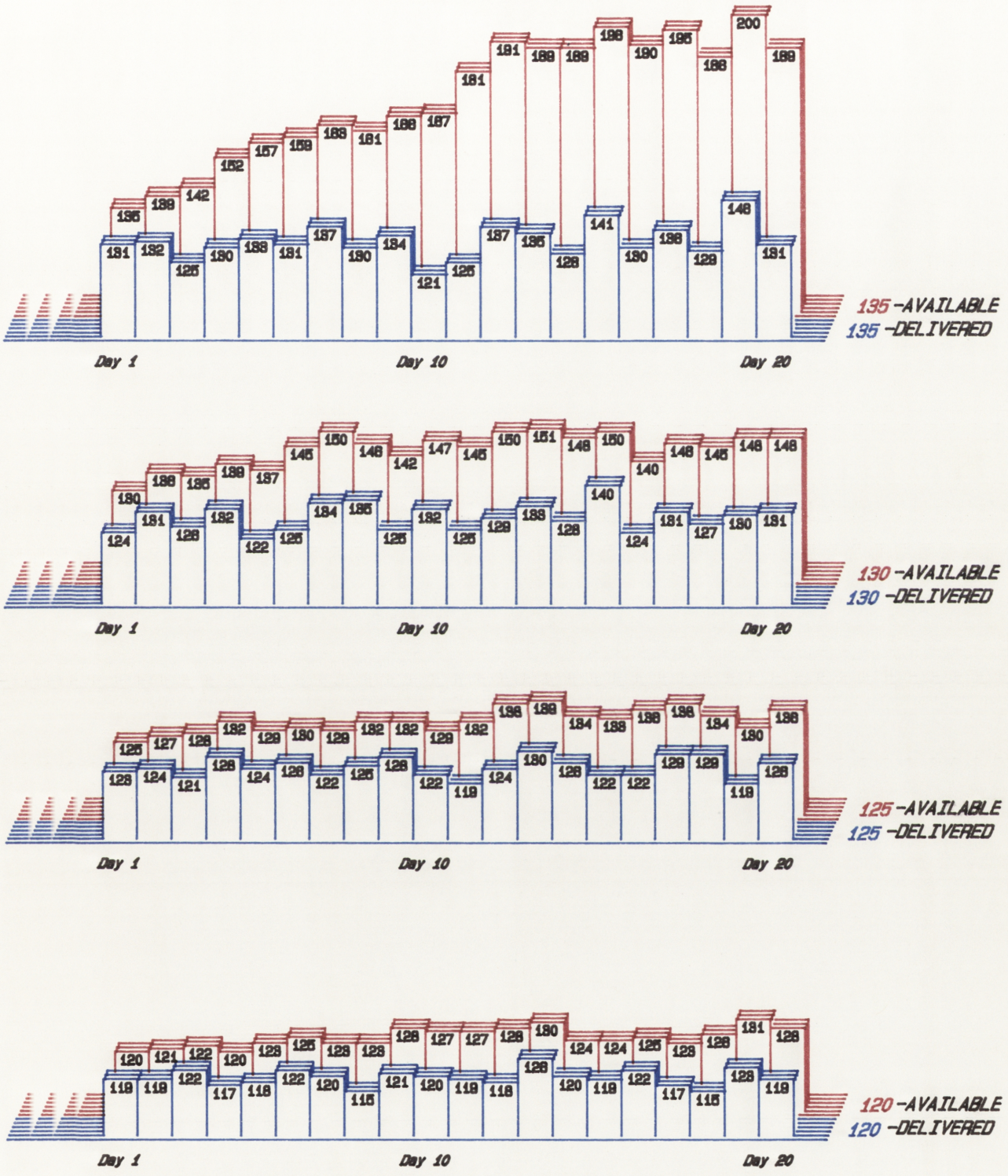


Figure 5.6 Trips Available and Delivered Daily
for Four Steady State Trip Demands

throughput at higher levels of backlog was confirmed . The maximum sustainable steady state level appeared to be between 135 and 140 trips per day for the simulated fleet . This level is about 2.5 trips per day , the apparent maximum sustained allocation pressure accepted in Section 5.3.4 , and about 93% theoretical maximum capacity . A knowledge of these levels for an operational fleet would be of great importance .

5.3 SUMMARY AND REVIEW

The linking of the heuristic allocation procedures, into the framework provided by the simulation model was readily achieved . A synthesised data set representing the 'full time ' component of the Eden truck fleet was derived and a simulation of the synthesised fleets without heuristic reallocation was run as a benchmark . Two series of test were then run to investigate the performance of the heuristic . A series of simulations of successively smaller fleet sizes demonstrated the response of the algorithm to 'allocation pressure ' . The importance for fleet size selection of a good knowledge of the long term patterns of demand and the acceptable levels of backlog was shown .

The likely optimal ranges for 'allocation pressure ' were illustrated by investigating distributions of daily total work times and those of steady state throughput . This provided a guide to the likely

numbers of trucks exceeding the desired workday and the likely levels of backlog . The experiments provide methods for obtaining information on 'allocation pressure' , 'throughput' and 'backlog' levels which in turn provides a basis for assessment of appropriate fleet sizes .

Successful application of the combined allocation / truck fleet simulation model offers a 'testbed' for the experimental investigation of the effects on fleet performance of

1. particular allocation policies defined by programmed algorithms and

2. levels and variation of the inputs to the model representing the particular performance of any particular fleet .

CHAPTER 6

REVIEW AND CONCLUSION

6.0 INTRODUCTION

The major objective of the study was to develop a computer based trip assignment system and evaluate its introduction in a centralized despatch system to replace organization of the haulage task by individual logging contractors using either their own trucks or owner drivers working as subcontractors. The development work was orientated toward possible application of centralized trip allocation for the log trucks serving the Eden chip mill .

6.1 INTRODUCTION OF CENTRALIZED DESPATCH

The principal constraint in the development of the proposed system was the need to maintain ownership and a large component of operational management of the trucks in the hands of contractors . Contractor controlled logging is almost universal in Australia since it reduces capital investment by the wood buyer and pro-

vides direct economic incentive to the immediate managers of the operational units , as contractors .

The principal difference between centralized despatch and current practice is the transfer of control of trip allocation now exercised by independent logging contractors , to the despatcher . Such a transfer would likely result in important changes in obligations and business risk as well as the hoped for improvement in operational efficiency .

Acceptance of such a change would depend on achievment of gains by the logging contractors and owner drivers and by the wood buyer to offset the costs associated with a centralized scheduling system . These gains could come from higher and more assured levels of truck utilization , a more predictable and uniform working day and a transport system capable of responding more flexibly to changed transport requirements .

Equitable distribution of the transport work was proposed as the most important criterion in such a complex system of many independent economic entities .

6.2 THE HEURISTIC TRIP ALLOCATION SYSTEM

A computer aided decision system , based on a heuristic allocation algorithm was accepted as the crucial technical innovation required for any proposed

system of centralized despatch . Development of such an algorithm has been achieved .

The allocation heuristic , based on 'best available trip allocation to worst off truck' and an associated points system was developed and tested for a simulated fleet of 20 trucks . The heuristic corrects the total time distribution for the fleet with each succeeding days allocation , the correction being towards equalizing the worktime of all trucks .

Results , for both a uniform distribution of trip times and one based on times observed at Eden , indicated considerable improvement in the equity of work allocation over monthly periods could be achieved by the application of the one pass heuristic as compared to random allocation (Test Series A and B , Chapter 2) .

The algorithm was extended by two additional passes based on reported approaches in the literature (Pass 2) and the need for maximum daylength constraint enforcement (Pass 3) . Subsequent detailed testing , (Series C and D , Chapter 2) , confirmed the capacity of the algorithm to reduce variation substantially , and provided indications of important behavioural characteristics such as improved throughput under conditions of 'overcapacity' and changes in the relative importance of the three passes under different conditions of load .

6.3 THE TRUCK FLEET SIMULATION MODEL

A truck fleet simulation model was required for testing the allocation procedures . Detailed simulation of such a complex socio-technical system as that encountered in the truck fleet , capable of accurately predicting human response , was accepted as difficult , if not impossible . Consequently , a design goal of a generalized and open model restricted primarily to technical system components , with explicit linkages between components , and which stressed the importance of user supplied control data , was accepted . The resulting model comprising about 1600 lines of SIMULA67 code has considerable flexibility for alternative application and expansion .

Performance data for the Eden truck fleet , used as a reference during model construction , provided the basic input for the model . Some arbitrary assumptions were required in areas where input data were inadequate. However , their impact was reduced because of the use of the model in a comparative way . A verification and validation check revealed consistent performance of the model , although there was some underestimation in the mill unloading times . The importance of this was again reduced since comparative testing was proposed . The model was accepted as a suitable 'testbed'.

6.4 TESTING OF THE HEURISTIC DESPATCH ALLOCATION SYSTEM IN A SIMULATED FLEET ENVIRONMENT

Considerable improvements in the inter-truck variations in monthly working times for the simulated fleet were obtained with the heuristic despatch allocation system . The performance of the system improved as greater load was placed on it . At the simulated 65 truck fleet level , 90% of the trucks fell within 5% of average monthly totals. Importantly , results similar to these were obtained across the range of fleet sizes from 55 to 70 trucks .

Allocation pressure , throughput and backlog were proposed as measures of system operation and performance. A maximum capacity of between 2.4 and 2.5 trips per day per truck was derived for the simulated fleet under conditions of steady demand . A maximum capacity of about 2.3 trips per day was indicated under more variable demand conditions . These represent throughputs of about 94% and 88% respectively of the theoretical capacity based on a 12 hour day .

Increased 'allocation pressure' resulted in improved throughput and lower variations between trucks in daily trip times , even at demand levels exceeding capacity . The improved throughput characteristic is important in promoting system stability under conditions of variable haulage demand . Benefits gained from low

variation in monthly, and to a lesser extent daily trip times, could be obtained in three ways.

1. By an 80 truck fleet working substantially the same average hours but with considerably more equality between the total times worked by each truck and a reduced probability of abnormally long hours on any day .

2. By a minimum sized fleet working at the level of sustained maximum throughput involving much longer hours but providing higher utilization . The maximum level could be chosen as the maximum probability of exceeding a 12 hour working day acceptable to the truck drivers .

3. A compromise between the two objectives , trading off the more stable daylength for the drivers offered by option one , with the greater utilization and longer day following fleet size reduction .

6.5 IMPROVED DATA FOR THE EDEN FLEET

Further testing and modification of both the heuristic algorithm and the simulation model to 'customise' it for application at Eden is possible .

The performance of the allocation system depends on the level of maximum daylength , allocation pressure and trip distribution . The choice of acceptable levels of maximum daylength are the prerogative of the truck drivers and owners . However , further testing of the allocation systems response to changed maximum daylengths most likely to be acceptable at Eden appears desirable .

Log truck fleets commonly experience fluctuations

in the daily transport demand . Before definitive recommendations could be made about the allocation systems suitability , experimentation on a wide range of representative monthly demand patterns for the Eden fleet would be desirable . A knowledge of the operational consequences of different backlog levels would also be required .

The simulation model developed for the testing of the allocation system provides a flexible framework simulating most truck fleet operations . Better operational data on truck speed , causes and duration of delays , and the times of terminal operations would improve its predictive validity . For example , field observations suggested considerable gains in system efficiency were possible if truck waiting time on landings could be reduced by the centrally controlled despatch of trucks to landings with wood available . The simulation model would provide a mechanism for subsequent evaluation of the efficiency consequences of such changes assuming reliable time study data on the causes , duration and frequency of landing delays .

A validated model for a particular fleet could also provide an experimental basis for the consideration of other questions such as performance gains attributable to higher truck power or increased loads .

6.6 DEVELOPMENT OF THE HEURISTIC ALGORITHM

The basic heuristic developed for the preparation of a prototype despatch table for a log truck fleet was implemented in a relatively simple form in order to simplify the evaluation of its operation. Modifications to improve the heuristic were considered at the time of its formulation and others became apparent during the experiments.

Three major developments seem feasible and all are based on attempting to maintain the optimal level of 'allocation pressure' .

With respect to the proportion of trucks exceeding the desired working day , daily readjustment of target daylength in the heuristic would achieve significant improvements in its performance at lower levels of 'allocation pressure' . Further experimentation would be required to determine the 'best' level for target daylength under different levels of demand .

The possibility of trucks having rostered days off became apparent during the development work of the study. A 'real' truck fleet would be planned to normally operate below maximum capacity to allow for some

buffer in the event of, for example, mishaps, breakdowns, and peak demands. Trucks could be deliberately removed from the system to increase the 'allocation pressure' on the algorithm toward maximum capacity . The limit to truck removal would be set by the estimated 90% maximum limit on throughput . The possibility of drivers , after notifying the despatcher , either to take days off at their discretion or use a day's credit to cover a day lost due to breakdown could be incorporated into the Pass 1 algorithm .

Thirdly , the allocation builder could also be modified to accommodate 'after hours' work. The points score system used to maintain equity between trucks could be modified to provide for allocation in accordance with desired levels of work specified by the truck owner . Individual points score targets could be preset in Pass 1 of the heuristic and the additional work resulting either from a 'spill' from a truck opting for a lower than average target or from 'excess demand' could be reallocated to trucks seeking more work.

These features could readily improve the acceptability of centralized despatch systems from the point of view of the truck owners and drivers . It is probable that many other such modificatons are possible with limitations on their application depending only on a capacity to express the desired performance change as a modification to the points allocation procedure .

6.7 CONTROL OF A CENTRALIZED DESPATCH SYSTEM

Control of a centralized despatch system for log trucks is likely to be exercised by a mill as the single wood buyer. However, another possibility would be a co-operative ownership of the system by the truck owners by arrangements similar to those used by taxi cooperatives . Another is the joint ownership of the facility by the truck owners and the mill .

There are considerable advantages , however , to the mill's logging management controlling the truck fleet since it increases their range of options to handle both short term and long term variations in the demand for wood. Such options could include the addition of one or two loaders under direct control of the despatcher and the purchase of a proportion of the logs from stockpiles at landings adjacent to major forest roads. The management of logs at stockpiles which could be both near to and far from the mill would enhance the opportunities for trip allocation and assist in operating the fleet at maximum capacity . The evaluation of location and use of such stockpiles is another potential future development in the use of the allocator and simulation model .

Centralized control and its associated detailed record system would provide the opportunity for a major revision of financial arrangements. In particular , the separation of payments for standing and travelling costs of the trucks could greatly enhance the level of finan-

cial equity for truck owners . Effectively , this would transfer much of the risk associated with reduced availability of wood or reduced demand for wood by the mill to the wood buyer since the fleet would receive standing time payments for periods when wood is not hauled. The mill would , of course , receive wood at considerably reduced transport cost in times of peak utilization . Such transfers in risk could be recognised in negotiating transport rates .

6.8 FUTURE RESEARCH

The area of computer aided planning and control of large scale forestry operations should receive greater attention to take advantage of recently available less costly computer equipment .

Research is needed into human performance in the use of computer aiding systems the better to identify strategies for improving the level of assistance rendered by the computer and to evaluate which areas of operational control can best benefit from such development .

The existing heuristic allocator could readily be developed to provide revision of the despatch allocation throughout the operating day . The simulation program could be extended , correspondingly , incorporating the extended heuristic to provide a simulated operational

environment for a human despatcher including real time decision support . Such systems could be used for both training and research . Much could be learned from these simulations , particularly about the levels of impact of a variety of 'user oriented' enhancements and the reponse of the system to them .

An improved flow of operational performance information can be anticipated following introduction of such computer aided decision systems , yielding new possibilities for improved planning and operational control . In particular , techniques of adaptive performance and transport requirement prediction could be developed based on the much richer data flow that would become available . These could provide for improved estimation of trip times to a particular landing based on a distribution of recent trip times

6.9 ACHIEVEMENTS

A goal of work equity was defined as one likely to be economically acceptable to all groups comprising large scale log transport systems .

A trip allocation method based on heuristic programming methods was developed and tested . Such an

algorithm could form the key component in an implementation of a computer aided despatch system .

A truck fleet computer simulation model sufficiently flexibile to readily incorporate alternative methods of trip allocation was programmed to evaluate the heuristic algorithm .

The heuristic algorithm demonstrated considerable capacity to achieve equitable work distribution .

Testing also provided indications of likely important system limitations , performance measures and determinants .

Together , these developments provide a feasible method for central despatch control , techniques and tools for evaluation of such developments and results achieved . These components would provide a baseline for the more specific work required for investigation or implementation of a specific transport control system .

BIBLIOGRAPHY

BARRY, D.L. and ROBINSON, D.F. (1977): ' A Heuristic for scheduling jobs on several identical machines ' , New Zealand Operational Research , Vol. 5(1) , pp45-53 .

BIRTWISTLE, G. (1981): 'A portable random number generator with built in well spread seeds', Proceedings of 1981 Winter Simulation Conference , IEEE , pp529-531.

BOWEN, H.C., FENTON, R.J., ROGERS, M.A.M., HURRION, R.D. and SECKER, R.L.J (1979): 'Interactive computng as an aid to decision makers' , in HADLEY G. (ed.) , OR 78 , North-Holland Publishing Company , Amsterdam .

BROWN, G.G. and GRAVES, G.W. (1981): 'Real time despatch of petroleum tank trucks' , Management Science, Vol. 27(1) , pp19-32 .

BUCK, J.R., DEISENROTH, M.P. and ALFORD, E.C. (1978): 'Man in the loop simulation' , Simulation , May 1978 , pp137-144 .

CEDER, A. and STERN, H.I. (1981): 'Deficit function bus scheduling with deadheading trip insertion for fleet size reduction' , Transportation Science , Vol. 16(4) , pp338-362 .

CONWAY, R.W. , MAXWELL, W.L. and MILLER , L.W (1967): Theory of Scheduling , Addison Wesley , Reading, Massachusetts .

CURRY, G.L. and SCHERMANN, A.L. (1968): 'A simulation solution of a transportation scheduling problem' , Journal of Industrial Engineering , Vol. 19 ,p 550 .

DAHL, O.J. and NYGAARD, K. (1966): 'SIMULA - An ALGOL based simulation language' , Communications of the ACM , Vol. 9 (9) , pp671-687 .

ELION, S. (1979): 'Production scheduling' ,in HADLEY G. (ed.) , OR 78 , North-Holland Publishing Company , Amsterdam .

FULLER, J.A. (1977): 'Optimal versus good solutions - An analysis of heuristic decision making' , Omega , Vol. 6 (6) , pp479-484 .

GAREY, M.R. , GRAHAM, R.L. and JOHNSON, D.S. (1978): 'Performance guarantees for scheduling algorithms' , Operations Research , Vol. 26(1) , pp3-21 .

GOLDEN, B.L. (1978): 'Closed form statistical estimates of optimal solution values to combinatorial problems' , Proceedings of the 1978 Winter Simulation Conference , IEEE , pp467-470 .

HELSGUAN, K. (1980): 'DISCO - A SIMULA based language for continuous , combined , and discrete simulation' , Simulation , July 1980 , pp1-12 .

HURRION, R.D. (1978) 'An investigation of visual interactive simulation methods using the job shop scheduling problem' , Journal of the Operations Research Society , Vol. 29 (11) , pp1085-1093 .

HURRION, R.D. and SECKER, R.J. (1978): 'Visual interactive simulation' , Omega , Vol. 6 (5) , pp419-426 .

JENSEN, P. and MYHRE, L. (1982): SIMULA Programmers Reference Manual , UNIVAC 1100 Series , Norwegian Computer Centre , Oslo .

JOHNSON, M.M. (1978): 'Sensible simulation for industrial systems' , Simulation , Vol. 30 (2) , p 55 .

KREUTZER, W. (1983): 'Of clocks and clouds , A guided walk through a simulator's tool box , New Zealand Operational Research , Vol. 11 (1) , pp51-120 .

LEEMING, A.M.C. (1981): 'A comparison of some discrete event simulation languages' , Simuletter , Vol. 12 (1-4) , pp9-16 .

LENSTRA, J.K. (1977): Sequencing by enumerative methods ,Mathematical Centre Tracts 69 , Mathematish Centrum , Amsterdam .

LUCAS, H.C. (1981): 'An experimental investigation of the use of computer- based graphics in decision making' , Management Science , Vol. 27 (7) , pp757-768.

MAISEL, H. and GNUGNOLI, G. (1972): Simulation of discrete stochastic systems , Science Research Associates INC , Palo Alto , United States of America .

MAISTER, D.H. (1980): Management of owner operator fleets , Lexington Books , United States of America .

MILLER-MERBACH, H. (1976): 'Heuristic procedures for solving combinatorial optimization problems in transportation' , Transportation Research , Vol. 8 , pp377-378 .

OSBORBE, M.R. (1977): 'Modelling using simulation languages' in OSBORNE, M.R. and WATTS, R.O. (eds.) Simulation and Modelling , University of Queensland Press , Brisbane .

PANWALKAR, S.S. and ISKANDER, W. (1977) 'A survey of scheduling rules' , Operations Research , Vol. 25(1), pp45-62 .

ROSS, G.J. , JONES, R.D. , KEMPTON, R.A., LAUCKNER, F.B., PAYNE, R.W., HAWKINS, D. and WHITE, R.P. (1978): MLP - Maximum Likelihood Program , Rothamsted Experimental Station , Harpenden , Hertsfordshire .

ROYAL STATISTICAL SOCIETY (1977) : Generalized Linear Interactive Modelling , London .

RYAN,T.A. , JOINER, B.L. and RYAN, B.F. (1976): MINITAB Student Handbook , Duxbury Press , Belmont , California .

SARGENT, R.C. (1979): 'Validation of simulation models' , Proceedings of 1979 Winter Simulation Conference , IEEE , pp497-503 .

SHANNON, R.E. (1981): 'Tests for the verification and validation of computer simulation models' , in OREN, T.I. , DELFOSSE, C.M. and SNUBS, C.M. (eds.) , Proceedings of the 1981 Winter Simulation Conference , IEEE , pp573-577 .

SMITH, H.T. and CRABTREE, R.G. (1975): Interactive planning : A study of computer aiding in the execution of a simulated scheduling task' , International Journal of Man Machine Studies , Vol. 7 , pp213-231 .

STANTON, R.B. (1977): 'Simulation and programming' in OSBORNE, M.R. and WATTS,R.O. (eds.) , Simulation and Modelling , University of Queensland Press , Brisbane .

STROUSTRUP, B. (1983): 'Adding classes to the C language : an exercise in language evolution' , Software - Practice and Experience , Vol. 13 , pp139-161 .

TAYLOR, R.N. (1975): 'Perception of problem constraints' , Management Science , Vol 22 (1) , pp22-29 .

WATTS , R.O. (1977): 'Modelling using simulation languages' in OSBORNE, M.R. and WATTS, R.O. (eds.) , Simulation and Modelling , University of Queensland Press , Brisbane .

WUNDERLICH, W.M. (1974): 'Integrated transport control of railways - Reasons for , results and aspects of future development on experience with 'Cybernetic Island , Hanover ' , in ACFET (ed.) , Traffic Control and Transportation Systems , North-Holland , Amsterdam .

ZEIGLER, B.P. (1976): Theory of modelling and simulation , John Wiley and Sons , New York .

HEURISTIC ALLOCATION AND TEST PROGRAMS

CLASS SETUP

function : generates the daily trip requirement list from the input file and sets the trips in a 'tasklist'

CLASS SCHEDULE

CLASS PASS 1

function : performs the first heuristic allocation based on an ordering of trip-sets

CLASS PASS 2

function : performs the second and third heuristic passes , the second is the 'tripswapper' , the third is the day-length checker and trip reallocater .

TRIP.4

function : main program for evaluation of 'uniform' trip distribution

CLASS SETUPB

function : modification of 'setup' above to handle EDEN trip distribution

TRIP.5

function : main program to handle EDEN trip distribution trials.

```
SIMULA-M  TRIP.SETUP
IMULA 3R8A 7AR1F2 12/27/82 13:30:55 (19)
      1     class   setup(comp,trips,u,u1,u2,trload,dim,returns,alpha);
      2                   integer array comp,trips,returns;
      3                   integer u,u1,u2,dim;
      4                      real alpha;
      5                 real trload;
      6       begin
      7
      8          procedure readin;
      9          begin
     10              integer ptr,numin,i;
     11              ref(infile) t;
     12              t:- new infile("tripmaster");
     13              t.open(blanks(20));
     14
     15              inspect t do
     16              begin
     17                     inimage;
     18                     numin:=ininт;
     19                     for ptr:=1 step 1 until numin do
     20                     begin
     21                            inimage;
     22                            comp(ptr,1):=t.image.sub(6,5).getint*alpha;
     23                            comp(ptr,2):=t.image.sub(11,6).getint;
     24                        sysout.outint(i,1);
     25                     end;
     26                     i:=1;
     27                     inimage;
     28                     while not endfile do
     29                     begin
     30                            ar(i):=ininт;
     31                            inimage;
     32                            i:=i+1;
     33                     end;
     34              end of insp;
     35                 t.close;
     36          end of readin;
     37
     38             procedure gettrips;
     39             begin
     40                 integer i,top,low,cpt,trk,todaystrips,base;
     41                    low:=dim * 1;
     42                    for i := 1 step 1 until len do
     43                       list(i):= 0;
     44                       if stochastic then
     45                       begin
     46                              todaystrips:=normal(trload,b,u1);
     47                              while todaystrips < low or todaystrips > len do
     48                                 todaystrips := normal(trload,b,u1);
     49
     50                           if DIAGSETUP  then
     51                              begin
     52                                 outtext("todaystrips ");outint(todaystrips,5);
     53                                 outimage;
     54                              end;
     55
     56                           for i := 1 step 1 until dim do
     57                              list(i) := histd(ar,u);
     58                        for i := 1 step 1 until dim * 3 do
```

```
     59                              prelist(i):=0;
     60                           base:=1;
     61                           for i := low step 1 until todaystrips do
     62                              begin
     63                                 prelist(base) := histd(ar,u);
     64                                 base:= base + 1;
     65                              end;
     66                           i := 1;
     67                           while returns(i) <> 0  and base < 100 do
     68                              begin
     69                                 prelist(base) := returns(i);
     70                                 base:= base + 1;
     71                                 i:=i + 1;
     72                              end;
     73                           if base > dim * 3 then
     74                              begin
     75                                 outtext(" trip overrun ");
     76                                 outint(base-todaystrips,4);outtext(" *********");
     77                                 base:=dim * 3 ;
     78                              end;
     79                              outint(base,4);
     80                              top :=  base-dim-1;
     81                           base := 1;
     82                           low := dim * 2;
     83                           for i := dim step 1 until low do
     84                              begin
     85                                 list(i) := prelist(base);
     86                                 base:= base + 1;
     87                              end;
     88                           for i :=  1 step 1 until top do
     89                              begin
     90                                 trk := randint(low,len,u2);
     91                                 while list(trk) <> 0  do
     92                                    trk := randint(low,len,u2);
     93                                 list(trk) := prelist(base);
     94                                 base:=base + 1;
     95                              end;
     96                        end
     97                        else
     98                           for i:= 1 step 1 until len do
     99                           begin
    100                                 list(i) := histd(ar,u);
    101                           end;
    102             END of gettrips;
    103
    104          procedure maketasklists(trnum);
    105                integer trnum;
    106             begin
    107                 integer i,j;
    108                 for i:=1 step 1 until trnum do
    109                    begin
    110                       for j:=1 step 1 until 7 do trips(i,j):=0;
    111                       for j:= 1 step 1 until 4 do
    112                       trips(i,j):=list( i+(j-1)*20);
    113                          trips(i,8):=i;
    114                    end;
    115             END of make;
    116
    117             procedure printer;
    118             begin
```

```
                    integer i,j;
                for i := 1 step 1 until dim do
                    begin
                        for j := 1 step 1 until 8 do
                            outint(trips(i,j),8);
                        outimage;
                    end;

            END of printer;


            Procedure newday(num);
                        integer num;
            begin
                    gettrips;
                    maketasklists(num);
                    allocate;
                       if diagsetup then printer;
            end;

            Procedure allocate;
            begin
                    integer i,j;
                    for i :=1 step  1 until  dim do
                       begin
                            for j := 1 step 1 until 4 do
                                if trips(i,j) <> 0 then
                                begin
                                    trips(i,5):=trips(i,5) + comp(trips(i,j),1);
                                    trips(i,6):=trips(i,6) + comp(trips(i,j),2);
                                end;
                            trips(i,7):= trips(i,6)+ trips(i,5) *2;
                       end;
               END of allocate;

          integer len;
          integer array list(1:dim*4),prelist(1:dim*4);
          real array ar(1:86);
             real b;
             boolean DIAGSETUP,STOCHASTIC;
             STOCHASTIC:= true;
             len := dim * 4;
             b:=trload/10;
             outtext("alpha");outfix(alpha,3,7);outimage;
          readin;
      end of setup;
```

..NOTE 123 LINE 58: UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE

ND SIMULA 3R8A. 164 LINES, NO ERRORS.

```
SIMULA:M   TRIP.SCHEDULE
IMULA 3R8A 74R1F2 12727782 13:31:23 (44)
      1       class schedule(trips,history,comps,numtr,returns);
      2                integer array comps,trips,history,returns;
      3                integer numtr;
      4       BEGIN
      5
      6
      7
      8           SIMSET class  pass1;
      9            begin
     10
     11                HEAD class tasklists; ;
     12
     13                LINK class taskcell(ar); real array ar;
     14                BEGIN
     15                       real array trips(1:7);
     16                       for i := 1 step 1 until 7 do
     17                          trips(i):= ar(i);
     18                END of taskcell;
     19
     20                Procedure pushtasks;
     21                BEGIN
     22                       integer k,d,l;
     23                       real array ar(1:7);
     24
     25                       for k := 1 step 1 until numtr do
     26                       begin
     27                             if (abs(todays(k,4)-ttmu)*2)/ttmu > abs(todays(k,5)-tkmu)
     28                                /tkmu then d:=1 else d:=2;
     29                                moretasks(d):= true;
     30
     31                             for l := 1 step 1 until 7 do
     32                                ar(l):= todays(k,l);
     33                             new taskcell(ar).into(tasker(d));
     34                       end;
     35                END of pushtasks;
     36
     37                Procedure printclass;
     38                begin
     39                       ref(tasklists) t;
     40                       ref(taskcell)  c;
     41                       integer i,j,k;
     42                       for i := 1 step 1 until 2 do
     43                       begin
     44                             t:- tasker(i);
     45                             c :- t.first;
     46                             for j := 1 step 1 until t.cardinal do
     47                             begin
     48                                    for k := 1 step 1 until 5 do
     49                                       outfix(c.trips(k),2,7);
     50                                    outimage;
     51                                    c:-c.suc;
     52                             end;
     53                       end;
     54                END;
     55
     56
     57                Procedure getmeans;
     58                BEGIN
     59                       integer i;
     60                       real array total(1:4);
     61                       for i := 1 step 1 until 4 do
     62                          total(i):= 0;
     63                       for i := 1 step 1 until numtr do
     64                          begin
     65                             total(1):= total (1) + todays(i,4);
```

```
                        total(2):= total (2) + todays(i,5);
                        total(3) :=total(3) + past(i,1);
                        total(4) :=total(4) + past(i,2);
                    end;

                ttmu:=total(1)/numtr;
                tkmu:=total(2)/numtr;
                ptmu:=total(3)/numtr;
                pkmu:=total(4)/numtr;

            if DIAGP1 then
                begin
                    outtext("totals");
                    for i := 1 step 1 until 4 do
                        outfix(total(i),2,10);
                    outimage;
                    outfix(tkmu,2,7);outtext("was tkmu");outimage;
                end;

        END of getmeans;
        Procedure getdirection(point1); integer point1;
        begin

                if(past(point1,2) - pkmu) > (past(i,1) - ptmu)
                    then
                        direction := 1
                    else
                        direction := 2;
                if not moretasks(direction)
                    then
                    begin
                        if direction = 1 then
                            direction := 2 else direction := 1;
                    end;
          end of getdir;

        Procedure dealouttrips;
        begin
                integer i,k;
                ref(tasklists) t;
                ref(taskcell) c;


                    t:-tasker(direction);
                    c:-t.first;
                    c.out;
                    for k := 1 step 1 until 7 do
                        tasks(k) := c.trips(k);
                if t.cardinal = 0 then moretasks(direction) := false;
        END of dealout;

        procedure printer;
        begin
            integer i;
            for i := 1 step 1 until numtr do
            begin
                    for j := 1 step 1 until 7 do
                        outint(todays(i,j), 7);
                    for j := 1 step 1 until 4 do
                        outint(past(i,j),8);



                    for j := 1 step 1 until 3 do
                            outint(history(i,j),6);
                    outimage;
            end;
        END;
        integer i,j,ttmu,tkmu,ptmu,pkmu,t,k,direction,point1,point2,tr;
        integer array todays(1:numtr,1:7),past(1:numtr,1:4),
                     tasks(1:7),first(1:numtr,1:4);
        boolean array  moretasks(1:2);
        boolean DIAGP1;
        ref(tasklists)array tasker(1:2);

        tasker(1) :- new tasklists;
        tasker(2) :- new tasklists;

        for i := 1 step 1 until numtr do
            begin
                first(i,1):= trips(i,1);
                first(i,2):=comps(trips(i,1),1);
                first(i,3):=comps(trips(i,1),2);
                first(i,4):= first(i,2)+2 + first(i,3);
                past(i,1):= history(i,1) + first(i,2);
                past(i,2):= history(i,2) + first(i,3);
                past(i,3):= history(i,3) + first(i,4);
                past(i,4):=i;
                for j := 2 step 1 until 4 do
                    begin
                        tr:=trips(i,j);
                        if tr > 0  and tr <= 82  then begin
                                todays(i,j-1):= tr;
                                todays(i,4):=todays(i,4) + comps(trips(i,j),1);
                                todays(i,5):=todays(i,5) + comps(trips(i,j),2);
                            end
                    else if tr <> 0 then
                        begin
                            outtext("rogue comp"); outint(tr,3); outimage;
                        end;
                    end;
                todays(i,6):= todays(i,4) + 2 + todays(i,5);
        END of instal loop;

        sort(todays,numtr,6,6);
        sort(past,numtr,4,3);
        getmeans;
        pushtasks;
        if DIAGP1 then  printer;
        for i := 1 step 1 until numtr do
        begin
                point1:= numtr+ 1 - i;
                point2:=past(point1,4);
                getdirection(point1);
                dealouttrips;
                for j :=1 step 1 until 3 do
                    trips(point2,j+1):=tasks(j);
                for j := 1 step 1 until 3 do
                    trips(point2,j+4):=past(point1,j) + tasks(j + 3);
        end;
        sort(trips,numtr,8,7);
    END of pass1;
```

```
186          simset class pass2;
187          begin
188                  head class alltrucks;;
189                     head class mytrips;;
190                     link class trip;
191                     begin
192                         integer id,time,dist,score;
193                     end of trips;
194
195                  link class trday(i); integer i;
196                  begin
197                      procedure maketrips;
198                      begin
199                             ref(trip) t;
200                             integer j,k;
201                             for j := 1 step 1 until 4 do
202                                if trips(i,j) <> 0 then
203                                   begin
204                                       t:- new trip;
205                                       t.id:=trips(i,j);
206                                       t.time:=comps(trips(i,j),1);
207                                       t.dist:=comps(trips(i,j),2);
208                                       t.score:=t.time * 2 + t.dist;
209                                       todaytime:=todaytime + t.time;
210                                       t.into(mytr);
211                                   end;
212                      end of maketrips;
213
214                      integer cpt,myscore,myt,myk,timeleft,todaytime,myident;
215                      ref(mytrips) mytr;
216                      mytr:- new mytrips;
217                      maketrips;
218                      myt:=trips(i,5);
219                      myk:=trips(i,6);
220                      myscore:=trips(i,7);
221                      myident:=trips(i,8);
222                      timeleft:=720 - todaytime;
223                      if diagprint then begin
224                          outtext("timeleft was");outint(timeleft,5);outtext("for");
225                          outint(myident,5);outimage;end;
226                  END of trday;
227
228                  Procedure instaltrips;
229                  begin
230                         integer i;
231                         for i := 1 step 1 until numtr do
232                            begin
233                                tdy:- new trday(i);
234                                tdy.into(all);
235
236                                  if DIAGPRINT then
237                                      begin
238                                          outtext("trday.myscore"); outint(tdy.myscore,5);
239                                          outimage;
240                                      end;
241                            end;
242                     END of instal;
243
244                Procedure instaltr(td,tr); ref(trday) td;ref(trip) tr;
245                begin
246                         td.myt:=td.myt+tr.time;
247                         td.myk:=td.myk +tr.dist;
248                         td.myscore:=td.myscore + tr.time * 2 + tr.dist;
249                         td.todaytime:=td.todaytime + tr.time;
250                           if diagprint then begin   !**************************;
251                             outtext(" trip");outint(tr.id,4);outtext("in");
252                             outtext(" truck");outint(td.myident,4);outimage;
253                         end;
254                END;
255
256                Procedure removetr(td,tr); ref(trday) td;ref(trip) tr;
257                begin
258                         td.myt:=td.myt-tr.time;
259                         td.myk:=td.myk -tr.dist;
260                         td.myscore:=td.myscore - tr.time * 2 - tr.dist;
261                         td.todaytime:=td.todaytime - tr.time;
262                           if diagprint then  begin !*********************;
263                             outtext(" trip");outint(tr.id,4);outtext("out");
264                             outtext(" truck");outint(td.myident,4);outimage;
265                         end;
266                END;
267
268                Procedure sorttimes;
269                begin
270                         ref(trday) t,temp,base;
271                         base:-all.first;
272                         while base =/= NONE do
273                            begin
274                                 temp :- t:- base;
275                                 while temp =/= all.last do
276                                    begin
277                                        temp:-temp.suc;
278                                        if temp.todaytime < t.todaytime then t:- temp;
279                                    end;
280                                 if t =/= base then t.precede(base)
281                                 else base :- base.suc;
282                                 if diagprint then begin
283                                    outtext("in sort");outint(t.myident,5);outimage;
284                                 end;
285                            end;
286                 END of sorttimes;
287
288                    procedure allwrite;
289                    begin
290                           ref(trday) td;
291                           ref(trip)tr;
292                           td:- all.first;
293                           while td =/= all.last do
294                           begin
295                               outtext(" trday ident in all was");outint(td.myident,3);
296                               tr:-td.mytr.first;
297                               while tr =/= none do
298                                  begin
299                                      outint(tr.id,4);
300                                      tr:-tr.suc;
301                                  end;
302                               outtext("todaytime");outint(td.todaytime,5);
303                               outimage;
304                               td:- td.suc;
305                           end;
```

```
306                 END of alwrite;

308             Procedure getsmallest(trk,trp);ref(trday) trk;ref(trip) trp;
309                     name trk,trp;
310             begin
311                     integer smallest;
312                     ref(trip) temp;
313                     temp:-trk.mytr.first;
314                     smallest:=temp.time;
315                     trp:-temp;
316                     WHILE temp =/= trk.mytr.last DO
317                        begin
318                             temp :-temp.suc;
319                             if temp.time < smallest and temp.id <> 0
320                             then
321                                 begin
322                                     trp:-temp;
323                                     smallest:=trp.time;
324                                 end;
325                     end;
326                         if DIAGprint then
327                            begin
328                                outtext("small was");outint(trp.id,3);outint(trp.time,3);
329                                outtext("for");outint(trk.myident,4);outimage;
330                                outimage;
331                            end;
332             END of smallest;
333
334             PROCEDURE getbiggest(trk,trp);ref(trday) trk;ref(trip)trp;
335                     name trk,trp;
336             begin
337                     integer biggest;
338                     ref(trip) temp;
339
340                     temp:-trk.mytr.first;
341                     trp:-temp;
342                     biggest:=trp.time;
343
344                     WHILE temp =/= trk.mytr.last
345                     DO
346                        begin
347                             temp:-temp.suc;
348                             if temp.time > biggest
349                                then
350                                begin
351                                     trp:-temp;
352                                     biggest:=trp.time;
353                                end;
354                        end;
355                        if DIAGprint then
356                           begin
357                                outtext("biggest was");outint(trp.id,3);outint(trp.time,3)
358                                outtext("for");outint(trk.myident,4);outimage;
359                                outimage;
360                           end;
361                END of biggest;
362
363             PROCEDURE tryswaps;
364             begin
365
366                     procedure swapit(t1,t2,tp1,tp2);ref(trday) t1,t2;
367                                                     ref(trip)tp1,tp2;
```

```
368                 begin
369                     ref(trip) temp;
370
371                     tp2.follow(tp1);
372                     installr(t1,tp2);
373                     removetr(t2,tp2);
374                     tp1.into(t2.mytr);
375                     installr(t2,tp1);
376                     removetr(t1,tp1);
377                 end of swapit;
378
379                 integer i;
380                 ref(trday) t1,t2;
381                 ref(trip) tp1,tp2;
382
383                 t1:- all.first;
384                 t2:-all.last;
385
386                 for i := 1 step 1 until 5 do
387                    begin
388                        getsmallest(t1,tp1);
389                        getbiggest(t2,tp2);
390                        if DIAGprint then
391                           begin
392                               outtext("biggest was");outint(tp1.id,3);outint(tp1.time
393                               outtext("for");outint(t1.myident,4);outimage;
394                               outimage;
395                           end;
396                        if tp1.time < tp2.time then
397                           swapit(t1,t2,tp1,tp2);
398                        t1:-t1.suc;
399                        t2:-t2.pred;
400                    end;
401             END of try;
402
```

```
404             Procedure Killovertime;
405             begin
406
407                     procedure putback(tr); ref(trip) tr;
408                     begin
409                           if diagprint then  begin !************************;
410                               outtext("LOST one in killovertime");
411                               outint(tr.id,4);outtext(" time");outint(tr.time,5);outimag
412                            end;
413                            rptr:=rptr + 1;
414                            returns(rptr):=tr.id;
415                     end;
416
417                     ref(trday) base,td1,td2;
418                     ref(trip) tr,temp;
419                     integer overtime;
420
421                     base:- all.last;
422                     while base.todaytime > 720 and base =/= all.first do
423                        base:-base.pred;
424
425                     td1 :- all.last;
426                                    begin
427                     end;
428                       overtime :=  td1.todaytime- 720;
429                     while overtime > 0 do
430                        begin
431                            if diagprint then  begin !************************;
432                              outtext("overtime");outint(overtime,5);outint(td1.myident,4);
433                               outimage;
434                            end;!************************;
435                            tr:- td1.mytr.first.suc;
436                            temp :- tr.suc;
437                            while temp =/= none do
438                               begin
439                                   if overtime-temp.time < 0 and temp.time < tr.time
440                                   then
441                                      tr:-temp;
442                                   temp:-temp.suc;
443                               end;
444                            tr.out;
445                            removetr(td1,tr);
446
447                               td2:-all.first;
448                               while td1.todaytime > td2.todaytime and td2 =/= all.last do
449                                   td2:-td2.suc;
450                               if td2 =/= td1 then  td1.precede(td2);
451                      if diagprint then  begin !************************;
452                            outtext("td1");outint(td1.myident,3);outtext("pred");
453                            outint(td2.myident,3);outtext("td1.todaytime");
454                            outint(td1.todaytime,5);outtext("td2.todaytime");
455                            outint(td2.todaytime,5);outimage;
456                        end ; !************************;
457                        td2:-all.first;
458                        if diagprint then
459                            begin
460                            outtext("td2 todaytime");outint(td2.todaytime,5);outimage;
461                        end;
462                        if tr.time + td2.todaytime > 720
463                           then putback(tr)
```

```
464                         else
465                             begin
466                                 insaltr(td2,tr);
467                                 td1:-td2.suc;
468                                 while td2.todaytime > td1.todaytime
469                                       and td1 =/= all.last do
470                                       td1:-td1.suc;
471                                 td2.precede(td1);
472                     if diagprint then  begin !**********************;
473                             outtext("td2");outint(td2.myident,3);outtext("pred");
474                             outint(td1.myident,3);outtext("td2.todaytime");
475                             outint(td2.todaytime,5);outtext("td1.todaytime");
476                             outint(td1.todaytime,5);outimage;
477                         end ; !**************************;
478                             end;
479                      td1:-all.last;
480                         overtime:=td1.todaytime-720;
481                 end;
482             END of proc overtime;
483
484                procedure printer;
485                begin
486                    integer i,j;
487                    for i := 1 step 1 until 20 do
488                       begin
489                           for j := 1 step 1 until 8 do
490                           outint(trips(i,j),7);
491                           outimage;
492                       end;
493                end;
494
495                procedure getmeanscore;
496                begin
497                       integer i,total;
498                       for i := 1 step 1 until numtr do
499                          total := total + trips(i,7);
500                       meanscore:=total / numtr;
501                end;
502
503                Procedure rewrite;
504                begin
505                       integer ident,i;
506                       ref(trip)  t;
507                       tdy :- all.first;
508                       while tdy =/= none do
509                       begin
510                              inspect tdy do
511                              begin
512                                    ident:=myident;
513                                    t:-mytr.first;
514                                    for i := 1 step 1 until 4 do trips(ident,i):= 0;
515                                    i:=1;
516                                    while t =/= none do
517                                    begin
518                                          trips(ident,i) := t.id;
519                                          t:-t.suc;
520                                          i :=i + 1;
521                                    end;
522                                    trips(ident,5):=myt;
523                                    trips(ident,6):=myk;
524                                    trips(ident,7):=myscore;
525                                    trips(ident,8):=myident;
526                              end of inspect;
527                              tdy :- tdy.suc;
528                       end;
529                END of proc rewrite;
530
531                integer i,meanscore,rptr;
532                boolean DIAGPRINT;
533                ref(alltrucks) all;
534                ref(trday) tdy;
535                all:- new alltrucks;
536                if diagprint then printer;
537                installtrips;
538                getmeanscore;
539                if diagprint then
540                   begin
541                      outtext(" MEANSCORE"); outint(meanscore,5);outimage;
542                   end;
543                tryswaps;
544                if diagprint then begin rewrite;printer;end;
545                sorttimes;
546                if diagprint then  allwrite; !**************************;
547                killovertime;
548                if diagprint then  allwrite; !**************************;
549                 rewrite;
550             END of pass2;
551
552                procedure sort(input,len,wid,key);
553                          integer array input;
554                          integer len,wid,key;
555
556                begin
557                       procedure test(i,inc); integer i,inc;
558                       begin
559                              integer l,temp;
560                              real last,thisone;
561                              l:=i-inc;
562                              if l >= 1 then
563                                 begin
564                                      last:=base(l,1);
565                                      thisone:=base(i,1);
566                                      if last > thisone then
567                                         begin
568                                            base(l,1) :=thisone;
569                                            base(i,1) := last;
570                                            temp := base(l,2);
571                                            base(l,2):=base(i,2);
572                                            base(i,2):=temp;
573                                            test(l,inc);
574                                         end;
575                                 end;
576                          end of test;
577
578                    integer i,j,k,l,max,inc;
579                    real array base(1:200,1:2),temp(1:200,1:8);
580
581                    max:=len;
582                    i:=1;
583                    for i:=1 step 1 until max do
```

```
                    begin
                        base(i,1):=input(i,key);
                        base(i,2):=i;
                    end;

                    for inc:=entier(max/4) step -1 until 1 do
                       for j:=1 step 1 until inc do
                          begin
                             i:=j + inc;
                             while i <= max do
                                begin
                                   test(i,inc);
                                   i:=i + inc;
                                end;
                          end of sort phase;

                    for i:=1 step 1 until max do
                       for j:=1 step 1 until wid do
                          temp(i,j):=input(base(i,2),j);

                    for i:=1 step 1 until max do
                       for j:=1 step 1 until wid do
                          input(i,j):=temp(i,j);
              end of proc sort;

              ref(pass2) p2;
              ref(pass1) p1;
              p1:-new pass1;
              p2:- new pass2;
      END of schedule;

...NOTE 132 LINE 45:   IMPLICIT ¤QUA¤, RUN TIME CHECK ON TASKCELL
...NOTE 123 LINE 46:   UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE
...NOTE 132 LINE 51:   IMPLICIT ¤QUA¤, RUN TIME CHECK ON TASKCELL
...NOTE 132 LINE 110:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TASKCELL
...NOTE 132 LINE 271:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 277:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 281:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 292:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 296:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 300:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 304:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 313:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 313:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 316:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 318:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 323:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 328:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 328:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 329:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 340:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 340:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 342:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 344:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 347:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 352:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 357:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 357:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 358:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 383:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY

...NOTE 132 LINE 384:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 398:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 399:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 421:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 423:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 425:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 435:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 436:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 442:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 447:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 449:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 457:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 467:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 470:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 479:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 507:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 132 LINE 513:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 519:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
...NOTE 132 LINE 527:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
...NOTE 164 LINE 602:  SUBSCRIPT EXPRESSION IS NOT INTEGER
...NOTE 183 LINE 315:  CALL BY NAME GENERATES RUNTIME CHECK ON ACTUAL QUALIFICATION
...NOTE 183 LINE 322:  CALL BY NAME GENERATES RUNTIME CHECK ON ACTUAL QUALIFICATION
...NOTE 183 LINE 341:  CALL BY NAME GENERATES RUNTIME CHECK ON ACTUAL QUALIFICATION
...NOTE 183 LINE 351:  CALL BY NAME GENERATES RUNTIME CHECK ON ACTUAL QUALIFICATION

END SIMULA 3R8A. 613 LINES, NO ERRORS.


@SIMULA,M   TRIP.4
SIMULA 3R8A 74R1F2 12/27/32 13:31:45 (39)
        1   BEGIN
        2         INTEGER DIM,LEN,U,u1,u2,rEPS,MAXLEN,i,j;
        3         REAL trload,alpha;
        4         DIM:=20;
        5         REPS:=5;
        6         LEN:=DIM * 4;
        7         MAXLEN:=DIM;
        8         alpha:=0.25;
        9            U:=entier(timeused*2) + 1;
       10            u1:= u + 4;
       11            u2:= u1 + 2 ;
       12            outtext("seeds");outint(u,15);outint(u1,15);
       13            outint(u2,15);outimage;
       14            OUTIMAGE;
       15            for  j :=i step 1  until 3 do
       16            begin
       17               alpha:=alpha + 0.25;
       18               for trload := 42 step 3 until 54 do
       19             BEGIN
       20                   external class setup;
       21                   external class schedule;
```

```
      INTEGER I,R,J,K,COUNT,CYCLE,IDENT,day,datatype;
      REAL MEAN,TOTSQ,STD,TOTAL,BASE,INTERVAL;
      REAL ARRAY MONTH(1:DIM,1:23,1:3),tempdat(1:maxlen),
           FRED(1: MAXLEN,1:3),BILL(1:MAXLEN,1:3),
                ar(1:36);
      INTEGER ARRAY trips(1:dim,1:8),comp(1:100,1:2),
                    returns(1:100),size(1:3),last(1:dim,1:3);
         BOOLEAN diagmain;
      ref(setup) set;
      ref(schedule) s1;

         outtext("trload");outint(trload,10);outimage;

      FOR CYCLE:=1 STEP 1 UNTIL REPS DO
         BEGIN
          for i := 1 step 1 until 3 do
                   SIZE(i):= 1;

             u:= u + 2;
             u1:=u1 + 2;
             u2:=u2 + 2;


        set:-new setup(comp,trips,u,u1,u2,trload,dim,returns,alpha);



          FOR i :=1 step 1 until 20 do
             FOR j :=21 step 1 until 23 do
                FOR k := 1 step 1 until 3 do
                   month(i,j,k) := 0;
          FOR i := 1 step 1 until 20 do
             for j := 1 step 1 until 3 do
                last(i,j) := 0;

          FOR day:=1 STEP 1 UNTIL 20 DO
             BEGIN
                SET.newday(dim);
                 for r := 1 step 1 until 40 do returns(r):=0;
                s1:-new schedule(trips,last,comp,dim,returns);
               if diagmain then  printer;
                gettodays; if diagmain then printer;
                DAYSTRIPS(day);
             END;

          for datatype:=1 step 1 until 3 do
          begin
              FOR IDENT:=1 STEP 1 UNTIL DIM  DO
              BEGIN
                      TOTAL:= 0;
                      TOTSQ:=0;
                      FOR day:= 1 STEP 1 UNTIL 20 DO
                         TOTAL:=TOTAL + MONTH(IDENT,day,datatype);
                             month(ident,21,datatype):=total;
                      MEAN:=TOTAL/20;
                             month(ident,22,datatype):=mean;
                      FRED(SIZE(datatype),datatype ):=TOTAL;
                      FOR day := 1 STEP 1 UNTIL 20 DO
                         TOTSQ:=TOTSQ + (MONTH(IDENT,day,datatype)-MEAN)**2;




                      STD:=SQRT(TOTSQ/19);
                      month(ident,23,datatype):=std;
                      BILL(SIZE(datatype),datatype):=STD;
                      SIZE(datatype):=SIZE(datatype)+1;
                END;

             END of datatype loop;


           for datatype := 1 step 1 until 3 do
              begin
                          outtext("DATATYPE "); outint(datatype,3);
                          outimage;
               transcribe(fred,tempdat,datatype,maxlen);
               VARIATION(tempdat,MAXLEN,MEAN,STD);
                     OUTTEXT("    MEAN TOTAL TRIP TIME");OUTREAL(MEAN,5,10);
                     OUTTEXT("    STD -TOTAL TRIP TIME");OUTREAL(STD,5,10);
                     OUTIMAGE;
                  BASE:=MEAN-STD*4.25;
                  INTERVAL:=STD/2;
                     transcribe(bill,tempdat,datatype,maxlen);
               VARIATION(tempdat,MAXLEN,MEAN,STD);
                     OUTTEXT("    MEAN DEVIATION     ");OUTREAL(MEAN,5,10);
                     OUTTEXT("    STD -DEVS  TRIP TIME");OUTREAL(STD,5,10);
                     OUTIMAGE;
                  BASE:=MEAN-STD*4.25;
                  INTERVAL:=STD/2;
               OUTIMAGE;
              end;
          end of cycle;
         END;
      END;
   END

END SIMULA 3R8A. 182 LINES, NO ERRORS.


&SIMULA,M  TRIP.SETUPB
SIMULA 3R8A 74R1F2 12/27/82 13:32:10 (7)
      class setupb(comp,trips,u,u1,u2,u4,trload,dim,returns,centre,spread);
                integer array comp,trips,returns;
                integer u,u1,u2,u4,dim,centre,spread;
             real trload;
    begin

       procedure readin;
       begin
           integer ptr,numin,i;
           numin:=80;
           for i := 1 step 1 until numin do
              begin
                   comp(i,1):= 50 +(i*4);
                   comp(i,2):= i*4;
              end;
       end of readin;

          procedure gettrips;
          begin
              integer i,top,low,cpt,trk,todaystrips,base,lowa,high;
```

```
                low:=dim + 1;
                for i := 1 step 1 until len do
                   list(i):= 0;
                   if stochastic then
                   begin
                         todaystrips:=normal(trload,b,u1);
                         while todaystrips < low or todaystrips > len do
                            todaystrips := normal(trload,b,u1);

                      if DIAGSETUP  then
                         begin
                             outtext("todaystrips ");outint(todaystrips,5);
                             outimage;
                         end;

                         lowa:=(centre-(spread/2))/6;
                         high:=(centre+(spread/2))/6;
                      for i := 1 step 1 until dim do
                         list(i) := randint(lowa,high,u4);
                      for i := 1 step 1 until dim * 3 do
                         prelist(i):=0;
                      base:=1;
                      for i := low step 1 until todaystrips do
                         begin
                             prelist(base) := randint(lowa,high,u4);
                             base:= base + 1;
                         end;
                      i := 1;
                      while returns(i) <> 0  and base < 100 do
                         begin
                             prelist(base) := returns(i);
                             base:= base + 1;
                             i:=i + 1;
                         end;
                      if base > dim * 3 then
                         begin
                             outtext(" trip overrun ");
                             outint(base-todaystrips,4);outtext(" *********");
                             base:=dim * 3 ;
                         end;
                         outint(base,4);
                         top :=  base-dim-1;
                      base := 1;
                      low := dim * 2;
                      for i := dim step 1 until low do
                         begin
                             list(i) := prelist(base);
                             base:= base + 1;
                         end;
                      for i :=  1 step 1 until top do
                         begin
                             trk := randint(low,len,u2);
                             while list(trk) <> 0  do
                                trk := randint(low,len,u2);
                             list(trk) := prelist(base);
                             base:=base + 1;
                         end;
                   end
                   else
                      for i:= 1 step 1 until len do

                         begin
                             list(i) := randint(lowa,high,u4);
                         end;
      END of gettrips;

   procedure maketasklists(trnum);
             integer trnum;
      begin
          integer i,j;
          for i:=1 step 1 until trnum do
             begin
                for j:=1 step 1 until 7 do trips(i,j):=0;
                for j:= 1 step 1 until 4 do
                trips(i,j):=list( i+(j-1)*20);
                  trips(i,8):=i;
             end;
      END of make;

      procedure printer;
      begin
             integer i,j;
          for i := 1 step 1 until dim do
             begin
                 for j := 1 step 1 until 8 do
                    outint(trips(i,j),8);
                 outimage;
             end;

      END of printer;


      Procedure newday(num);
                integer num;
      begin
             gettrips;
             maketasklists(num);
             allocate;
                if diagsetup then printer;
      end;

      Procedure allocate;
      begin
             integer i,j;
             for i :=1 step  1 until  dim do
                begin
                    for j := 1 step 1 until 4 do
                          if trips(i,j) <> 0 then
                          begin
                              trips(i,5):=trips(i,5) + comp(trips(i,j),1);
                              trips(i,6):=trips(i,6) + comp(trips(i,j),2);
                          end;
                       trips(i,7):= trips(i,6)+ trips(i,5) *2;
                end;
          END of allocate;

   integer len;
   integer array list(1:dim*4),prelist(1:dim*4);
      real b;
      boolean DIAGSETUP,STOCHASTIC;
      STOCHASTIC:= true;
```

```
      141             len := dim * 4;
      142             b:=trload/10;
      143             outtext("spread");outfix(spread,3,7);outimage;
      144          readin;
      145       end of setup;

...NOTE 123 LINE 40:   UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE

END SIMULA 3R8A. 145 LINES, NO ERRORS.


)SIMULA;M  TRIP;S
SIMULA 3R8A 74R1F2 12/27/82 13:32:26 (2)
        1     BEGIN
        2           INTEGER DIM,LEN,U,u1,u2,u4,rEPS,MAXLEN,i,j,centre,spread;
        3           REAL trload,alpha;
        4           DIM:=20;
        5           REPS:=5;
        6           LEN:=DIM * 4;
        7           MAXLEN:=DIM;
        8           centre:=200;
        9              U:=entier(timeused*2) + 1;
       10              u1:= u + 4;
       11              u2:= u1 + 2 ;
       12              u4:=u2 + 2;
       13              outtext("seeds");outint(u,15);outint(u1,15);
       14              outint(u2,15);outimage;
       15              OUTIMAGE;
       16              for  spread :=50 step 50  until 150 do
       17              begin
       18                 for trload := 42 step 3 until 54 do
       19               BEGIN
       20                       external class setupb;
       21                       external class schedule;
```

```
       23                 PROCEDURE  VARIATION(DATA,LAST,MEAN,VAR);
       24                              NAME MEAN,VAR;
       25                     REAL ARRAY DATA;
       26                        INTEGER LAST;
       27                           REAL MEAN,VAR;
       28                 BEGIN
       29                        INTEGER I ;
       30                        REAL TOTAL;
       31
       32                        TOTAL:=0;
       33                    FOR I:=1 STEP 1 UNTIL LAST DO OUTFIX(DATA(I),2,8);
       34                    OUTIMAGE;
       35                        FOR I := 1 STEP 1 UNTIL LAST DO
       36                           TOTAL:=TOTAL + DATA(I);
       37                        MEAN :=TOTAL/LAST;
       38                        TOTAL:=0;
       39                        FOR I :=1 STEP 1 UNTIL LAST DO
       40                           TOTAL:=TOTAL + (DATA(I)-MEAN)**2;
       41                        VAR:=SQRT(TOTAL/LAST);
       42                    END OF VARIATION;
       43
       44
       45                    procedure transcribe(datin,datout,col,size);
       46                                real array datin,datout;
       47                                integer col,size;
       48                    begin
       49                           integer i;
       50                           for i := 1 step 1 until size do
       51                              datout(i) := datin(i,col);
       52                    END of transcribe;
       53
```

```
55                  PROCEDURE DAYSTRIPS(DATE);
56                        INTEGER DATE ;
57                  BEGIN
58                      INTEGER I;
59                         FOR I:=1 STEP 1 UNTIL DIM DO
60                            begin
61                                MONTH(I,DATE,1):=trips(i,5) ;
62                                MONTH(I,DATE,2):=trips(i,6) ;
63                                MONTH(I,DATE,3):=trips(i,7) ;
64                                last(i,1):=last(i,1)+ month(i,date,1);
65                                last(i,2):=last(i,2)+ month(i,date,2);
66                                last(i,3):=last(i,3)+ month(i,date,3);
67                            end;
68                     END of daystrips;
69
70
71                  PROCEDURE gettodays;
72                  begin
73                         integer i,j;
74                         for i := 1 step 1 until dim do
75                           for j := 1 step 1 until 3 do
76                          trips(i,j+4):=trips(i,j+4) - last(i,j);
77                  END of gettodays;
78                  PROCEDURE printer;
79                  begin
80                         integer i,j;
81                         for i := 1 step 1 until dim do
82                         begin
83                                for j := 1 step 1 until 8 do
84                                outint(trips(i,j),8);
85                             for j := 1 step 1 until 3 do outint(last(i,j),8);
86                             outimage;
87                         end;
88                  END of printer;
```

```
90                  INTEGER I,R,J,K,COUNT,CYCLE,IDENT,day,datatype;
91                  REAL MEAN,TOTSQ,STD,TOTAL,BASE,INTERVAL;
92                  REAL ARRAY MONTH(1:DIM,1:23,1:3),tempdat(1:maxlen),
93                         FRED(1:MAXLEN,1:3),BILL(1:MAXLEN,1:3),
94                              ar(1:86);
95                  INTEGER ARRAY trips(1:dim,1:8),comp(1:100,1:2),
96                              returns(1:100),size(1:3),last(1:dim,1:3);
97                     BOOLEAN diagmain;
98                  ref(setupb) set;
99                  ref(schedule) s1;
100
101                    outtext("trload");outint(trload,10);outimage;
102
103                 FOR CYCLE:=1 STEP 1 UNTIL REPS DO
104                    BEGIN
105                     for i := 1 step 1 until 3 do
106                              SIZE(i):= 1;
107
108                        u:= u + 2;
109                        u1:=u1 + 2;
110                        u2:=u2 + 2;
111
112
113                   set:-new setupb(comp,trips,u,u1,u2,u4,trload,dim,returns,centre,spread
114
115
116
117
118                     FOR i :=1 step 1 until 20 do
119                        FOR j :=21 step 1 until 23 do
120                           FOR k := 1 step 1 until 3 do
121                              month(i,j,k) := 0;
122                     FOR i := 1 step 1 until 20 do
123                        for j := 1 step 1 until 3 do
124                           last(i,j) := 0;
125
126                     FOR day:=1 STEP 1 UNTIL 20 DO
127                        BEGIN
128                            SET.newday(dim);
129                             for r := 1 step 1 until 40 do returns(r):=0;
130                            s1:-new schedule(trips,last,comp,dim,returns);
131                           if diagmain then  printer;
132                            gettodays; if diagmain then printer;
133                            DAYSTRIPS(day);
134                        END;
135
136                    for datatype:=1 step 1 until 3 do
137                    begin
138                        FOR IDENT:=1 STEP 1 UNTIL DIM  DO
139                        BEGIN
140                                TOTAL:= 0;
141                                TOTSQ:=0;
142                                FOR day:= 1 STEP 1 UNTIL 20 DO
143                                   TOTAL:=TOTAL + MONTH(IDENT,day,datatype);
144                                      month(ident,21,datatype):=total;
145                                MEAN:=TOTAL/20;
146                                      month(ident,22,datatype):=mean;
147                                FRED(SIZE(datatype),datatype ):=TOTAL;
148                                FOR day := 1 STEP 1 UNTIL 20 DO
149                                   TOTSQ:=TOTSQ + (MONTH(IDENT,day,datatype)-MEAN)**2;
```

```
150                                     STD:=SQRT(TOTSQ/19);
151                                     month(ident,23,datatype):=std;
152                                     BILL(SIZE(datatype),datatype):=STD;
153                                     SIZE(datatype):=SIZE(datatype)+1;
154                                 END;
155
156                             END of datatype loop;
157
158
159                         for datatype := 1 step 1 until 3 do
160                             begin
161                                     outtext("DATATYPE "); outint(datatype,3);
162                                     outimage;
163                                 transcribe(fred,tempdat,datatype,maxlen);
164                                 VARIATION(tempdat,MAXLEN,MEAN,STD);
165                                     OUTTEXT("    MEAN TOTAL TRIP TIME");OUTREAL(MEAN,5,10);
166                                     OUTTEXT("    STD -TOTAL TRIP TIME");OUTREAL(STD,5,10);
167                                     OUTIMAGE;
168                                 BASE:=MEAN-STD*4.25;
169                                 INTERVAL:=STD/2;
170                                     transcribe(bill,tempdat,datatype,maxlen);
171                                 VARIATION(tempdat,MAXLEN,MEAN,STD);
172                                     OUTTEXT("    MEAN DEVIATION     ");OUTREAL(MEAN,5,10);
173                                     OUTTEXT("    STD -DEVS  TRIP TIME");OUTREAL(STD,5,10);
174                                     OUTIMAGE;
175                                 BASE:=MEAN-STD*4.25;
176                                 INTERVAL:=STD/2;
177                                 OUTIMAGE;
178                             end;
179                         end of cycle;
180                 END;
181             END;
182     END
```

END SIMULA 3R8A. 182 LINES, NO ERRORS.

)RESUME,P ASCII

CLASS BASIC function: maintains data storage , file access and statistical collection

Class runtimelists function: provides array of linked lists for trips pushed back , includes list insertion and removal .

Procedure setcptmasterfile
function: reads in data for all known compartments

Procedure opentripstat
Procedure openlogfile
Procedure opentripfile
Procedure cleartripfile
Procedure opendaylog
Procedure opentaskinput
function: opens and initializes all required input , data storage and output files .
Procedure gettime
Procedure wrdat
Procedure wrday
Procedure writehistory
Procedure cleardayvars
function: maintain and manipulate the main data storage area for truck statistics

Procedure getdump
function: gets next trip assignment from the despatch allocation table for despatcher

Procedure gettripdetails
function: gets time and distance details from the compartment master file for a specified compartment
Procedure Procedure checktrucklist
Procedure outpage
Procedure tripwriter
Procedure logheader
Procedure getrunsequence
functions: utilitys for display and startup

```
SIMULATION CLASS BASIC;

   BEGIN
        CLASS runtimelists;
        BEGIN

              Procedure updatebrk(idt,dntime,nowtime,grge);
                   integer idt,dntime,nowtime;boolean grge;
              begin
                    brkdnlst(1,pointerb):=idt;
                    brkdnlst(2,pointerb):=dntime;
                    brkdnlst(3,pointerb):=nowtime;
                   if grge then brkdnlst(4,pointerb):=1
                           else brkdnlst(4,pointerb):=0;
                    pointerb:=pointerb +1;
                    if pointerb > 50 then
                       begin
                           outtext("runtime error in brkdownlist");
                           outimage;
                       end;
              end;

              Procedure updatenotime(idt,rest);
                   integer idt,rest;
              begin
                    notmlst(1,pointern):=idt;
                    notmlst(2,pointern):=rest;
                      pointern:=pointern+1;
                    if pointern > 50 then
                       begin
                           outtext("runtime error in notimelist");
                           outimage;
                       end;
              end;

            Procedure outlists;
            Begin
                  inspect log do
                     begin
                         outimage;eject(1);setpos(50);outtext("BREAKDOWNS FOR DAY");
                         outimage;outimage;
                         for i :=1 step 1 until pointerb-1 do
                            begin
                                setpos(10);outint(brkdnlst(1,i),3);
                                outint(brkdnlst(2,i),5);outint(brkdnlst(3,i),5);
                                outint(brkdnlst(4,i),3);outimage;
                            end;
                         outimage;setpos(50);outtext("TRIPS PUSHED BACK");
                         outimage;outimage;
                         for i:=1 step 1 until pointern-1 do
                            begin
                                setpos(10);outint(notmlst(1,i),3);
                                outint(notmlst(2,i),5);outimage;
                            end;
                            outimage;
                     end of inspect;
             END of proc;

            integer array brkdnlst(1:4,1:50) ;
            integer array notmlst(1:2,1:100);
            integer pointerb,pointern,i;
               pointerb:=1;pointern:=1;
        end class runtimelists;

          PROCEDURE pushreturn(cpt);
                integer cpt;
          BEGIN
                outtext("pushreturn");outint(retptr,4 );
                return(retptr):= cpt;
                retptr:=retptr + 1;
          END OF pushreturn;


          procedure cleanupreturn;
          begin
                integer i;
                for i := 1 step 1 until 200 do return(i):=0;
                retptr:=1;
             end;
```

```
83
84              comment**************************************************
85              *         These procedures open all main files and initali
86              *                  where  necessary
87              **************************************************
88
89          PROCEDURE set_cpt_masterfile;
90              BEGIN
91                  text filename;
92                  integer csfval;
93                  ref (infile) logref;
94
95                  csfval:=csf("@asg,a  logmaster ");
96                   filename:- copy("logmaster") ;
97                  logref:- new infile(filename);
98                  outtext("          ----->COMPARTMENT MASTER OPENED          ");
99                  outtext("           "); outint(csfval,12);outimage;
100                 cptregstr:-new directfile("logfile");
101                 cptregstr.open(blanks(20));
102                 logref.open(blanks(20));
103                 while not logref.endfile do
104                    BEGIN
105                           logref.inimage;
106                           cptregstr.outtext(logref.image);
107                           cptregstr.outimage;
108                    end;
109                 logref.close;
110             END of set_cpt_masterfile;
111
112
113         PROCEDURE opentripstat;
114            begin
115                   text filid,opntxt,freetxt;
116                      integer csfval;
117                   filid:-copy("tripzzz      ");
118                   opntxt:-copy("@asg,up              ");
119                   freetxt:-copy("@free              ");
120                   filid.sub(7,5):=runseq;
121                   opntxt.sub(9,12):=filid;
122                   freetxt.sub(7,12):=filid;
123                   outint(csf(freetxt),10);
124                   csfval:=  csf(opntxt);
125                tripstat:- new directfile(filid);
126                tripstat.open(blanks(132));
127                tripstat.outtext(hdg);tripstat.outimage;
128
129                   outtext("----->FILE OPENED FOR TRIP STATISTICS  ");
130                   outtext(filid);outint(csfval,12);outimage;
131            end of opentripstat;
132
133
134         PROCEDURE open_logfile;
135            BEGIN
136                   integer csfval,freval;
137                   log:- new printfile("runlog");
138                   log.open(blanks(125));
139                   log.linesperpage(60);
140               log.outimage;
141                   print:=false;
142                   outtext("          ----->STD FILE OPENED FOR RUN MESSAGES ");
143                   outtext("           ");outint(csfval,12);outimage;
144            END of tripstat;
145
146         PROCEDURE opentripfile;
147         BEGIN
148                integer csfval;
149                csfval:= csf("@asg,a   tripfile");
150                tripfile:- new directfile ("tripfile");
151                tripfile.open(blanks(130));
152         END of opentripfile;
153
154         PROCEDURE cleartripfile;
155         BEGIN
156             integer i,j;
157             text lin;
158             lin:-blanks(130);
159                inspect tripfile do
160                   begin
161                       locate(1);
162                       for i:=1 step 1 until 200 do
163                          begin
164                              for j:= 1 step 1 until 130  do lin.sub(j,1).putint(0);
165                              lin.sub(3,3).putint(11);        ! points to next loc;
166                              lin.sub(6,3).putint(11);        ! points to start loc;
167                              outtext(lin);
168                              outimage;
169                           end
170                   end of inspect;
171                   outtext("          ----->TRIPFILE INITILIZED ");
172                   outimage;
173         END of cleartripfile;
174
175
176
177          PROCEDURE opendaylog;
178             BEGIN
179                      text filid,opntxt,freetxt;
180                      integer csfval,i,j;
181                      filid:-copy("daylogz      ");
182                      freetxt:-copy("@free              ");
183                      opntxt:-copy("@asg,up             ");
184                      filid.sub(7,5):=runseq;
185                      opntxt.sub(9,12):=filid;
186                      freetxt.sub(7,12):=filid;
187                      outint(csf(freetxt),10);
188                      csfval:=csf(opntxt);
189                   daylog:- new directfile(filid);
190                   daylog.open(blanks(125));
191                  outtext("----->FILE OPENED FOR DAY TOTALS ----- ");
192                  outtext(filid);outint(csfval,12);outimage;
193                         inspect daylog do
194                       begin
195                           outtext(hdg);outimage;
196                           for i :=1 step 1 until 638 do
197                              begin
198                                for j := 1 step 1 until 20 do
199                                   outint(0,5);outimage;
200                              end;
201                       end;
202
```

```
203             END of daylog;
204
205
206               PROCEDURE open_taskinput;
207                  BEGIN
208                       outimage;outtext("    FILE for task input ?");
209                       outimage;
210                       inimage;
211                       taskinput:-new infile(sysin.image.strip);
212                       taskinput.open(blanks(40));
213                  END;
214
215
216             comment ************************************************
217             *       PRIMATIVES used to manipulate the "truckdat array"
218             *                  which holds the operational statistics
219             *                    - used directly from main prog
220             ************************************************************;
221          procedure gettime(trk,tottime);
222                    name tottime;
223                    integer trk,tottime;
224          begin
225                 integer tempid;
226                 tempid:=trk;
227                 tottime:=truckdat(tempid,3);
228          end;
229
230       procedure wrdat(id,phase,val); integer id,phase,val;
231       begin
232
233
234           integer tempid,datloc;
235           tempid:=id;
236              truckdat(tempid,1):=phase;
237              if phase <= 9 then
238              truckdat(tempid,4):=truckdat(tempid,4)+val;
239              if phase = 2 or phase = 5 then
240                 begin
241                           truckdat(tempid,2):=truckdat(tempid,2) + val;
242                           truckdat(tempid,7):=truckdat(tempid,7)+val;
243                 end;
244              if phase < 8 then truckdat(tempid,3):=truckdat(tempid,3)+val;
245              if phase = 9 then truckdat(tempid,5):=truckdat(tempid,5)+val;
246              if phase = 10 then truckdat(tempid,6):=truckdat(tempid,6)+val;
247
248              datloc:=phase*2+6;
249              truckdat(tempid,datloc):=truckdat(tempid,datloc)+val;
250              truckdat(tempid,datloc+1):=truckdat(tempid,datloc+1)+1;
251
252       end of wrdat;
253
254          PROCEDURE wrday(id); integer id;
255             BEGIN
256                    integer n,val,tempid,i,paidtime,reptime,kms,totaltime,
257                                lastnum,drivetime;
258                     tempid:=id;
259                       drivetime:=truckdat(tempid,2);
260                       paidtime:=truckdat(tempid,3);
261                       totaltime:=truckdat(tempid,4);
262                       reptime:=truckdat(tempid,5);
263                       kms:=truckdat(tempid,6);
264
265
266                       inspect daylog do
267                          begin
268                               locate(tempid+3);
269                               inimage;
270                              lastnum:=image.sub(daynum*5,5).getint;
271                               image.sub(daynum*5,5).putint(paidtime);
272                               image.sub(daynum*5+5,5).putint(paidtime+ lastnum);
273                               locate(tempid+3);
274                               outimage;
275                               locate(tempid+130);
```

```
                    inimage;
                    lastnum:=image.sub(daynum*5,5).getint;
                    image.sub(daynum*5 ,5).putint(reptime);
                    image.sub(daynum*5+5,5).putint(reptime + lastnum);
                    locate(tempid +130);
                    outimage;
                    locate(tempid+257);
                    inimage;
                    lastnum:=image.sub(daynum*5,5).getint;
                    image.sub(daynum*5 ,5).putint(kms);
                    image.sub(daynum*5+5,5).putint(kms + lastnum);
                    locate(tempid +257);
                    outimage;
                    locate(tempid+384);
                    inimage;
                    image.sub(daynum*5 ,5).putint(totaltime);
                    locate(tempid +384);
                    outimage;
                    locate(tempid+511);
                    inimage;
                    image.sub(daynum*5 ,5).putint(drivetime);
                    locate(tempid +511);
                    outimage;
                end;
  END;

PROCEDURE writehistory;
BEGIN
      integer i,j;
      for i := 1 step 1 until plan_ntrk do
      history(i,1):=history(i,1) + truckdat(i,3);
      history(i,2):=history(i,2) + truckdat(i,6);
      history(i,3):=history(i,1)*2 + history(i,2);
END;


PROCEDURE clrdayvars(id); integer id;
  BEGIN
        integer i,pointer,loc;
        pointer:=id;
           truckdat(pointer,2):=0;
           truckdat(pointer,3):=0;
           truckdat(pointer,5):=0;
           truckdat(pointer,6):=0;

  END;









PROCEDURE getdump(cpt,did,pointer);integer cpt,did,pointer;
          name did,pointer;
 BEGIN
      integer gap,record,thiscpt,filelength;
      boolean above,below,found,finished;
      if diag1 then begin
         outtext("getdump");outint(cpt,4);outimage; end;
      INSPECT cptregstr do
      BEGIN
            above:=false;below:=false;finished:=false;
            found:=false;

            locate(1);
            inimage;
            filelength:=image.sub(1,5).getint+1; !add one for count;
            thiscpt:=0;
            gap:=filelength;
            WHILE gap >1  and not finished   do
               BEGIN
                     gap:=entier(gap/2);
                     if thiscpt < cpt then record:=record + gap;
                     if thiscpt > cpt then record:=record - gap;
                     locate(record);
                     inimage;
                     thiscpt:=image.sub(1,5).getint;
                     if thiscpt=cpt then
                     begin
                          finished:=true;
                          found:=true;
                     end
               END    !  gap is now 1 or cpt is found;

            WHILE NOT finished DO
               BEGIN
                     if cpt <thiscpt then
                        begin
                          record:=record -1;
                          above:= TRUE;
                        end;
                     if cpt >thiscpt then
                        begin
                          record:=record +1;
                          below:= TRUE;
                        end;
                     if record <=filelength   then
                     begin
                         locate(record);
                         inimage;
                         thiscpt:=image.sub(1,5).getint;
                     end

                   else
                         finished:=true;
                     if thiscpt = cpt then
                        begin
                          finished:=true;
                          found:=true;
                         end;
```

```
                                if above and below  then
                                    finished:=true;
                            END of search loop gap=1;

                            if  found then
                                begin
                                    did:=cptregstr.image.sub(6,5).getint;
                                    pointer:=record;
                                end
                            else
                                begin
                                    pointer:=0;
                                    did:=99;   !default cpt of truck;
                                end;
                    end of inspect;
            END of getdump;

        procedure writetrips;
        begin
                integer i,col,top;
                cleartriparray; ! zero array;
                listptr:=listptr-1;  ! left one too high by setuplist;
                outtext(" actual list length");outint(listptr,4);outimage;
                if listptr > plan_ntrk * 4 then
                begin
                        outtext("overflow in list "); outimage;
                        listptr:=plan_ntrk * 4;
                end;
                top:=listptr;
                col:=1;
                while top > plan_ntrk do
                    begin
                        for i := 1 step 1 until plan_ntrk do
                            begin
                                trips(i,col):=lstcpt(listptr);
                                listptr := listptr - 1;
                            end;
                        top := listptr;
                        col := col + 1;
                    end;
                    while listptr > 0 do
                        begin
                            i := randint(1,plan_ntrk,u5);
                            while trips(i,col) <> 0 do
                            i := randint(1,plan_ntrk,u5);
                            trips(i,col) := lstcpt(listptr);
                    if diag1 then begin outtext("write");outint(lstcpt(listptr),4);
                        outint(listptr,4);outint(i,2);outimage;end;
                        listptr := listptr - 1;
                        end;
                    for i := 1 step 1 until plan_ntrk do
                        trips(i,8):=i;
            END of write trip;



            comment*****************************************************
            *       gettripdetails works for the despatcher
            *             using the pointer to get dump, traveltime etc
            ************************************************************;

        PROCEDURE GETTRIPDETAILS(pointer,dump,outtime,mediant,dist);
                                        name dump,outtime,mediant,dist;
                                integer pointer,dump,outtime,mediant,dist;
        BEGIN
            integer temppoint;
            temppoint:=pointer;
            if temppoint = 0 then temppoint:= 20;
            inspect cptregstr do
                begin
                    locate(temppoint);
                    inimage;
                    dump:=image.sub(6,5).getint;
                    mediant:=image.sub(11,5).getint;
                    dist:=image.sub(16,5).getint;
                        outtime:=(mediant-25)/3;
                end of inspect;
        END of gettripdetails;

        PROCEDURE gettime_dist(cpt,ttime,dist);
                name ttime,dist;
                integer ttime,dist,cpt;
        begin
                integer point,dump,outt;
                getdump(cpt,dump,point);
                gettripdetails(point,dump,outt,ttime,dist);
        end;
```

```
PROCEDURE check_trucklist;              ! makes sure that;
BEGIN                                   ! some trucks are used;
      integer i;

      if ntruck <= 0 then
         begin
               outtext("no trucks requested CHECK TASKLIST!!!");
               outimage;
         end
END of check_trucklist;

PROCEDURE outpage(intarr,title,max);integer array intarr;
                        text title;integer max;
   BEGIN
       integer count,pointer,i,j;
      INSPECT log do
       BEGIN
           eject(1);
           setpos(50);outtext(title);outimage;
          eject(5);
           count:=1;
           while count<=max do
              begin
                  for j:=1 step 1 until 40  do
                     begin
                         for i :=0 step 1 until 9 do
                            begin
                               pointer:=count + (I*40);
                               if pointer <= max then outint(intarr(pointer),1
                            end of one line;
                         outimage;
                         count:=count + 1;
                  end;
                  eject(1);
                  count:=count + 400;
           end;
       end of inspect;
   END of outpage;

PROCEDURE tripwriter;
begin
       integer i,j;
       for i := 1 step 1 until plan_ntrk do
       begin
           for j := 1 step 1 until 8 do
              outint  (trips(i,j),8);
           outimage;
       end;
END of tripwriter;

PROCEDURE CLEARTRIPARRAY;
begin
       integer i,j;
       for i := 1 step 1 until plan_ntrk do
          for j := 1 step 1 until 4 do
             trips(i,j):=0;
       outtext(" trips cleared");outimage;
END of cleartriparray;
PROCEDURE SET_HDG;

      BEGIN
            integer i;
            text d;
            hdg:-blanks(90);
            d:-copy(date);
            hdg.setpos(1);
            for i :=1 step 1 until 20 do
               hdg.putchar('*');
            hdg.sub(21,20):= copy("  SIMULATION RUN ON " );
            hdg.sub(45,2):=d.sub(1,2);
            hdg.sub(48,2):=d.sub(3,2);
            hdg.sub(51,2):=d.sub(5,2);
            hdg.sub(55,6):= copy(" AT ");
            hdg.sub(62,2):= d.sub(7,2);
            hdg.sub(66,2):= d.sub(9,2);
            hdg.sub(70,2):= d.sub(11,2);

      END of set_hdg;

PROCEDURE get_runseq;
   begin
       text txt;
       txt:-blanks(100);
       outimage;outtext("                    ");
       outtext(" What run sequence number for this run ?--");
       outimage;
       inimage;
                   runseq:- copy(sysin.image.sub(1,5));
       outtext(image.sub(1,5));
       outtext(" RUN SEQUENCE NUMBER FOR THIS RUN IS --");
          outtext(runseq);outtext(image.sub(1,5));outimage;
   end;

PROCEDURE logheader;
Begin;
       inspect log do
          begin
              eject(1);outtext(hdg);setpos(110);outtext("DAY NUM");
              outint(daynum,4);outimage;outimage;
          end;
END;
```

```
583           ref(printfile) log;
584           ref(directfile) daylog;
585           ref(directfile) cptregstr;
586           ref(directfile) tripstat  ;
587           ref(directfile) tripfile;
588           ref(infile) taskinput;
589           ref(runtimelists) list;
590
591
592           boolean array truck_required(1:200);
593           boolean array dumprequired(1:100);
594
595           integer array truckdat(1:125,1:30);
596           integer array wakelist(1:200); ! holds wakeuptime;
597           integer array dumplist(1:100); ! holds loads for day;
598           integer array trips(1:125,1:8), return(1:200),listcpt(1:400);
599           integer array history (1:125,1:3);
600
601           integer ntruck,daynum,retptr,listptr,plan_ntrk,u5,wakeu;
602           boolean print,anotherday  ,diag1;
603           text hdg,runseq;
604              daynum:=0;
605              u5:= 3030303;
606              wakeu :=101010101;
607              outtext("HOW MANY TRUCKS ?");outimage;
608              inimage;plan_ntrk:=inint;outint(plan_ntrk,5);outimage;
609
610                 set_hdg;
611                 outtext(hdg);outimage;outimage;
612                 get_runseq;
613                 open_logfile;
614                 opentripstat;
615                 opendaylog;
616                 opentripfile;
617                 cleartripfile;
618                 list:-new runtimelists;
619                 open_taskinput;
620        END

.NOTE 123 LINE 42:   UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE
.NOTE 123 LINE 50:   UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE

:D SIMULA 3R8A. 620 LINES, NO ERRORS.


IMULA,M     SIM4.DATASET,,BL/1
:MULA 3R8A 74R1F2 12/27782 13:26:54 (42)
```

APPENDIX C

CLASS DATASET ("Direct allocation version")

function: sets up daily despatch allocation from an input of the backlog and the new days demand

Procedure getdump
function: gets data on dump location from masterfile

Procedure instaltrip
function: puts a trip into the despatch allocation table

Procedure updatedumplist
Procedure updatetrucklist
function: updates status lists of trucks and dumps active

Procedure wake
function: assigns a daily startup time to each truck on each day

Procedure startday
function: supervises the collection of trip requirement from backlog and new demand and instalation in the daily allocation table

Procedure checkoutput
function: provides a daily display of active units

Procedure getassignment
Procedure gettripdetails
function: get information for despatcher from despatch allocation and compartment master file .

```
external class basic;
basic class dataset;
BEGIN


    comment**********************************************************
    *       getdump works with external master file "CPTREGSTR"
    *               to get cpt details
    **********************************************************;


PROCEDURE getdump(cpt,did,pointer);integer cpt,did,pointer;
          name did,pointer;
BEGIN
       integer gap,record,thiscpt,filelength;
       boolean above,below,found,finished;

       INSPECT cptregstr do
       BEGIN
              above:=false;below:=false;finished:=false;
              found:=false;

              locate(1);
              inimage;
              filelength:=image.sub(1,5).getint+1; !add one for count;
              thiscpt:=0;
              gap:=filelength;
              WHILE gap >1  and not finished   do
                 BEGIN
                        gap:=entier(gap/2);
                        if thiscpt < cpt then record:=record + gap;
                        if thiscpt > cpt then record:=record - gap;
                        locate(record);
                        inimage;
                        thiscpt:=image.sub(1,5).getint;
                        if thiscpt=cpt then
                        begin
                              finished:=true;
                              found:=true;
                        end
                 END    ! gap is now 1 or cpt is found;


              WHILE NOT finished DO
                 BEGIN
                        if cpt <thiscpt then
                           begin
                              record:=record -1;
                              above:= TRUE;
                           end;
                        if cpt >thiscpt then
                           begin
                              record:=record +1;
                              below:= TRUE;
                           end;
                        if record <=filelength   then
                        begin
                           locate(record);
```



```
                              inimage;
                             thiscpt:=image.sub(1,5).getint;
                        end

                     else
                              finished:=true;
                        if thiscpt = cpt then
                           begin
                              finished:=true;
                              found:=true;
                           end;
                        if above and below  then
                              finished:=true;
                     END of search loop gap=1;

                     if  found then
                        begin
                           did:=cptregstr.image.sub(6,5).getint;
                           pointer:=record;
                        end
                     else
                        begin
                           pointer:=0;
                           did:=99;   !default cpt of truck;
                        end;
              end of inspect;
       END of getdump;
```

TRUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

```
 91
 92                  comment**********************************************************
 93                  *       installtrip work with internal scratch file "tripfile"
 94                  *                  to build up one days work assignment for
 95                  *                    the despatcher
 96                  *             store provides the actual file interface
 97                  *
 98                  *             updatedumplist marks each active dump in the
 99                  *                      master array "dumplist"
100                  **********************************************************;
101
102
103           PROCEDURE installtrip(trnum,cpt); integer trnum,cpt;
104                 begin
105
106                       PROCEDURE logfile3;
107                             begin
108                                   log.outtext("truck number");
109                                   log.outint(trnum,4);
110                                   log.outtext("no such compartment");
111                                   log.outint(cpt,5);
112                                   log.outtext("*********");log.outimage;
113                             end;
114
115
116                       PROCEDURE store(truckid,cptid,dumppointer);
117                                 integer truckid,cptid,dumppointer;
118                       BEGIN
119                             integer cptsasg,freeloc,cpt;
120                             text trline;
121
122                             cpt:=cptid;
123                             inspect tripfile do
124                                 begin
125                                     locate(truckid);
126                                     inimage;
127                                     trline:= image;
128                                     cptsasg:=trline.sub(1,2).getint;
129                                     if cptsasg < 12 then
130                                        begin
131                                             freeloc:=trline.sub(3,3).getint;
132                                             trline.sub(freeloc,5).putint(cptid);
133                                             trline.sub(freeloc+5,5).putint(dumppointer);
134                                             trline.sub(1,2).putint(cptsasg+1);
135                                             trline.sub(3,3).putint(freeloc+10);
136                                             locate(truckid);
137                                            outtext(trline);
138                                             outimage;
139                                        end
140                                    else
141                                        begin
142                                            inspect notdone do putback(truckid,cpt);
143                                            sysout.outtext(" overflow in tripfile for truck")
144                                            sysout.outint(truckid,5); sysout.outimage;
145                                        end;
146                             end of inspect;
147                       END of store;
148
149
150
```

TRUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

```
151                       PROCEDURE update_dumplist(dumpid); name dumpid;integer dumpid;
152                           begin
153                               if dumpid <= 100 then
154                                  begin
155                                        dumprequired(dumpid):=true;
156                                        dumplist(dumpid):=dumplist(dumpid)+1;
157                                  end
158                               else
159                                  begin
160                                        outtext(" funny dump at ");outint(line,5);
161                                        outtext(" of taskinput "); outimage;
162                                  end;
163                       END of update_dumplist;
164
165                       PROCEDURE update_trucklist;
166                          BEGIN
167                               if trnum<= 200 then
168                                 begin
169                                        truck_required(trnum):=true;
170                                        wakelist(trnum):=wake;
171                                         ntruck:=ntruck + 1;
172                                 end
173                               else
174                                  begin
175                                       outtext(" funny truck at ");outint(line,5);
176                                       outtext(" of taskinput "); outimage;
177                                  end;
178                       END of update_trucklist;
179
180                       INTEGER PROCEDURE wake;
181                       BEGIN
182                             wake:=entier(normal(wakemu,wakesigma,wakeu));
183                       END;
184
185                             integer dump,dumppointer,wakeu;
186                             real wakemu,wakesigma;
187                             wakemu:=110;wakesigma:=10;
188                                 wakeu:=10101010;
189                                   getdump(cpt,dump,dumppointer);
190                                   if dump ne 99 then
191                                     begin
192                                           store(trnum,cpt,dumppointer);
193                                           update_dumplist(dump);
194                                           update_trucklist;
195                                     end
196                                   else
197                                     begin
198                                          logfile3;
199                                          sysout.outtext("no such compartment");
200                                          sysout.outtext("*********");
201                                          sysout.outint(cpt,5);sysout.outimage;
202                                     end;
203                 END of installtrip;
204
```

```
TRUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

      206               comment ******* startday may become redundant;
      207
      208           PROCEDURE STARTDAY;
      209            BEGIN
      210                     comment**************************************************
      211                     *         setuptasklist  first reads leftovers from notdone
      212                     *                 then the main days work from externalfile
      213                     *                  "tasklist".  scheduler will have to fit
      214                     *                   in here somewhere.
      215                     **************************************************;
      216
      217                   PROCEDURE SET_UP_TASKLIST;
      218                   BEGIN
      219
      220
      221
      222
      223
      224                      INTEGER trucknum,thisday,cpt,count,i,day;
      225                        BOOLEAN tomorrow;
      226
      227                      pointer:=1;
      228                      ntruck:=0;
      229                      line:=0;
      230                         daynum:=daynum + 1;
      231                         if print  then logheader;
      232
      233                         for i:=1 step 1 until 200 do
      234                            begin
      235                                DO
.....................................................I
WARNING 147 LINE 235:  THIS NON-SIMULA FEATURE WILL BE WITHDRAWN - PLEASE CHANGE IT ...
      236                                        begin
      237                                            inspect notdone do pop_yesterday(i,cpt);
      238                                            if cpt > 0 then
      239                                            begin installtrip(i,cpt);count:=count+1;
      240                                                outtext("truck");outint(i,3);outtext("cpt");
      241                                                outint(cpt,5);   outimage; end;
      242                                        end
      243                                    UNTIL cpt = 0
      244                            end;
      245                            outtext("   Trips from yesterday --- ");outint(count,5);
      246                            outimage;
      247                         inspect taskinput do
      248                            BEGIN
      249
      250                             tomorrow := false; !note thisday comes from dummy read;
      251                             inimage;
      252                             if not endfile then thisday := image.sub(1,2).getint;
      253                             sysout.outint(thisday,5);sysout.outimage;
      254                             WHILE NOT endfile and NOT tomorrow  do
      255                                BEGIN
      256                                    day:=image.sub(1,2).getint;
      257                                       day :=  image.sub(1,2).getint;
      258                                       if day ne thisday then tomorrow := true
      259                                       else
      260                                          begin
      261                                              cpt:=image.sub(21,5).getint;
      262                                              trucknum:=image.sub(11,5).getint;
      263                                              if cpt <> 0 then installtrip(trucknum,cpt);


TRUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

      264                                          inimage;
      265                                          line:=line + 1;
      266                                       end;
      267                                end;
      268                         END of inspect;
      269                   END of setup ;
      270
      271
      272
```

TRUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

```
274
275                     PROCEDURE checkoutput;
276                       BEGIN
277                           INSPECT LOG DO
278                              BEGIN
279                              setpos(50);outtext("TRUCK LIST");outimage;
280                               outimage;
281                              for I:=1 step 1 until 125 do
282                                 if truck_required(i)then outchar('T') else outchar('N'
283                                 outimage;
284                               for i:=1 step 1 until 13 do outtext("    !    |");
285                              outimage;
286                               for i:=10 step 10 until 120 do outint(i,10);
287                              outimage;outimage;
288                                setpos(50);outtext(" DUMPS REQUIRED");outimage;outimag
289                                for I:=1 step 1 until 100 do
290                               if dumprequired(i) then outchar('T') else outchar('N');
291                                 outimage;
292                                 for i:=1 step 1 until 10 do outtext("    !    |");
293                                 outimage;
294                                 for i:=10 step 10 until 100 do outint(i,10);
295                              outimage;outimage;
296
297                              setpos(50);
298                              outtext("  DUMP STATS  ");
299                              outimage;outimage;
300                              outtext("  ID   ");
301                              for i:=1 step 1 until 100 do
302                              if dumprequired(i) then
303                                 outint(i,3);
304                                 outimage;
305                                 outtext(" DUMP ");
306                              for i:=1 step 1 until 100 do
307                              if dumprequired(i) then
308                                 outint(dumplist(i),3);
309                                 outimage;
310                           END of inspect;
311                     END of checkoutput;
312
```

TRUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

```
314                 integer i,pointer,line;
315                    set_up_tasklist;
316                    check_trucklist;
317                    if print then checkoutput;
318
319             END OF STARTDAY;
320
```

```
RUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

                    comment**************************************************
                    *       getassignment works for despatcher to interogate
                    *          the master file of the days work held in
                    *            "tripfile", built up by installtrip
                    **************************************************;

                PROCEDURE GETASSIGNMENT(truckid,cpt,cptptr);
                                        name cpt,cptptr;
                                        integer truckid,cpt,cptptr;
                BEGIN
                    integer loc,ident;
                    ident:=truckid;
                    inspect tripfile do
                        begin
                            locate(ident);
                            inimage;
                            loc:=image.sub(6,3).getint;
                            cpt:=image.sub(loc+1,4).getint;
                            if cpt = 0 then
                                begin
                                    image.sub(10,1).putint(1);
                                                    ! flag whole record as finished
                                                      NOTE that end of day is picked
                                                      up as zero ctp number;
                                end
                            else
                                begin
                                    image.sub(6,3).putint(loc+10);
                                    cptptr:=image.sub(loc+5,5).getint;
                                    image.sub(8,2).putint(image.sub(8,2).getint+1);
                                                    ! update cpts done counter;
                                    image.sub(loc,1).putint(1);
                                                    ! flag this one as done;
                                end;

                            locate(ident);
                            outtext(image);  ! store updated record;
                            outimage;
                        end of inspect
                END of getassignment;

                    comment**************************************************
                    *     gettripdetails works for the despatcher
                    *        using the pointer to get dump, traveltime etc
                    **************************************************;

                PROCEDURE GETTRIPDETAILS(pointer,dump,outtime,mediant,dist);
                                        name dump,outtime,mediant,dist;
                                    integer pointer,dump,outtime,mediant,dist;
                BEGIN
                    integer temppoint;
                    temppoint:=pointer;
                    inspect cptregstr do
                        begin
                            locate(temppoint);
                            inimage;
                            dump:=image.sub(6,5).getint;
                            mediant:=image.sub(11,5).getint;
                            dist:=image.sub(16,5).getint;
                                outtime:=(mediant-25)/3;


RUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

                        end of inspect;
                END of gettripdetails;
```

```
TRUCKFLEET SIMULATOR ----DATASTRUCTURE SUBSYSTEM

      386              ref(head) repairshop;
      387              repairshop:- new head;
      388              set_cpt_masterfile;
      389       end of dataset;

WARNING 120 LINE 235:  NON-SIMULA CONSTRUCTION - THE PROGRAM IS  N O T  PORTABLE

END SIMULA 3R8A. 389 LINES, NO ERRORS.


@SIMULA,M  SIM2,1
SIMULA 3R8A 74R1F2 12/27/82 13:29:16 (49)
      1    %rtsoptions  = "u" ;
.........................I
WARNING 182 LINE 1:     ILLEGAL COMPILER DIRECTIVE OR MISPLACED % - SKIPPED
```

APPENDIX D

CLASS CHIPMILL

function: provides the main body of the simulation model

Class truck
function: provides the template model of the truck with a data structure and controls the sequence of activities for objects of this type

Class dump
function: provides a landing model with a loader and a 'wait loader' queue
Class unloader
function: provides the mill terminal model with an unloader queue and unloader

Class despatcher
function: provides the despatcher model , working from a supplied despatch allocation table

Class reporter
function: provides the runtime VDU display of system activity at user supplied fixed intervals

setup
function: provides the daily startup routine by calls to class dataset to setup the despatch allocation table , then starts the required trucks and loaders

endday
function: provides daily shutdown , mostly by looking in the repairshop for any trucks still broken down at close of day and pushing their trips back onto the backlog for next day . Trucks are otherwise sent to the garage by the despatcher when they have completed work .

Action body of chipmill
function: provides a reference and keeps track of all the truck , dump , unloader , and despatcher objects

```
BEGIN
        external class basic;
        external class dataset;
         dataset CLASS CHIPMILL;
      BEGIN

            PROCESS CLASS Truck(trucknum);integer trucknum;
            begin


                PROCEDURE roundtrip;
                !************************;
                  begin
                        status:="S";
                        INTO(wait_despatcher);
                        getthere:=time;
                        if NOT sam.despatcher_busy then ACTIVATE sam;
                        passivate;
                       if finishedwork THEN GOTO ENDDAY;
                        gettripdetails(cptptr,dumpnum,outtime,mediant,kms);
                        if firsttrip then
                        begin
                             milllikely:=mod((time + tripestimate),1600);
                             millarrive:=randint(120,165,u3);
                             if milllikely < millarrive then
                               begin
                                   tooearly:=millarrive-milllikely;
                                      hold(tooearly);
                                      getthere:=getthere + tooearly;
                               end;
                             firsttrip:=false;
                         end;
                          TODAYSTRIPS:=TODAYSTRIPS + 1;

                        wrdat(trucknum,10,kms);
                        d:=landing(dumpnum);
                        timeout:=(logn_ttime(mediant))/3;
                        wd:=time-getthere;

                           if wd > 0 then wrdat(trucknum,1,wd);

                        despatch:=time;
                    !******DRIVE OUT******************;

                         status:="D";
                         drout:=normal(timeout,std,u);
                         if drout <0 then drout:=2;
                          HOLD(drout);

                          wrdat(trucknum,2,drout);                !update driveout;

                      !************LOAD *************;

                         getthere:=time;
                        HOLD(5); !prepare to load;
                        INTO(d.loadque);
                        status:="W";
                        if NOT d.loaderbusy THEN ACTIVATE d ;
                        PASSIVATE;

                        HOLD(10); ! chain up;
                        leave:=time;
                        ondump:=leave-getthere;
                         wl:= ondump-ld;

                           if wl > 0 then  wrdat(trucknum,3,wl); !update waitload;
                           wrdat(trucknum,4,ld);                 !update load;

                    !************ DRIVE IN ***********;

                        status:="d";
                        drin:=timeout*2.0;
                        if drin < 6  then drin:=6 ;
                        if drin > 300 then drin:=300 ;
                        HOLD(drin);

                        wrdat(trucknum,5,drin);

                    !************ UNLOAD AT MILL ************;
                         getthere:=time;
                          hold(3); ! weighin;
                          INTO(unloadque);
                           status:="M";
                           if NOT Wagner.busy then ACTIVATE Wagner;
                            PASSIVATE;

                             hold(3); !weigh off;
                            leave:=time;
                            inmill:=leave-getthere;
                            unload:=leave-unloadstart;
                            waitmill:=inmill-unload;
                             wrdat(trucknum,6,waitmill);
                             if waitmill > 0 then wrdat(trucknum,6,waitmill);
                              wrdat(trucknum,7,unload);              !update unload;
                    if DRAW(refuelprob,u1) then
                    begin
                           status:="R";
                           pcrst:=10;
                           pcrst:=normal(pcrstmu,pcrstsigma,u4);
                             if pcrst < 8 then pcrst :=8;
                           HOLD(pcrst);
                           wrdat(trucknum,8,pcrst);   !update  refuel;

                    end of refuel;

                    if DRAW(breakdownprob,u1) then
                    begin
                           integer RESTOFDAY;
                           real mubrkdn,sigmabrkdn;
                           mubrkdn:=200;sigmabrkdn:=200;
                           status:="B";
                           REPAIRTIME := NORMAL(mubrkdn,sigmabrkdn,U2);
                           if REPAIRTIME < 10 then REPAIRTIME := 10;
                           restofday:= 800- mod(time,1000);
                       outtext("BREAKDOWN");outint(trucknum,3);outint(REPAIRTIME,5);
                       outint(RESTOFDAY,5); outimage;
                           if REPAIRTIME > RESTOFDAY then grge := true;
                             inspect list do
                                    updatebrk(trucknum,repairtime,time,grge);
                              if     grge
```

```
                        then
                            begin
                                repairtime:=repairtime - restofday;
                                wrdat(trucknum,9,restofday);
                                wait(repairshop);
                            end
                        else
                            wrdat(trucknum,9,repairtime);
                                HOLD(repairtime);
                                setpos(20);outtext("REPAIR DONE Truck");
                                outint(trucknum,4);outimage;
                    grge:=false;
            END;

            wrdat(trucknum,11,1);              !update trips;
            wrdat(trucknum,10,kms);              !update kms;
        END of roundtrip;


    PROCEDURE  TRIP_WRITE;
    begin
            inspect tripstat do
              begin
                outint(trucknum,3);outint(cpt,5);
                outint(despatch,7);outint(drout,7);
                outint(wl,7);outint(ld,7);
                outint(drin,7);
                outint(inmill,7);outint(pcrst,7);
                outint(rep,7);
                outint(kms,7);outint(time,7);
                outimage;
              end of inspect;
    END of tripwrite;


    INTEGER  PROCEDURE logn_ttime(median);
                    integer median;
    BEGIN
            integer mya;
            real mym,mys,    aa,ab,    ma,mb,    sa,sb;
            aa:=3.22;      ab:=0.7452;
            ma:=2.926;     mb:=0.00344;
            sa:=0.7175;    sb:=-0.000456;

            mya:=aa + ab * median;
            mym:=ma + mb * median;
            mys:=sa + sb * median;

            logn_ttime := mya + entier(exp(normal(mym,mys,u5)));
    END of logn;


       REF(dump) d;
       REAL timeout,total,dt,a,std,refuelprob,breakdownprob;
       real pcrstmu,pcrstsigma;
       CHARACTER status;
        integer u,u1,u2,u3,start, finish, cpt,outtime,cptptr,
             dumpnum, u4,u5,  ! u5 for logn,u4 for service station;

                    id
                    drout   ,
                    dayswork,
                    wl      ,
                    ld      ,
                    drin    ,
                    unloadstart,unload,waitmill,
                    pcrst   ,
                    rep     ,
                    kms     ,tooearly,
                    repairtime,millarrive,tripestimate,milllikely,
                    getthere,leave , despatch,inmill,ondump,mediant;

      BOOLEAN FINISHEDWORK,grge,firsttrip;
        status:="s";
        std:= 1.0;
        u:= ((trucknum*2)+10001);
        u1:=trucknum * 10  + 1;
        u2:=trucknum * 100 +1;
        u3:=trucknum * 1000  + 1;
        u4:=trucknum * 10000+ 1;
        u5:=trucknum * 100000 + 1;
        refuelprob:=.5;
        breakdownprob:=.03;
        pcrstmu:=10.0;
        pcrstsigma:=3.0;

        wait(garage);
         firsttrip:=true;

    BEGINDAY:
        HOLD(wakelist(trucknum));
        HOLD(repairtime); !/* will be 0 unless come from garage*/;

        while TRUE do
           BEGIN
                    roundtrip;
                    trip_write;
           end;
    ENDDAY:
             status:="f";
             wait(garage);
             FINISHEDWORK:= false;
             goto BEGINDAY ;

    END of truck;
```

```
201                 PROCESS CLASS dump(id);integer id;
202                    BEGIN
203
204                         procedure report;
205                         begin
206                                 outtext(" dump");outint(id,3);
207                                 outtext("loader ");outfix(time,3,7);
208                                 outimage;
209                         end;
210
211                         REF(truck) thisone;
212                         REF(head) loadque;
213
214                            BOOLEAN loaderbusy;
215                               REAL loadmu,loadsigma;
216                               INTEGER u,loadt;
217                                  loadmu:=15.0; loadsigma:=3.0;
218                               u := (id) * 200 + 1;
219                            loadque:-new head;
220                            WHILE TRUE DO
221
222                            BEGIN loaderbusy:=true;
223                                 WHILE NOT loadque.EMPTY DO
224                                    BEGIN thisone:-loadque.first;
225                                          thisone.out;
226                                          thisone.status:='L';
227                                          loadt:=normal(loadmu,loadsigma,u);
228                                          if loadt < 5 then loadt :=7;
229                                          HOLD(loadt);
230
231                                          thisone.ld:=loadt;
232                                          ACTIVATE thisone;
233                                    END;
234                                 loaderbusy:=false;
235                                 PASSIVATE;
236
237                            END;
238                 END of dump;
239
```

```
241                 PROCESS CLASS unloader;
242                 BEGIN
243
244                         REAL  PROCEDURE unloadtime;
245                            BEGIN
246                                    real unld;
247                                    unld:=normal(mu,std,u);
248                                      if unld < 1.0 then unloadtime := 4.0
249                                                    else unloadtime:= unld;
250                            END of unloadtime;
251
252                         REF(truck) mynext;
253                         INTEGER u;
254                         REAL mu,std;
255                         BOOLEAN busy;
256
257                         mu:=3.0;std:=1.0;u:=1010101;
258
259                         WHILE TRUE  DO
260                            BEGIN
261                                    busy:=TRUE;
262                                    WHILE NOT Unloadque.empty DO
263                                       BEGIN
264                                             mynext:-Unloadque.first;
265                                             mynext.out;
266                                             mynext.status:='U';
267                                             mynext.unloadstart:=TIME;
268                                             HOLD(unloadtime);
269
270                                             ACTIVATE mynext;
271                                       END of one unload cycle;
272                                    busy:=false;
273                                    PASSIVATE;
274
275                             END of unloader body;
276
277                     END of unloader class;
278             !*********************************************;
279
```

```
PROCESS CLASS despatcher;
BEGIN
          procedure display;
          begin
              integer i;
              for i:=1 step 1 until 100 do
                 if truck_required(i) then outchar('t') else outchar('n');
              outimage;
          end display;

      procedure report;
      begin
          INSPECT LOG DO
             BEGIN
                 outtext(" TRK ");outint(thistruck.trucknum,3);
                 outtext(" C ");outint(cpt,4);
              END of inspect;
      end;

      PROCEDURE clr_trip_vars;
      begin
             inspect thistruck do begin
                 drout:=0;
                 wl:=0;
                 ld:=0;
                 rep:=0;
                 drin:=0;
                 inmill:=0;
                 pcrst:=0;
                 wd:=0;
                 kms:=0;
                 despatch:=0;
             end;
      END of clr_trip_vars;

         ref(truck)thistruck;
         integer nextdump,cpt,tr_id,cptptr,outtime,dumpid,kms,
                 restofday,esttriptime,notime,todaytime,mediant;
         boolean despatcher_busy;

         display;
         passivate;

            WHILE TRUE DO
               begin despatcher_busy:=TRUE;
                  WHILE NOT wait_despatcher.empty DO
                  begin
                        thistruck:-wait_despatcher.first;
                        thistruck.out;
                        inspect thistruck do  clr_trip_vars;
                        tr_id:=thistruck.trucknum;
                        getassignment(tr_id,cpt,cptptr);
                        gettripdetails(cptptr,dumpid,outtime,mediant,kms);
                        gettime(tr_id,todaytime);
                        restofday:=950- todaytime;
                        esttriptime:=outtime*2.5 + 30 ;
                        thistruck.tripestimate:=esttriptime;

                        report;
                        if  cpt <> 0 and restofday > esttriptime then
                           begin  !cpt = 0 means no more work from despatch
                                  thistruck.cpt:=cpt;
                                  thistruck.cptptr:=cptptr;
                                  log.outtext(" TIME ");log.outint(todaytim
                                  HOLD(.1);
                                  REACTIVATE thistruck ;
                           end
                     else
                        begin
                              thistruck.status:='F';
                              thistruck.finish:=time;
                              thistruck.FINISHEDWORK:= true;
                              if cpt ne 0 then
                                 begin
                                          pushreturn(cpt);
                                       notime:=notime + 1;
                                       setpos(40);
                                       outtext("NOTIME  Truck ");
                                       outint(tr_id,3);
                                       outint(cpt,5);outimage;
                                       inspect log do
                                          outtext(" NO TIME  ");
                                    inspect list do
                                          updatenotime(tr_id,restofday);
                                    end;
                               if cpt eq 0 then begin
                                       log.outtext(" FIN  ");
                                       log.outint(todaytime,4);
                               end;
                              HOLD(.01);
                              REACtIVATE thistruck;
                        end;
                  END of busy loop;
                  despatcher_busy:=false;
                  passivate;
            END of TRUE loop;
   END of despatcher;
!************************************;
```

```
410             PROCESS CLASS REPORTER;
411                begin
412                     integer i;
413                     while true do
414                     begin
415                          outint(time,5);
416                          for i:=1 step 1 until 120 do begin
417                                if fleet(i) =/= none then
418                                   outchar(fleet(I).status) else outchar("*");
419                           end;
420                           outimage;
421                           HOLD(100);
422
423                     end;
424             end reporter;
425
426
427          comment  START OF MAIN PROGRAM ************;
428
429           PROCEDURE  SETUP;
430              BEGIN
431
432                   PROCEDURE setup_dumps;
433                      begin
434                          integer i;
435                             for i:=1 step 1 until 100 do
436                             begin
437                                   if dumprequired(i) and landing(i)== none  then
438                                      begin
439                                            landing(i):- new dump(i);
440                                            activate landing(i);
441                                            ndumps:=ndumps +1;
442                                      end
443                             end;
444                      END of setup_dumps;
445                PROCEDURE setup_a_fleet;
446                  begin
447
448                       integer i;
449                       FOR I:=1 STEP 1 UNTIL 200 do
450                          begin
451                                if truck_required(i) and fleet(i) == none then
452                                begin
453                                    fleet(i):- NEW truck(i);
454                                    activate fleet(i);
455                                    ntrucks:=ntrucks+1;
456                                end;
457
458                          end
459                END of setup_a_fleet;
460
461                PROCEDURE startup_trucks;
462                   BEGIN
463                          integer outa;
464
465                       ref(truck) t,t1;
466                       t:- garage.first;
467                       while t =/= none do
468                          begin
469                              t1:-t.suc;
470                              if truck_required(t.trucknum) then
471                                 begin
472                                    activate t;
473                                    t.out;
474                                 end;
475                               t:-t1;
476                          end;
477                              inspect log do begin outimage;setpos(50);
478                                 outtext("TRUCKS RELEASED FROM REPAIRSHOP");outimage;
479                                 outimage;end;
480                       while not repairshop.empty do
481                          begin
482                              t:-repairshop.first;
483                              if t.repairtime > 600
484                                 then
485                                 begin
486                                          t.repairtime:=t.repairtime-600 ;
```

```
                                        wrdat(t.trucknum,9,600);
                                    end
                                else begin
                                        activate t   ;
                                        t.out;
                                        log.outint(t.trucknum,4);
                                        outa:=outa + 1;
                                        wrdat(t.trucknum,9,t.repairtime);
                                end;
                                log.outimage;
                        end;
                        outtext("  TRUCKS RELEASED FROM REPAIRSHOP");
                        outint(outa,4); outimage;
                        outa:=0;
                END of startup_trucks;


                    STARTDAY;
                    setup_dumps;
                    setup_a_fleet;
                    startup_trucks;
                    outint(ndumps,3);outtext("DUMPS and ");outint(ntrucks,3);
                    outtext("TRUCKS created "); outimage;
                    log.eject(1);
                    log.setpos(50);
                    log.outtext("SIMULATION DESPATCH AND ARRIVAL LOG");
                    log.outimage;
                    log.eject(5);

                END of SETUP;


                PROCEDURE END_DAY;
                   begin


                        ref(truck) t;
                        integer i,cpt,ptr,count;
                        count:=0;
                        for i :=1 step 1 until 200 do
                           if truck_required(i) then
                           begin
                                DO
!****************************!
!NING 147 LINE 530:  THIS NON-SIMULA FEATURE WILL BE WITHDRAWN - PLEASE CHANGE IT ...
                                begin
                                    getassignment(i,cpt,ptr);
                                    if cpt > 0 then
                                       begin
                                          pushreturn(cpt);
                                           count:=count+1
                                       end;
                                end
                                UNTIL cpt = 0 ;
                                wrday(i);
                                clrdayvars(i);
                           end;
                           cleartripfile;
                           outtext("    Trips not completed --");
                           outint(count + sam.notime ,5);outimage;
                           outtext("    Trucks in garage --");
                           outint(repairshop.cardinal,5);outimage;
                        sam.notime:= 0;
                        outtext("Todaystrips = ");outint(todaystrips,4);
                        outimage;
                         for i:=1 step 1 until list.pointerb -1  do
                            begin
                                outint(list.brkdnlst(1,i),3);
                                 outint(list.brkdnlst(2,i),5);
                                  outint(list.brkdnlst(3,i),6);
                                outint(list.brkdnlst(4,i),3);outimage;
                            end;
                          for i:=1 step 1 until list.pointern -1  do
                            begin
                                outint(list.notmlst(1,i),3);
                                 outint(list.notmlst(2,i),5);
                                 outimage;
                            end;
                             inspect list do outlists;
                             list.pointerb:=1;
                             list.pointern:=1;
                END of END_DAY;

                procedure dispose;
                begin
                        outtext("          DO you want the runlog printed ?");
                        outimage;
                        inimage;
                        if inchar = 'y' then
                           begin
                              csf("@free   runlog");
                              csf("@sym    runlog,,csclp");
                           end
                        else
                           csf("@free   runlog");
                end;

            INTEGER ndumps,ntrucks,i,todaystrips;
            REF(dump)ARRAY landing(1:100); !WARNING 100 DUMPS MAX;
            REF(truck)ARRAY fleet(1:200);
            REF(despatcher) sam;
            REF(reporter) printout;
            REF(head) dumps_active,unloadque,wait_despatcher,garage;
            REF(unloader) wagner;

            ndumps:=0;ntrucks:=0;
            ANOTHERDAY:=true;


            wait_despatcher:-new head;
            unloadque:- NEW head;
            wagner:- NEW unloader;
            sam:- new despatcher;
            garage:-NEW HEAD;
            printout:-new reporter;

            ACTIVATE wagner;

               outtext(hdg);outimage;
```

```
605
606                 WHILE ANOTHERDAY DO
607                 begin
608                       TODAYSTRIPS:=0;
609                        setup;
610                       ACTIVATE sam;
611                       ACTIVATE printout;
612                        hold( 1600);
613                           writehistory;
614                        end_day;
615                 end;
616
617               cptregstr.close;
618               daylog.close;
619               log.close;
620               tripfile.close;
621                taskinput.close;
622               tripstat.close;
623               if print then dispose;
624               outtext("runseq was  ");outtext(runseq);outimage;
625               outtext(" end at ");outfix(time,3,9);outimage;
626         END of chipmill;
627           REF(chipmill) a;
628           a:-new chipmill;
629     end

WARNING 120 LINE 25:   NON-SIMULA CONSTRUCTION - THE PROGRAM IS  N O T  PORTABLE
WARNING 120 LINE 116:  NON-SIMULA CONSTRUCTION - THE PROGRAM IS  N O T  PORTABLE
...NOTE 132 LINE 252:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRUCK
...NOTE 132 LINE 291:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRUCK
...NOTE 132 LINE 358:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRUCK
...NOTE 132 LINE 466:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRUCK
...NOTE 132 LINE 469:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRUCK
...NOTE 132 LINE 482:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRUCK
...NOTE 123 LINE 551:  UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE
...NOTE 123 LINE 558:  UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE
WARNING 120 LINE 530:  NON-SIMULA CONSTRUCTION - THE PROGRAM IS  N O T  PORTABLE

END SIMULA 3R8A. 629 LINES, NO ERRORS.
```

APPENDIX E

E.1 SAMPLE SCREEN OUTPUT MONITORING ONE DAY

Function : provides experimenter with dynamic display of system activity

E.2 SAMPLE LOGFILE FOR ONE DAY

Function : These data are written to the main Logfile produced by each run and provide the experimenter with an summary of system behaviour including breakdowns , trips not completed and trucks used .

E.3 SAMPLE DESPATCH AND ARRIVAL LOG

Function : These data include , in a highly coded form , all significant system events and are written to the Logfile every simulated day .

E.4 SAMPLE TRIP RECORD FILE ENTRY

Function : A single line entry is written to a trip Logfile at the completion of every roundtrip . The various columns represent the individual trip elements such as drive out , queue at landing , load , drive in , queue to unload , unload and refuel . The file is used primarily for system validation in the simulation system , but the potential for use as a direct input to both an accounting system , and a system performance monitoring program is evident .

E.1 SAMPLE SCREEN OUTPUT MONITERING ONE DAY

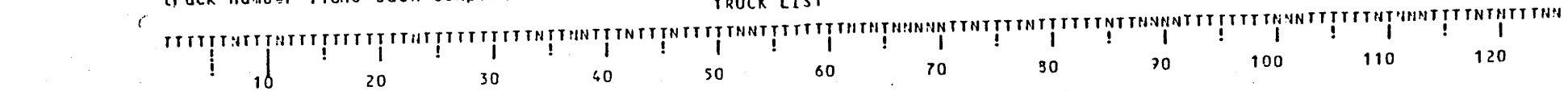
Sim4.1 55 123827

```
HOW MANY TRUCKS ?
   55
********************  SIMULATION RUN ON     12 27 82    AT     12  38  54

             What run sequence number for this run ?--
     RUN SEQUENCE NUMBER FOR THIS RUN IS --ccc05
           ----->STD FILE OPENED FOR RUN MESSAGES                    0
8589934592----->FILE OPENED FOR TRIP STATISTICS  tripzzccc05 -30064639999
8589934592----->FILE OPENED FOR DAY TOTALS ----- daylogccc05 -30064639999
           ----->TRIPFILE INITILIZED

    FILE for task input ?
           ----->COMPARTMENT MASTER OPENED                   8589934592
********************  SIMULATION RUN ON     12 27 82    AT     12  38  54
    1
 trips cleared
 actual list length 116
 RETURNS from killovertime --
pushreturn   1 trips decoded --written 115
  TRUCKS RELEASED FROM REPAIRSHOP    0
 24DUMPS and  55TRUCKS created
tttttttttttttttttttttttttttttttttttttttttttttttnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
    0ssssssssssssssssssssssssssssssssssssssssssss*******************************************************
  100ssDsssssssDssssssDsssssssssssssDDsssssssssssssssssss*******************************************************
  200dD0DdDDDDDDDDdDddDDDLdLDLddddddDdDdLDLdDDdDDLddddDddDD*******************************************************
  300ddddddDdddddddddddddddddddDDUDdddMdMdddDdddddddDDDdDDdd*******************************************************
  400Dddddddddddd Rdfd0DdddddDDDdDdDLdDDdLdDDdDDddDddDDDDdddDdD*******************************************************
  500dDddDddddDLddfDddRddRdDdDdMdddddDddddddddDdddDddddUddddL*******************************************************
  600DdDDdddDDdddDfdddDDUdUddddfdDdfddDdddDdddDdDddddfLLddd*******************************************************
BREAKDOWN 22    10   170
                     REPAIR DONE Truck  22
  700dfddffdMddffdfdfUddddfdffffdddfMddfffdfffddfffddffddUfd*******************************************************
BREAKDOWN 47   695    73
BREAKDOWN 52   257    69
  800ffffffffRfffdffffffdffdffffffdffffffffRfffffffffBffffBfff*******************************************************
  900fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
 1000fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
 1100fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
 1200fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
 1300fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
 1400fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
 1500fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
          ----->TRIPFILE INITILIZED
   Trips not completed --   0
   Trucks in garage --      2
Todaystrips = 115
 22   10   610  0
 47  696   727  1
 52  257-  731  1
   2
 trips cleared
 actual list length 140
 RETURNS from killovertime --   5
pushreturn   1pushreturn   2pushreturn   3pushreturn   4pushreturn   5 trips decoded --written  135
  TRUCKS RELEASED FROM REPAIRSHOP    2
 25DUMPS and  55TRUCKS created
 1600fffffffffffffffffffffffffffffffffffffffffffffffBffffBfff*******************************************************
```

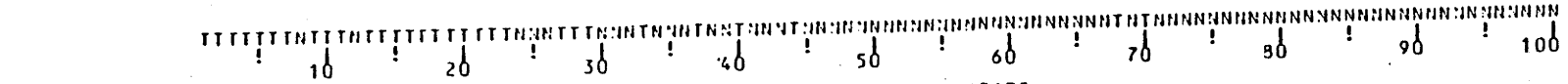
E.2 SAMPLE LOGFILE FOR ONE DAY

```
DAY NUM    2
********** SIMULATION RUN ON     12 21 32   AT    00 11 03

truck number   42no such compartment  324**********
truck number   42no such compartment  324**********
truck number   42no such compartment  324**********
truck number  110no such compartment  324**********
truck number  110no such compartment  324**********
```

TRUCK LIST



DUMPS REQUIRED



DUMP STATS

ID	1	2	3	4	5	6	7	9	10	11	13	14	15	16	17	18	19	20	21	22	23	27	28	29	33	37	40	44	63	70
DUMP	14	19	3	20	1	24	12	16	15	6	15	5	3	11	6	12	7	17	9	19	3	32	8	1	8	13	1	8	1	1

TRUCKS RELEASED FROM REPAIRSHOP

BREAKDOWNS FOR DAY

21	107	1277	0
22	10	1283	0
48	85	1291	0
75	62	1376	0
117	124	1406	0
80	429	1422	1
43	150	1541	0

TRIPS PUSHED BACK

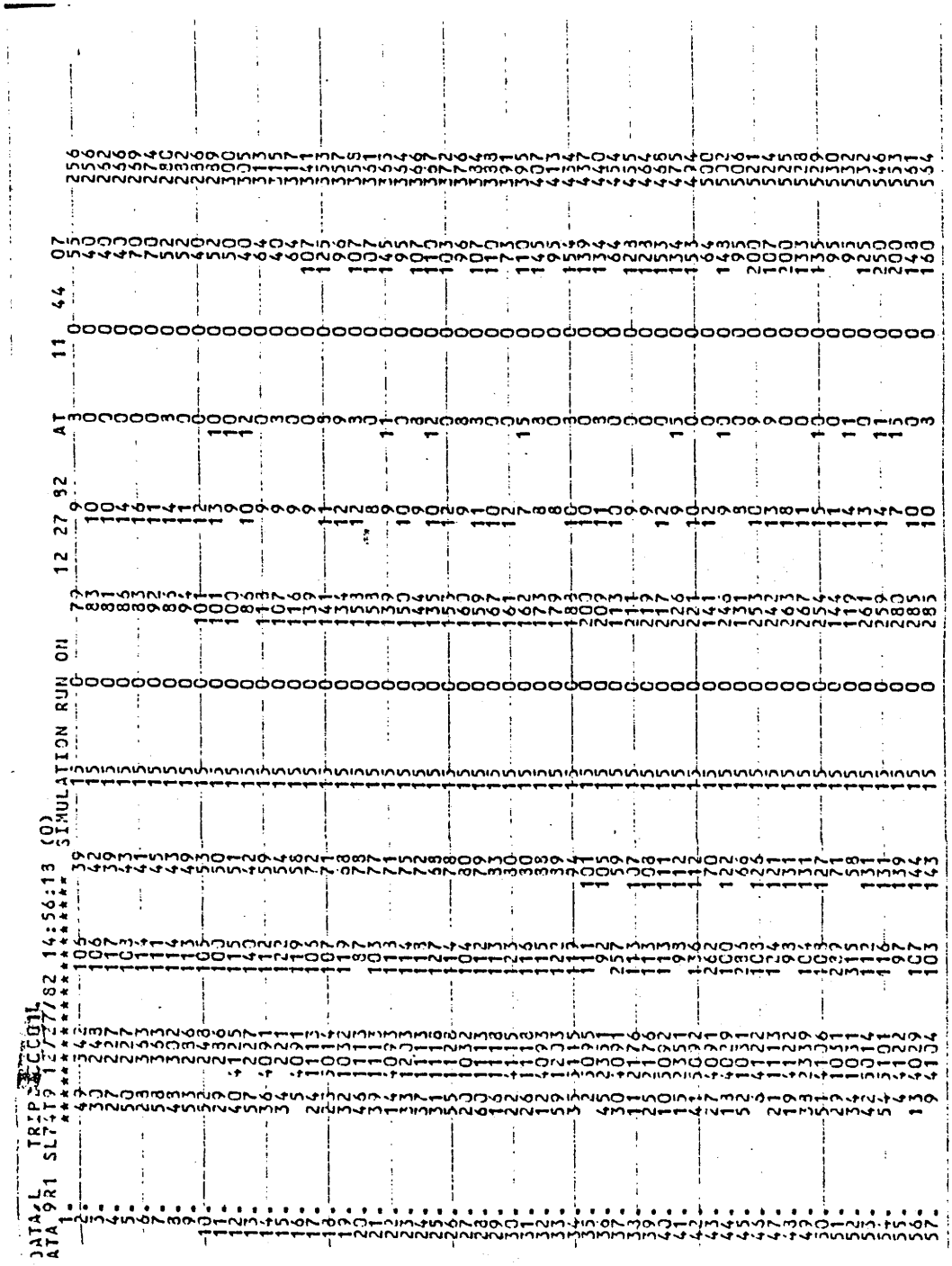
50	383
97	372
95	371
43	259

SIMULATION DESPATCH AND ARRIVAL LOG

```
TRK  72 C     3 FIN   105 TRK   3 C   130 TIME  105 TRK   6 C   180 TIME  105 TRK 103 C     0 FIN   105 TRK  22 C  1129 TIME  105
TRK  24 C  1203 TIME  105 TRK  74 C  1031 TIME  105 TRK  65 C  5103 TIME  106 TRK  25 C  1118 TIME  106 TRK   3 C  4116 TIME  106
TRK  63 C  1203 TIME  106 TRK  23 C   302 TIME  106 TRK  49 C   227 TIME  106 TRK  33 C   302 TIME  106 TRK  75 C  2179 TIME  106
TRK  98 C  5095 TIME  106 TRK 103 C  5109 TIME  106 TRK  59 C  2339 TIME  107 TRK  15 C  5115 TIME  107 TRK  30 C  2176 TIME  107
TRK  13 C  5115 TIME  107 TRK  61 C  2339 TIME  107 TRK   1 C  4106 TIME  107 TRK 123 C     0 FIN   107 TRK   2 C  4106 TIME  107
TRK 116 C  5103 TIME  107 TRK  32 C  5098 TIME  107 TRK  76 C     0 FIN   107 TRK  55 C  5022 TIME  107 TRK  20 C  2324 TIME  108
TRK  26 C  4029 TIME  103 TRK  60 C   248 TIME  108 TRK  16 C  5115 TIME  108 TRK 104 C  5101 TIME  108 TRK  32 C  4029 TIME  108
TRK  96 C  5110 TIME  108 TRK  93 C   964 TIME  108 TRK  39 C  5103 TIME  108 TRK  45 C  1032 TIME  108 TRK  54 C  4122 TIME  109
TRK  71 C   363 TIME  109 TRK 119 C  5113 TIME  109 TRK  17 C  1113 TIME  109 TRK 122 C  5113 TIME  109 TRK  92 C  5103 TIME  109
TRK  84 C  5014 TIME  109 TRK  53 C  4104 TIME  109 TRK  27 C     0 FIN   109 TRK  29 C  1113 TIME  109 TRK  79 C     0 FIN   109
TRK  58 C  1113 TIME  109 TRK  40 C  1113 TIME  110 TRK  35 C  1113 TIME  110 TRK  77 C  1113 TIME  110 TRK  33 C   286 TIME  110
TRK  21 C  4091 TIME  110 TRK  12 C  2178 TIME  110 TRK  14 C  1031 TIME  110 TRK  51 C  4116 TIME  110 TRK  87 C  4125 TIME  110
TRK   4 C  2351 TIME  110 TRK  43 C  1032 TIME  111 TRK  99 C  5095 TIME  111 TRK  31 C  2176 TIME  111 TRK 107 C  5102 TIME  111
TRK 117 C  5095 TIME  111 TRK  19 C  5014 TIME  111 TRK 115 C  5103 TIME  111 TRK 114 C  5095 TIME  111 TRK  74 C     0 FIN   111
TRK  56 C  1113 TIME  111 TRK  18 C     0 FIN   111 TRK  80 C  2351 TIME  111 TRK  44 C  4091 TIME  112 TRK   9 C  1113 TIME  112
TRK  31 C  4029 TIME  112 TRK 106 C  5101 TIME  112 TRK  43 C  4122 TIME  112 TRK  97 C  5105 TIME  112 TRK   5 C  4029 TIME  112
TRK  10 C  1113 TIME  112 TRK  36 C   132 TIME  112 TRK  41 C   176 TIME  112 TRK  57 C   302 TIME  113 TRK  86 C  5095 TIME  113
TRK  95 C  5105 TIME  113 TRK 105 C  5101 TIME  113 TRK 110 C  4111 TIME  113 TRK 121 C  4111 TIME  113 TRK  47 C  4125 TIME  113
TRK  60 C   243 TIME  127 TRK   8 C   221 TIME  198 TRK   6 C   180 TIME  207 TRK  36 C   192 TIME  211 TRK  49 C   180 TIME  252
TRK  28 C   302 TIME  219 TRK  83 C   286 TIME  225 TRK  33 C   302 TIME  239 TRK  71 C     0 FIN   242 TRK  57 C   180 TIME  271
TRK  93 C   964 TIME  255 TRK  87 C  4125 TIME  255 TRK  41 C   176 TIME  264 TRK  60 C   248 TIME  270 TRK  44 C  4091 TIME  297
TRK  74 C     0 FIN   277 TRK  14 C  1031 TIME  290 TRK  45 C  1032 TIME  292 TRK  22 C  1128 TIME  291 TRK  24 C  1203 TIME  315
TRK  63 C  1203 TIME  306 TPK  84 C  5014 TIME  308 TRK   8 C   221 TIME  311 TRK  92 C  1203 TIME  313 TRK   6 C   180 TIME  331
TRK  35 C  1113 TIME  320 TRK  25 C   964 TIME  320 TRK  19 C     0 FIN   325 TRK  35 C   192 TIME  326 TRK  56 C  1113 TIME  346
TRK  17 C  1113 TIME  333 TRK  29 C  1113 TIME  338 TRK  40 C  1113 TIME  343 TRK  47 C   227 TIME  345 TRK  37 C   286 TIME  359
TRK  58 C     0 FIN   348 TRK  57 C   180 TIME  349 TRK  28 C   302 TIME  350 TRK  77 C  1031 TIME  355 TRK  33 C     0 FIN   381
TRK  60 C   243 TIME  362 TRK   9 C  1113 TIME  362 TRK  10 C  1113 TIME  369 TRK  43 C  1032 TIME  376 TRK  12 C  2178 TIME  398
TRK  21 C  4091 TIME  384 TRK  87 C  4125 TIME  335 TRK 103 C  5109 TIME  390 TRK  15 C     0 FIN   396 TRK  98 C  5095 TIME  418
TRK  59 C  2339 TIME  403 TRK  30 C     0 FIN   403 TRK 107 C     0 FIN   410 TRK  65 C  1128 TIME  412 TRK   3 C     0 FIN   430
TRK  61 C  2337 TIME  419 TRK  31 C     0 FIN   423 TRK  13 C     0 FIN   425 TRK  99 C  5095 TIME  427 TRK  93 C   964 TIME  442
TRK  75 C  2178 TIME  433 TRK 116 C  5103 TIME  438 TRK   2 C  4106 TIME  440 TRK  16 C     0 FIN   440 TRK   6 C   180 TIME  454
TRK  51 C  4116 TIME  446 TRK  20 C     0 FIN   446 TRK  41 C   176 TIME  450 TRK 114 C  5095 TIME  452 TRK   8 C     0 FIN   475
TRK  55 C  5092 TIME  462 TRK   1 C     0 FIN   466 TRK  36 C   192 TIME  468 TRK  44 C  4091 TIME  468 TRK  39 C     0 FIN   494
TRK  36 C     0 FIN   476 TRK   4 C     0 FIN   478 TRK  92 C     0 FIN   482 TRK 115 C     0 FIN   491 TRK  60 C   243 TIME  509
TRK   5 C     0 FIN   495 TRK  26 C  4122 TIME  497 TRK  32 C  5092 TIME  498 TRK  57 C     0 FIN   499 TRK  49 C     0 FIN   523
TRK  22 C  1123 TIME  510 TRK  14 C  1031 TIME  513 TRK  31 C  4029 TIME  515 TRK 104 C  5101 TIME  516 TRK  26 C  5110 TIME  534
TRK  54 C  4122 TIME  527 TRK 117 C  5095 TIME  530 TRK  33 C   286 TIME  531 TRK  25 C  1118 TIME  533 TRK  28 C   302 TIME  554
TRK  45 C     0 FIN   541 TRK 105 C  5101 TIME  546 TRK  24 C  5115 TIME  549 TRK  92 C     0 FIN   552 TRK 106 C     0 FIN   564
TRK  84 C     0 FIN   557 TRK  63 C  1031 TIME  557 TRK 122 C     0 FIN   560 TRK  47 C  4122 TIME  562 TRK 119 C     0 FIN
TRK  37 C  4125 TIME  566 TRK  50 C  4104 NO TIME  TRK  35 C     0 FIN   575 TRK  97 C  5105 NO TIME  TRK  95 C  5105 NO TIME
TRK  77 C  1113 TIME  582 TRK  29 C     0 FIN   538 TRK  17 C  1113 TIME  591 TRK  56 C  1118 TIME  591 TRK  21 C     0 FIN   594
TRK   6 C     0 FIN   594 TRK  40 C  1113 TIME  600 TRK   9 C  1113 TIME  604 TRK  48 C  1032 TIME  607 TRK  93 C   964 TIME  617
TRK  36 C   192 TIME  613 TRK  65 C     0 FIN   620 TRK  60 C     0 FIN   621 TRK  10 C     0 FIN   623 TRK  41 C     0 FIN   624
TRK  44 C     0 FIN   632 TRK 110 C     0 FIN   637 TRK 121 C     0 FIN   639 TRK  12 C     0 FIN   645 TRK  28 C     0 FIN   649
TRK  93 C     0 FIN   657 TRK  94 C     0 FIN   681 TRK 103 C     0 FIN   682 TRK  14 C     0 FIN   635 TRK  59 C     0 FIN   686
TRK  22 C     0 FIN   683 TRK  43 C  4122 NO TIME  TRK  75 C     0 FIN   703 TRK  87 C     0 FIN   706 TRK  99 C     0 FIN   707
TRK  61 C     0 FIN   712 TRK  36 C     0 FIN   723 TRK  25 C     0 FIN   727 TRK  63 C     0 FIN   727 TRK 114 C     0 FIN   745
TRK 116 C     0 FIN   748 TRK   2 C     0 FIN   753 TRK  51 C     0 FIN   756 TRK  55 C     0 FIN   732 TRK  48 C     0 FIN   786
TRK  77 C     0 FIN   783 TRK  93 C     0 FIN   738 TRK  56 C     0 FIN   791 TRK  40 C     0 FIN   803 TRK  17 C     0 FIN   806
TRK 117 C     0 FIN   311 TRK   9 C     0 FIN   814 TRK  32 C     0 FIN   817 TRK  24 C     0 FIN   843 TRK  31 C     0 FIN   851
TRK  26 C     0 FIN   864 TRK 104 C     0 FIN   869 TRK  96 C     0 FIN   875 TRK  54 C     0 FIN   894 TRK 105 C     0 FIN   900
TRK  47 C     0 FIN   923
```

E.3 SAMPLE DESPATCH AND ARRIVAL LOG

E.4 SAMPLE TRIP RECORD FILE ENTRY



APPENDIX F

F.1 RESULTS OF ROUNDTRIP STUDY

The table provides summary data for roundtrip time data gathered in the July study month.

F.2 DISTRIBUTION OF PARAMETERS FROM MAXIMUM LIKELIHOOD ESTIMATION

The table provides summary statistical data from the maximum likelihood estimation of three parameter lognormal continuous distributions from roundtrip data.

RESULTS OF ROUNDTRIP STUDY

COMP-ART MENT	DISTANCE (KMS)	N OF TRIPS	N OF TRIPS	MEDIAN TRIP TIME	25% QUARTILES	75%	COMP-ART MENT	DISTANCE (KMS)	N OF TRIPS	N OF TRIPS	MEDIAN TRIP TIME	25% QUARTILE	75%
16	35	15	3	120	0	0	4048	43	7	4	114	87	191
111	37	4	4	167	0	0	4065	140	2	0	350	0	0
176	26	28	20	191	159	239	4072	25	51	35	158	123	197
180	23	130	84	122	104	154	4080	26	12	0	187	0	0
192	23	68	48	127	102	156	4088	200	54	21	418	402	449
221	40	44	31	132	124	149	4089	200	5	0	450	0	0
227	40	73	51	138	128	164	4091	64	134	92	191	177	204
229	40	42	31	130	119	139	4093	145	4	0	260	0	0
248	40	110	88	104	95	127	4104	160	19	4	480	465	489
286	52	93	65	148	137	162	4106	135	44	11	370	338	435
302	55	228	156	148	134	163	4107	140	15	0	350	0	0
342	55	41	23	193	164	237	4109	92	8	0	210	0	0
363	70	4	0	0	0	0	4110	42	43	21	296	237	316
964	74	103	71	187	176	202	4111	250	21	0	600	0	0
1031	95	141	91	203	197	214	4115	73	34	17	268	225	287
1032	96	138	89	218	209	237	4116	145	8	0	370	0	0
1113	117	271	157	249	236	275	4118	120	4	0	260	0	0
1114	113	59	33	271	239	285	4119	88	27	15	258	247	274
1115	115	25	16	246	228	317	4122	200	28	11	468	381	498
1118	112	90	59	243	221	262	4125	50	12	9	166	144	213
1125	110	24	15	223	214	242	5014	125	88	54	244	232	251
1128	105	114	72	206	201	221	5017	129	39	10	436	425	448
1201	90	4	2	210	0	0	5060	220	4	0	450	0	0
1203	95	103	59	223	211	239	5062	130	5	4	265	262	326
1800	40	2	1	120	0	0	5066	130	3	1	360	0	0
2176	127	62	26	346	306	357	5082	140	7	1	392	0	0
2177	127	45	23	313	289	374	5085	90	3	2	238	0	0
2178	128	30	12	316	302	322	5092	153	24	10	387	340	439
2207	136	19	5	409	372	438	5095	139	90	38	345	329	371
2316	137	53	26	384	365	394	5098	159	67	32	409	361	438
2324	136	19	7	361	340	409	5099	235	10	3	482	0	0
2338	132	99	45	378	345	428	5101	250	70	22	434	413	468
2339	133	75	30	349	330	366	5103	126	97	42	366	317	394
2351	134	21	6	376	368	425	5104	153	4	3	360	0	0
3023	144	15	6	254	210	370	5105	235	63	22	485	446	507
3040	160	37	18	379	346	403	5109	150	12	0	350	0	0
3042	40	32	18	166	155	175	5110	193	45	17	431	390	443
3917	260	10	0	400	0	0	5113	230	47	12	468	439	545
3918	260	14	0	400	0	0	5115	154	49	13	353	340	386
4010	145	41	14	383	342	411	5118	160	13	0	390	0	0
4029	148	93	40	399	378	419							

DISTRIBUTION PARAMETERS FROM MAXIMUM LIKELIHOOD ESTIMATION

MEDIAN (mins)	'A'	'M'	'S'	N	CHI	df
153	85.2	4.2	.28	22	2.11	2
205	175.1	3.4	.26	31	.32	2
219	194.9	3.2	.82	23	.06	2
198	119.2	4.4	.38	33	2.44	2
148	117.0	3.4	.59	44	.65	2
226	196.7	3.4	.80	21	1.70	2
196	104.2	4.5	.19	45	1.80	2
124	66.4	4.1	.63	44	2.65	2
249	232.9	2.8	.88	21	.67	2
188	159.1	3.4	.68	49	1.55	2
222	185.9	3.6	.44	33	2.14	2
140	105.6	3.5	.66	45	6.30	2
259	145.5	4.7	.25	31	1.63	2
103	80.8	3.1	.81	72	4.79	2
210	189.8	3.0	.72	31	2.81	2
155	80.9	3.6	1.01	27	.70	2
205	193.4	2.5	.67	29	1.29	2
152	109.2	3.8	.57	23	.42	2
131	90.9	3.7	.52	44	2.68	2
148	126.3	3.1	.73	39	.42	2
246	214.4	3.5	.49	24	1.19	2
133	120.6	2.5	1.11	39	3.32	2
362	243.1	4.8	.55	11	-	0
375	362.3	2.6	1.17	12	.68	1
238	179.5	4.1	.42	18	.65	1
223	198.1	3.1	.40	10	-	0
358	205.9	5.0	.27	12	-	0
196	124.9	4.3	.24	12	-	0
427	317.1	4.6	.40	13	.15	1
367	160.1	5.3	.17	15	.01	1
191	140.4	3.9	1.05	17	.08	1
350	337.5	2.6	1.35	10	.33	1
432	369.1	4.2	.80	10	.21	1
400	284.5	4.7	.30	12	-	0
115	101.9	2.6	1.06	11	-	0
264	165.8	4.6	.40	10	-	0

APPENDIX G

CLASS Dataset (Second version containing the heuristics)

Class schedule

Class pass1

Class tasklists
Class taskcell
function : linked list system for truck tripsets

Procedure printclass
Procedure getmeans
Procedure getdirection
Procedure dealouttrips
Procedure printer
function ; utilitys for the operation of the "best to worst" heuristic

Class pass2

Class alltrucks
class mytrips
class trip
class trday
function : linked list systems to hold trucks and trip sets during reallocation

Procedure instaltr
Procedure removetr
Procedure sorttimes
Procedure allwrite
Procedure getsmallest
Procedure getbiggest
Procedure tryswaps
function : utilities used in the second pass of the heruistic

Procedure killovertime
Procedure putback
function : utilities for third pass to check for long days and reallocate if possible
Procedure printer
Procedure getmeanscore
Procedure rewrite
function : utilities used in overall control of pass 2

Procedure instaltrip
function : puts trips in despatch table

Procedure updatedumplist
Procedure updatetrucklist

procedure wake
function : from earlier dataset , APPENDIX C

Procedure decodetrips
function : interprets output of heuristic reallocation and calls instaltrips to put in despatch table

Procedure startday
function : manages the operation of collecting the trips required , heuristic reallocation and preparing the despatch table each day .

Procedure checkoutput
function : daily display of units required for daily log and VDU

Procedure getassignment
function : works for despatcher by interrogating the despatch table

```
external class basic;
basic class dataset;
BEGIN


        comment**********************************************************
        *       getdump works with external master file "CPTREGSTR"
        *             to get cpt details
        **********************************************************;


 class schedule(trips,history,numtr,returns);
            integer array trips,history,returns;
            integer numtr;
BEGIN


      SIMSET class  pass1;
       begin

         HEAD class tasklists; ;

         LINK class taskcell(ar); real array ar;
         BEGIN
               real array trips(1:7);
               for i := 1 step 1 until 7 do
                  trips(i):= ar(i);
         END of taskcell;

         Procedure pushtasks;
         BEGIN
               integer k,d,l;
               real array ar(1:7);

               for k := 1 step 1 until numtr do
               begin
                    if (abs(todays(k,4)-ttmu)*2)/ttmu > abs(todays(k,5)-tkmu)
                        /tkmu then d:=1 else d:=2;
                        moretasks(d):= true;

                     for l := 1 step 1 until 7 do
                        ar(l):= todays(k,l);
                     new taskcell(ar).into(tasker(d));
               end;
         END of pushtasks;

         Procedure printclass;
         begin
               ref(tasklists) t;
               ref(taskcell)  c;
               integer i,j,k;
               for i := 1 step 1 until 2 do
               begin
                     t:- tasker(i);
                       c :- t.first;
                     for j := 1 step 1 until t.cardinal do
                     begin
                             for k := 1 step 1 until 5 do
                                   outfix(c.trips(k),2,7);
                           outimage;
                           c:-c.suc;
                     end;
               end;
         END;


         Procedure getmeans;
         BEGIN
               integer i;
               real array total(1:4);
               for i := 1 step 1 until 4 do
                     total(i):= 0;
               for i := 1 step 1 until numtr do
                  begin
                       total(1):= total (1) + todays(i,4);
                       total(2):= total (2) + todays(i,5);
                       total(3) :=total(3) + past(i,1);
                       total(4) :=total(4) + past(i,2);
                  end;

               ttmu:=total(1)/numtr;
               tkmu:=total(2)/numtr;
               ptmu:=total(3)/numtr;
               pkmu:=total(4)/numtr;

            if DIAGP1 then
               begin
                    outtext("totals");
                    for i := 1 step 1 until 4 do
                       outfix(total(i),2,10);
                    outimage;
                    outfix(tkmu,2,7);outtext("was tkmu");outimage;
               end;

         END of getmeans;
         Procedure getdirection(point1); integer point1;
         begin

                  if(past(point1,2) - pkmu) > (past(i,1) - ptmu)
                     then
                        direction := 1
                     else
                        direction := 2;
                  if not moretasks(direction)
                     then
                     begin
                          if direction = 1 then
                                direction := 2 else direction := 1;
                     end;
           end of getdir;

         Procedure dealouttrips;
         begin
               integer i,k;
               ref(tasklists) t;
               ref(taskcell) c;
```

```
                t:-tasker(direction);
                c:-t.first;
                c.out;
                for k := 1 step 1 until 7 do
                   tasks(k) := c.trips(k);
            if t.cardinal = 0 then moretasks(direction) := false;
        END of dealout;

        procedure printer;
        begin
            integer i;
            for i := 1 step 1 until numtr do
            begin
                  for j := 1 step 1 until 7 do
                     outint(todays(i,j), 7);
                  for j := 1 step 1 until 4 do
                     outint(past(i,j),8);
                  for j := 1 step 1 until 3 do
                        outint(history(i,j),6);
                  outimage;
            end;
        END;
        integer i,j,ttmu,tkmu,ptmu,pkmu,t,k,direction,point1,point2,tr;
        integer cpt,ttime,dist;
        integer array todays(1:numtr,1:7),past(1:numtr,1:4),
                     tasks(1:7),first(1:numtr,1:4);
        boolean array  moretasks(1:2);
        boolean DIAGP1;
        ref(tasklists) array tasker(1:2);

        tasker(1) :- new tasklists;
        tasker(2) :- new tasklists;

        for i := 1 step 1 until numtr do
           begin
               tr:=trips(i,1);
               gettime_dist(tr,ttime,dist);
               first(i,1):= trips(i,1);
               first(i,2):=ttime;
               first(i,3):=dist;
               first(i,4):= first(i,2)*2 + first(i,3);
               past(i,1):= history(i,1) + first(i,2);
               past(i,2):= history(i,2) + first(i,3);
               past(i,3):= history(i,3) + first(i,4);
               past(i,4):=i;
               for j := 2 step 1 until 4 do
                  begin
                     tr:=trips(i,j);
                     if tr > 0 then begin
                        gettime_dist(tr,ttime,dist);
                           todays(i,j-1):= tr;
                           todays(i,4):=todays(i,4) +ttime;
                           todays(i,5):=todays(i,5) +dist;
                        end
                  else if tr <> 0 then
                     begin
                        outtext("rogue comp"); outint(tr,3); outimage;
                     end;
                  end;
               todays(i,6):= todays(i,4) * 2 + todays(i,5);
        END of instal loop;

        sort(todays,numtr,6,6);
        sort(past,numtr,4,3);
        getmeans;
        pushtasks;
        if DIAGP1 then  printer;
        for i := 1 step 1 until numtr do
        begin
               point1:= numtr+ 1 - i;
               point2:=past(point1,4);
               getdirection(point1);
               dealouttrips;
               for j :=1 step 1 until 3 do
                  trips(point2,j+1):=tasks(j);
               for j := 1 step 1 until 3 do
                  trips(point2,j+4):=past(point1,j) + tasks(j + 3);
        end;
        sort(trips,numtr,8,7);
    END of pass1;
```

```
simset class pass2;
begin
        head class alltrucks;;
           head class mytrips;;
           link class trip;
           begin
               integer id,time,dist,score;
           end of trips;

        link class trday(i); integer i;
        begin
            procedure maketrips;
            begin
                  ref(trip) t;
                  integer j,k,cpt,ttime,dista;
                  for j := 1 step 1 until 4 do
                     if trips(i,j) <> 0 then
                     begin
                         t:- new trip;
                         cpt:=trips(i,j);
                         gettime_dist(cpt,ttime,dista);
                         t.id:=trips(i,j);
                         t.time:=ttime;
                         t.dist:=dista;
                         t.score:=t.time * 2 + t.dist;
                         todaytime:=todaytime + t.time;
                         t.into(mytr);
                     end;
            end of maketrips;

            integer cpt,myscore,myt,myk,timeleft,todaytime,myident;
            ref(mytrips) mytr;
            mytr:- new mytrips;
            maketrips;
            myt:=trips(i,5);
            myk:=trips(i,6);
            myscore:=trips(i,7);
            myident:=trips(i,8);
            timeleft:=720 - todaytime;
            if diagprint  then begin
                outtext("timeleft was");outint(timeleft,5);outtext("for");
                outint(myident,5);outimage;end;
        END of trday;

        Procedure instaltrips;
        begin
               integer i;
               for i := 1 step 1 until numtr do
                  begin
                      tdy:- new trday(i);
                      tdy.into(all);

                        if DIAGPRINT then
                            begin
                                outtext("trday.myscore"); outint(tdy.myscore,5);
                                outimage;
                            end;
               end;
        END of instal;

-------------------------------------------------------------------------------

        Procedure instaltr(td,tr); ref(trday) td;ref(trip) tr;
        begin
               td.myt:=td.myt+tr.time;
               td.myk:=td.myk +tr.dist;
               td.myscore:=td.myscore + tr.time * 2 + tr.dist;
               td.todaytime:=td.todaytime + tr.time;
                 if diagprint then begin  !*****************************;
                   outtext(" trip");outint(tr.id,4);outtext("in");
                   outtext(" truck");outint(td.myident,4);outimage;
               end;
        END;

        Procedure removetr(td,tr); ref(trday) td;ref(trip) tr;
        begin
               td.myt:=td.myt-tr.time;
               td.myk:=td.myk -tr.dist;
               td.myscore:=td.myscore - tr.time * 2 - tr.dist;
               td.todaytime:=td.todaytime - tr.time;
                if diagprint then  begin !*********************;
                   outtext(" trip");outint(tr.id,4);outtext("out");
                   outtext(" truck");outint(td.myident,4);outimage;
               end;
        END;

        Procedure sorttimes;
        begin
               ref(trday) t,temp,base;
               base:-all.first;
               while base =/= NONE do
                  begin
                      temp :- t:- base;
                      while temp =/= all.last do
                         begin
                             temp:-temp.suc;
                             if temp.todaytime < t.todaytime then t:- temp;
                         end;
                      if t =/= base then t.precede(base)
                         else base :- base.suc;
                      if diagprint then begin
                         outtext("in sort");outint(t.myident,5);outimage;
                      end;
                  end;
        END of sorttimes;

           procedure allwrite;
           begin
                  ref(trday) td;
                  ref(trip)tr;
                  td:- all.first;
                  while td =/= all.last do
                  begin
                      outtext(" trday ident in all was");outint(td.myident,3);
                      tr:-td.mytr.first;
                      while tr =/= none do
                         begin
                             outint(tr.id,4);
                             tr:-tr.suc;
                         end;
                      outtext("todaytime");outint(td.todaytime,5);
                      outimage;
```

```
                td:- td.suc;
            end;
        END of alwrite;
```

```
Procedure getsmallest(trk,trp);ref(trday) trk;ref(trip) trp;
            name trk,trp;
begin
        integer smallest;
        ref(trip) temp;
        temp:-trk.mytr.first;
        smallest:=temp.time;
        trp:-temp;
        WHILE temp =/= trk.mytr.last DO
           begin
                temp :-temp.suc;
                if temp.time < smallest and temp.id <> 0
                then
                   begin
                       trp:-temp;
                       smallest:=trp.time;
                   end;
        end;
           if DIAGprint then
              begin
                 outtext("small was");outint(trp.id,3);outint(trp.time,3);
                 outtext("for");outint(trk.myident,4);outimage;
                 outimage;
              end;
END of smallest;

PROCEDURE getbiggest(trk,trp);ref(trday) trk;ref(trip)trp;
            name trk,trp;
begin
        integer biggest;
        ref(trip) temp;

        temp:-trk.mytr.first;
        trp:-temp;
        biggest:=trp.time;

        WHILE temp =/= trk.mytr.last
        DO
           begin
                temp:-temp.suc;
                if temp.time > biggest
                   then
                   begin
                       trp:-temp;
                       biggest:=trp.time;
                   end;
           end;
           if DIAGprint then
              begin
                 outtext("biggest was");outint(trp.id,3);outint(trp.time,3);
                 outtext("for");outint(trk.myident,4);outimage;
                 outimage;
              end;
   END of biggest;

PROCEDURE tryswaps;
begin

        procedure swapit(t1,t2,tp1,tp2);ref(trday) t1,t2;
                                        ref(trip)tp1.tp2;
```

```
387                 begin
388                     ref(trip) temp;
389
390                     tp2.follow(tp1);
391                     installtr(t1,tp2);
392                     removetr(t2,tp2);
393                     tp1.into(t2.mytr);
394                     installtr(t2,tp1);
395                     removetr(t1,tp1);
396                 end of swapit;
397
398                 integer i;
399                 ref(trday) t1,t2;
400                 ref(trip) tp1,tp2;
401
402                 t1:- al1.first;
403                 t2:-al1.last;
404
405                 for i := 1 step 1 until 5 do
406                    begin
407                        getsmallest(t1,tp1);
408                        getbiggest(t2,tp2);
409                        if DIAGprint then
410                           begin
411                               outtext("biggest was");outint(tp1.id,3);outint(tp1.time,3
412                               outtext("for");outint(t1.myident,4);outimage;
413                               outimage;
414                           end;
415                        if tp1.time < tp2.time then
416                        swapit(t1,t2,tp1,tp2);
417                        t1:-t1.suc;
418                        t2:-t2.pred;
419                    end;
420             END of try;
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
```

```
423             Procedure Killovertime;
424             begin
425
426                     procedure putback(tr); ref(trip) tr;
427                     begin
428                             if DIAGPRINT then    begin !************************;
429                                 outtext("LOST one in killovertime");
430                                 outint(tr.id,4);outtext(" time");outint(tr.time,5);outimage;
431                             end;
432                             rptr:=rptr + 1;
433                             returns(rptr):=tr.id;
434                     end;
435
436                     ref(trday) base,td1,td2;
437                     ref(trip) tr,temp;
438                     integer overtime;
439
440                     base:- al1.last;
441                     while base.todaytime > 720 and base =/= al1.first do
442                        base:-base.pred;
443
444                     td1 :- al1.last;
445                                     begin
446                     end;
447                       overtime :=  td1.todaytime- 720;
448                     while overtime > 0 do
449                        begin
450                            if diagprint then  begin !************************;
451                              outtext("overtime");outint(overtime,5);outint(td1.myident,4);
452                                outimage;
453                            end;!************************;
454                            tr:- td1.mytr.first.suc;
455                            temp :- tr.suc;
456                            while temp =/= none do
457                               begin
458                                   if overtime-temp.time < 0 and temp.time < tr.time
459                                      then
460                                         tr:-temp;
461                                      temp:-temp.suc;
462                               end;
463                            tr.out;
464                            removetr(td1,tr);
465
466                               td2:-al1.first;
467                               while td1.todaytime > td2.todaytime and td2 =/= al1.last do
468                                   td2:-td2.suc;
469                               if td2 =/= td1 then  td1.precede(td2);
470                         if diagprint then  begin !************************;
471                               outtext("td1");outint(td1.myident,3);outtext("pred");
472                               outint(td2.myident,3);outtext("td1.todaytime");
473                               outint(td1.todaytime,5);outtext("td2.todaytime");
474                               outint(td2.todaytime,5);outimage;
475                           end ; !************************;
476                           td2:-al1.first;
477                           if diagprint then
478                              begin
479                           outtext("td2 todaytime");outint(td2.todaytime,5);outimage;
480                           end;
481                           if tr.time + td2.todaytime > 720
482                              then putback(tr)
```

```
                    else
                       begin
                            tr.into(td2.mytr);
                            installtr(td2,tr);
                            td1:-td2.suc;
                            while td2.todaytime > td1.todaytime
                                  and td1 =/= al1.last do
                                  td1:-td1.suc;
                            td2.precede(td1);
                 if diagprint then  begin !************************;
                       outtext("td2");outint(td2.myident,3);outtext("pred");
                       outint(td1.myident,3);outtext("td2.todaytime");
                       outint(td2.todaytime,5);outtext("td1.todaytime");
                       outint(td1.todaytime,5);outimage;
                    end ; !*****************************;
                       end;
                 td1:-al1.last;
                    overtime:=td1.todaytime-720;
              end;
        END of proc overtime;

           procedure printer;
           begin
               integer i,j;
               for i := 1 step 1 until plan_ntrk do
                  begin
                     for j := 1 step 1 until 8 do
                     outint(trips(i,j),7);
                     outimage;
                  end;
           end;

           procedure getmeanscore;
           begin
                  integer i,total;
                  for i := 1 step 1 until numtr do
                     total := total + trips(i,7);
                  meanscore:=total / numtr;
           end;

           Procedure rewrite;
           begin
                  integer ident,i;
                  ref(trip)  t;
                  tdy :- al1.first;
                  while tdy =/= none do
                  begin
                         inspect tdy do
                         begin
                                ident:=myident;
                                t:-mytr.first;
                                for i := 1 step 1 until 4 do trips(ident,i):= 0;
                                i:=1;
                                while t =/=,none do
                                begin
                                       trips(ident,i) := t.id;
                                       t:-t.suc;
                                       i :=i + 1;
                                end;
                                trips(ident,5):=myt;

                                trips(ident,6):=myk;
                                trips(ident,7):=myscore;
                                trips(ident,8):=myident;
                         end of inspect;
                         tdy :- tdy.suc;
                  end;
           END of proc rewrite;

           integer i,meanscore,rptr;
           boolean DIAGPRINT;
           ref(alltrucks) al1;
           ref(trday) tdy;
           al1:- new alltrucks;
           if diagprint then printer;
           installtrips;
           getmeanscore;
           if diagprint then
              begin
                 outtext(" MEANSCORE"); outint(meanscore,5);outimage;
              end;
           tryswaps;
           if diagprint then begin rewrite;printer;end;
           sorttimes;
           if diagprint then  al1write; !*************************;
           killovertime;
           if diagprint then  al1write; !*************************;
            rewrite;
           outtext(" RETURNS from killovertime --");
           outint(rptr,4);outimage;
        END of pass2;

           procedure sort(input,len,wid,key);
                       integer array input;
                       integer len,wid,key;

           begin
                  procedure test(i,inc); integer i,inc;
                  begin
                         integer l,temp;
                         real last,thisone;
                         l:=i-inc;
                         if l >= 1 then
                            begin
                                 last:=base(l,1);
                                 thisone:=base(i,1);
                                 if last > thisone then
                                    begin
                                        base(l,1) :=thisone;
                                        base(i,1) := last;
                                        temp := base(l,2);
                                        base(l,2):=base(i,2);
                                        base(i,2):=temp;
                                        test(l,inc);
                                    end;
                            end;
                     end of test;

                integer i,j,k,l,max,inc;
                real array base(1:200,1:2),temp(1:200,1:8);
```

```
                max:=len;
                i:=1;
                for i:=1 step 1 until max do
                begin
                     base(i,1):=input(i,key);
                     base(i,2):=i;
                end;

                for inc:=entier(max/4) step -1 until 1 do
                   for j:=1 step 1 until inc do
                      begin
                          i:=j + inc;
                          while i <= max do
                             begin
                                 test(i,inc);
                                 i:=i + inc;
                             end;
                      end of sort phase;

                for i:=1 step 1 until max do
                   for j:=1 step 1 until wid do
                      temp(i,j):=input(base(i,2),j);

                for i:=1 step 1 until max do
                   for j:=1 step 1 until wid do
                      input(i,j):=temp(i,j);
          end of proc sort;

          ref(pass2) p2;
          ref(pass1) p1;
          p1:-new pass1;
          p2:- new pass2;
END of schedule;
```

```
          comment*****************************************************
          *         instaltrip work with internal scratch file "tripfile"
          *                    to build up one days work assignment for
          *                     the despatcher
          *               store provides the actual file interface
          *
          *               updatedumplist marks each active dump in the
          *                    master array "dumplist"
          *****************************************************;

     PROCEDURE instaltrip(trnum,cpt); integer trnum,cpt;
               begin

          PROCEDURE logfile3;
                begin
                     log.outtext("truck number");
                     log.outint(trnum,4);
                     log.outtext("no such compartment");
                     log.outint(cpt,5);
                     log.outtext("**********");log.outimage;
                end;


          PROCEDURE store(truckid,cptid,dumppointer);
                    integer truckid,cptid,dumppointer;
          BEGIN
                integer cptsasg,freeloc,cpt;
                text trline;

                cpt:=cptid;
                inspect tripfile do
                   begin
                        locate(truckid);
                        inimage;
                        trline:- image;
                        cptsasg:=trline.sub(1,2).getint;
                        if cptsasg < 12 then
                           begin
                                freeloc:=trline.sub(3,3).getint;
                                trline.sub(freeloc,5).putint(cptid);
                                trline.sub(freeloc+5,5).putint(dumppointer);
                                trline.sub(1,2).putint(cptsasg+1);
                                trline.sub(3,3).putint(freeloc+10);
                                locate(truckid);
                               outtext(trline);
                                outimage;
                           end
                      else
                           begin
                               pushreturn(cpt);
                               sysout.outtext(" overflow in tripfile for truck");
                               sysout.outint(truckid,5); sysout.outimage;
                           end;
                end of inspect;
          END of store;
```

```
698          PROCEDURE update_dumplist(dumpid); name dumpid;integer dumpid:
699             begin
700                  if dumpid <= 100 then
701                     begin
702                          dumprequired(dumpid):=true;
703                          dumplist(dumpid):=dumplist(dumpid)+1;
704                     end
705                  else
706                     begin
707                          outtext(" funny dump at ");outint(line,5);
708                          outtext(" of taskinput "); outimage;
709                     end;
710          END of update_dumplist;
711
712          PROCEDURE update_trucklist;
713             BEGIN
714                  if trnum<= 200 then
715                    begin
716                          truck_required(trnum):=true;
717                          wakelist(trnum):=wake;
718                           if wakelist(trnum) > 200 then wakelist(trnum):=200;
719                              ntruck:=ntruck + 1;
720                    end
721                  else
722                     begin
723                          outtext(" funny truck at ");outint(line,5);
724                          outtext(" of taskinput "); outimage;
725                     end;
726          END of update_trucklist;
727
728          INTEGER PROCEDURE wake;
729          BEGIN
730                wake:=entier(normal(wakemu,wakesigma,wakeu));
731          END;
732
733                integer dump,dumppointer;
734                real wakemu,wakesigma;
735                wakemu:=110;wakesigma:=10;
736                     getdump(cpt,dump,dumppointer);
737                     if dump ne 99 then
738                       begin
739                            store(trnum,cpt,dumppointer);
740                            update_dumplist(dump);
741                            update_trucklist;
742                       end
743                     else
744                       begin
745                            logfile3;
746                            sysout.outtext("no such compartment");
747                            sysout.outtext("*********");
748                            sysout.outint(cpt,5);sysout.outimage;
749                       end;
750     END of instaltrip;
751
752     PROCEDURE DECODETRIPS;
753     BEGIN
754            integer i,j,count;
755            count:=0;
756            for i := 1 step 1 until plan_ntrk do
757               for i := 1 step 1 until 4 do
758
759                  if trips(i,j) <> 0 then
760                  begin
761                            instaltrip(i,trips(i,j));
762                          count:=count + 1 ;
763                     end;
764         outtext(" trips decoded --written ");outint(count,4);outimage;
765     END of Decodetrips;
```

```
766             comment ******* startday may become redundant;
767
768         PROCEDURE STARTDAY;
769          BEGIN
770                 comment*****************************************************;
771                 ***************************************************************
772
773               PROCEDURE SET_UP_SCHEDLIST;
774               BEGIN
775
776
777
778
779
780                   INTEGER trucknum,thisday,cpt,count,i,day;
781                      BOOLEAN tomorrow;
782
783                   pointer:=1;
784                   ntruck:=0;
785                   listptr:=1;
786                   line:=0;
787                      daynum:=daynum + 1;
788                      if print  then logheader;
789
790                   inspect taskinput do
791                      BEGIN
792
793                       tomorrow := false; !note thisday comes from dummy read;
794                       inimage;
795                       if not endfile then thisday := image.sub(1,2).getint;
796                       sysout.outint(thisday,5);sysout.outimage;
797                       WHILE NOT endfile and NOT tomorrow  do
798                          BEGIN
799                              day:=image.sub(1,2).getint;
800                             if DIAG2 then begin sysout.outtext(image);sysout.outimage;end;
801                                 day :=  image.sub(1,2).getint;
802                                 if day ne thisday then tomorrow := true
803                                 else
804                                    begin
805                                        cpt:=image.sub(21,5).getint;
806                                        if cpt <> 0 then
807                                           begin
808                                               listcpt(listptr):=cpt;
809                                               listptr:=listptr + 1;
810                                           end;
811                                      inimage;
812                                      line:=line + 1;
813                                    end;
814                          end;
815                           if endfile then anotherday := false;
816                       END of inspect;
817                          if retptr > 1 then
818                      for i := retptr -1     step- 1 until 1 do
819                          begin
820                           if DIAG2 then begin outtext(" readreturn   ");outint(i,3);
821                                outint(return(i),4);outimage; end;
822                              listcpt(listptr):=return(i);
823                              listptr:=listptr + 1;
824                          end;
825                              cleanupreturn;
826                          writetrips;
827                          sched:-new schedule(trips,history,plan_ntrk,schedreturn);
828                          i :=1;
829                          while schedreturn(i) <> 0 do
830                             begin
831                                 pushreturn(schedreturn(i));
832                                 schedreturn(i):=0;
833                                 i :=i + 1;
834                             end;
835                          decodetrips;
836                              if DIAG2 then  tripwriter;
837               END of setup ;
838
839
840
```

```
PROCEDURE checkoutput;
   BEGIN
      INSPECT LOG DO
         BEGIN
         setpos(50);outtext("TRUCK LIST");outimage;
          outimage;
         for i:=1 step 1 until 125 do
            if truck_required(i)then outchar('T') else outchar('N');
           outimage;
          for i:=1 step 1 until 13 do outtext("    !    |");
          outimage;
          for i:=10 step 10 until 120 do outint(i,10);
          outimage;outimage;
            setpos(50);outtext(" DUMPS REQUIRED");outimage;outimage;
            for I:=1 step 1 until 100 do
          if dumprequired(i) then outchar('T') else outchar('N');
             outimage;
             for i:=1 step 1 until 10 do outtext("    !    |");
             outimage;
             for i:=10 step 10 until 100 do outint(i,10);
          outimage;outimage;

          setpos(50);
          outtext("  DUMP STATS  ");
          outimage;outimage;
          outtext("  ID  ");
          for i:=1 step 1 until 100 do
          if dumprequired(i) then
             outint(i,3);
             outimage;
             outtext(" DUMP ");
          for i:=1 step 1 until 100 do
          if dumprequired(i) then
             outint(dumplist(i),3);
             outimage;
      END of inspect;
END of checkoutput;
```

```
integer i,pointer,line;
   set_up_schedlist;
   check_trucklist;
   if print then checkoutput;

END OF STARTDAY;
```

```
890             comment**********************************************************
891             *       getassignment works for despatcher to interogate
892             *            the master file of the days work held in
893             *              "tripfile", built up by installtrip
894             **********************************************************
895
896           PROCEDURE GETASSIGNMENT(truckid,cpt,cptptr);
897                                   name cpt,cptptr;
898                                   integer truckid,cpt,cptptr;
899           BEGIN
900                 integer loc,ident;
901                 ident:=truckid;
902                 inspect tripfile do
903                    begin
904                         locate(ident);
905                         inimage;
906                         loc:=image.sub(6,3).getint;
907                         cpt:=image.sub(loc+1,4).getint;
908                         if cpt = 0 then
909                            begin
910                                 image.sub(10,1).putint(1);
911                                                 ! flag whole record as finished
912                                                   NOTE that end of day is picked
913                                                   up as zero ctp number;
914                            end
915                         else
916                            begin
917                                image.sub(6,3).putint(loc+10);
918                                cptptr:=image.sub(loc+5,5).getint;
919                               image.sub(8,2).putint(image.sub(8,2).getint+1);
920                                                 ! update cpts done counter;
921                               image.sub(loc,1).putint(1);
922                                                 ! flag this one as done;
923                            end;
924
925                         locate(ident);
926                         outtext(image);  ! store updated record;
927                         outimage;
928                 end of inspect
929           END of getassignment;
930
```

```
932             integer array schedreturn(1:100);
933             ref(head) repairshop;
934             ref(schedule)sched;
935             boolean diag2;!set this to enable diagnostic for this class;
936             repairshop:- new head;
937              set_cpt_masterfile;
938       end of dataset;
```

```
..NOTE 132 LINE 58:   IMPLICIT ¤QUA¤, RUN TIME CHECK ON TASKCELL
..NOTE 123 LINE 59:   UNTIL-EXPRESSION MAY CAUSE REPEATED EXECUTION OF REDUNDANT CODE
..NOTE 132 LINE 64:   IMPLICIT ¤QUA¤, RUN TIME CHECK ON TASKCELL
..NOTE 132 LINE 123:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TASKCELL
..NOTE 132 LINE 290:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 296:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 300:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 311:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 315:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 319:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 323:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 332:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 332:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 335:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 337:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 342:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 347:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 347:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 348:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 359:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 359:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 361:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 363:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 366:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 371:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 376:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 376:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 377:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 402:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 403:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 417:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 418:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 440:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 442:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 444:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 454:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 455:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 461:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 466:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 468:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 476:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 487:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 490:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 499:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 527:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 132 LINE 533:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 539:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRIP
..NOTE 132 LINE 547:  IMPLICIT ¤QUA¤, RUN TIME CHECK ON TRDAY
..NOTE 164 LINE 624:  SUBSCRIPT EXPRESSION IS NOT INTEGER
..NOTE 183 LINE 334:  CALL BY NAME GENERATES RUNTIME CHECK ON ACTUAL QUALIFICATION
..NOTE 183 LINE 341:  CALL BY NAME GENERATES RUNTIME CHECK ON ACTUAL QUALIFICATION
..NOTE 183 LINE 360:  CALL BY NAME GENERATES RUNTIME CHECK ON ACTUAL QUALIFICATION
```