INFORMATION THEORETIC MODELS OF PREPROCESSORS AND DECISION AIDS

by

GLORIA HIU-LAI CHYEN

S.B., Massachusetts Institute of Technology
(1981)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF

MASTER OF SCIENCE

IN

OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1984

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
June 4, 1984

Certified by
Alexander H. Levis
Thesis Supervisor

Accepted by
Richard C. Larson
Chairman, Interdepartmental Committee

INFORMATION THEORETIC MODELS OF PREPROCESSORS AND DECISION AIDS

by

GLORIA HIU-LAI CHYEN

Submitted to the Department of Electrical Engineering and Computer Science on June 18, 1984 in partial fulfillment of the requirements for the Degree of Master of Science in Operations Research

ABSTRACT

Models of preprocessors are developed for a boundedly rational decisionmaker. These models may be used to model either internal preprocessing in the situation assessment stage or external decision aids. Several preprocessing functions are analyzed including filtering and aggregation of data. It is shown that properly designed preprocessors can reduce the workload of the decisionmaker and relax the bounded rationality constraint. Finally, a dual task is defined and the trade-off between workload and performance is analyzed. The results are shown to be in agreement with experimental observations.

Thesis Supervisor: Alexander H. Levis
Title: Senior Research Scientist

ACKNOWLEDGEMENT

It is the concern and encouragement of many people that bring my thesis to a completion. First of all, I am deeply grateful to my thesis supervisor and his wife. Alex and Ilze Levis, whose patience, guidance and continual support instilled confidence in me to sustain my thesis progress. It is really a blessing to have such an understanding advisor.

I also wish to thank Lisa Babine for her well-timed and meticulous typing skill and Art Giordiani for his beautiful drafting work.

There are so many brothers and sisters in Christ that I want to thank. Without their warmth and unspoken prayers for me, the thesis could not have been finished.

Finally, it is beyond my words to express gratitude to my family whose unfailing love and comfort strengthened my determination to finish the thesis.

This research was conducted at the Laboratory for Information and Decision Systems, at M.I.T. with support provided by the Air Force Office of Scientific Research under Contract AFOSR- 80-0229.

TABLE OF CONTENTS

LIST OF FIGURES

TO

MY

FAMILY

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The current advances in sensor technology vastly increase the volume and complexity of battlefield information. For instance, the electronic intelligent receiver/locator and the near real-time photo reconnaissance data transmission produce an extremely high unfiltered data rate for processing, storage and dissemination. In addition, the development of sophisticated weapon systems demands quick reaction capability. Therefore, the time allowed for decision response in a tactical environment is severely constrained [1]. Consequently, the designer of military decisionmaking organizations faces a great challenge: how to handle the large amount of data in a timely manner without overloading the components, human or machine, of the organization.

The decisionmaking organization of interest is one supported by a command, control and communication (C^3) system. The C^3 system is a tactical decision-support system made up of "a collection of equipment and procedures used by commanders and their staff to process information, arrive at decisions, and communicate these decisions to the appropriate organization units" [2] at the right time. Essential components of such an organization are the commanders, the human decisionmakers.

Due to the high data rate, the decisionmaker will receive many data to process and disseminate. In order to avoid overloading the decisionmakers with the frequent arrival of unfiltered data, a preprocessor is inserted between the data source and the decisionmaker to alleviate his heavy workload. In reality, the preprocessing function can be performed by a human or a machine; hence, its workload can count as being either part of the human workload or not. There may be more than one preprocessor within an organization to support the decisionmakers. If the workload of the preprocessors does not add to the human workload of, (i.e., the preprocessors function as decision aids for the decisionmakers) then the organization's performance can be greatly enhanced.

1.2 GOALS

The objective of this thesis is to develop mathematical models for various preprocessors capable of reducing the decisionmaker's high processing load. Such a purpose can be achieved by means of several mechanisms. One way is to restructure the decision strategy for processing the input according to the input's characteristics, and thus reduce the decisionmaker's workload. Another mechanism is to slow down the input rate by filtering the irrelevant data through the preprocessor. The latter mechanism saves time and processing resources; the decisionmaker need attend only to the significant data. Therefore, the system performance is enhanced. Examples and applications employing these models are also presented to illustrate the utility of the approaches.

The model of the human decisionmaker adopted in this thesis is the descriptive one developed by Boettcher [3], [4]. It is a two-stage information-theoretic model of a decisionmaking process with limited processing capacity. It differs from previous models in that it characterizes the input-output relationship in terms of the total internal processing activity rather than only its throughput behavior. The capacity limitation is modeled as a constraint on the total internal processing activity.

1.3 OUTLINE OF THESIS

This thesis has been organized into six chapters as follows. In Chapter 2, the basic tools of information theory and the partition law of information used in analyzing the various models are given. The basic deterministic decisionmaker model is then reviewed; it is followed by the development of the stochastic counterpart. The modeling of the limited processing capacity of a decisionmaker is briefly discussed. In Chapter 3, the basic model of preprocessor and its extension to include filtering are presented. The preprocessor model is connected to the decisionmaker model for system evaluation. A numerical example is provided at the end of the chapter to illustrate the utility of a preprocessor in a decisionmaking system. In Chapter 4, two generic designs of preprocessors with well-defined purpose are described to illustrate some of the possible applications. Also, guidelines for the role of a preprocessor in a decisionmaking system are developed. In Chapter 5, the question of how a preprocessor can facilitate dual-task processing by a decisionmaker is investigated by formulating two versions of a dual-task problem. Different approaches are employed to examine the effects of dual-task processing on system performance as well as individual task performance. Finally, in Chapter 6, the main points of this thesis are summarized, and some promising areas of future research are suggested.

CHAPTER 2

THE DECISIONMAKING MODEL

2.1 INFORMATION THEORETIC FRAMEWORK

Information theory, originally developed as communication theory [5],[6] is useful in the analysis of information-processing systems. In this section, two fundamental quantities in information theory are reviewed: entropy and transmission. In addition, the Partition Law of Information [7] which can be used for decomposing the total activity of an information-processing system, is discussed.

Entropy

For a variable x, which is an element of the alphabet X, with probability distribution p(x), its entropy H(x) is defined to be

$$H(x) \equiv - \sum_{x} p(x) \log p(x) \tag{2.1}$$

when the logarithm's base is two, the entropy is measured in bits. Entropy is also defined as the average information or the uncertainty of x, where information does not refer to the content of the variable x, but rather to the average amount by which knowledge of x, reduces the uncertainty about it.

Transmission

For two variables x and y, elements of the alphabets X and Y with distributions p(x) and p(y) respectively, the transmission or average mutual information between x and y, T(x:y), is defined to be

$$T(x:y) \equiv H(x) - H_y(x) \tag{2.2}$$

where $H_y(x)$ is the conditional entropy in the variable x, given full knowledge of the value of the variable y. With $p(x|y)$ as the conditional probability of x given the value of y, $H_y(x)$ is defined to be

$$H_y(x) \equiv - \sum_y p(y) \sum_x p(x|y) \log p(x|y) \tag{2.3}$$

Transmission measures the relatedness or constraint holding between two variables. It can be interpreted as the amount by which knowledge of y reduces the uncertainty in x, or vice versa, as it is a symmetric quantity in x and y.

N-Dimensional Mutual Information

McGill [8] extended the concept of mutual information to N variables. Equation (2.2) can be written as

$$T(x:y) = H(x) + H(y) - H(x,y) \tag{2.4}$$

McGill's extension to N dimensions, namely,

$$T(x_1:x_2:\ldots:x_N) \equiv \sum_{i=1} H(x_i) - H(x_1,x_2,\ldots,x_N) \tag{2.5}$$

seems quite natural. The N-dimensional mutual information measures the total constraint or interrelatedness holding among all N variables of a system. Conant [7] has pointed out that this measure may be expressed as the sum of simplier quantities. For example, a system may be decomposed into subsystems, and the N-dimensional mutual information of this system is the sum of two quantities: the transmissions of the individual subsystems and the transmission between subsystems. With N=4, this might be expressed:

$$T(x_1:x_2:x_3:x_4) = T(x_1:x_2) + T(x_3:x_4) + T(x_1,x_2:x_3,x_4) \tag{2.6}$$

Partition Law of Information

The Partition Law of Information [7] is defined for a system with n variables, say, n-1 internal variables w_1 through w_{n-1}, and an output variable y, also called w_n. The law states that

$$\sum_{i=1}^{N} H(w_i) = T(x:y) + T_y(x:w_1,w_2,\ldots,w_{n-1}) + H_x(w_1,w_2,\ldots,w_{n-1},y)$$

$$+ T(w_1:w_2:\ldots:w_{n-1}:y) \qquad (2.7)$$

and is easily derived using information theoretic identities. The sum of the marginal entropies of the n variables within the system on the left-hand side of Eq. (2.7) is defined as the total information-processing activity G of the system. Each of the quantities on the right-hand side of Eq. (2.7) has its own interpretation and will be discussed now.

The first term, T(x:y), is called the throughput of the system and is denoted by G_t. It describes the input-output relationship of the system and measures the amount by which the output of the system is related to the input. This is the quantity that a communications channel tries to maximize. The second term

$$T_y(x:w_1,w_2,\ldots,w_{n-1}) = T(x:w_1,w_2,\ldots,w_{n-1},y) - T(x:y) \qquad (2.8)$$

is called the blockage of the system, G_b. As the above expansion demonstrates, blockage may be thought of as the amount of information in the input to the system that is not included in the output. The third term, $H_x(w_1,w_2,\ldots,w_{n-1},y)$ is called the noise of the system, G_n. It represents the uncertainty that remains in the system variables when the input is completely known. Such uncertainty may come from noise generated within the system. It may also be thought of as internally-generated information, i.e., information supplied by the system to supplement the input and facilitate the decisionmaker. The last term, $T(x:w_1:w_2:\ldots:w_{n-1}:y)$ is called the coordination of the system, G_c. It is just the n-dimensional mutual information of the system, the amount by which all of the internal and output variables in the system constrain each other. It reflects all system variable interactions and can be interpreted as the coordination required among the system variables to accomplish the processing of the inputs to obtain the output. As noted before, if the system is broken up into independent subsystems, then this coordination term may be expressed as a simple sum of the coordinations of the individual subsystems. Finally, the Partition Law of Information may be abbreviated as:

$$G = G_t + G_b + G_n + G_c \tag{2.9}$$

2.2 THE BASIC DECISIONMAKING MODEL

The model of the preprocessor that is developed in this thesis will be linked with the model of decisionmaker so that its utility in alleviating the processing workload of the decisionmaker can be analyzed. In this section, a brief overview of the concepts and results of the basic decisionmaking model will be presented. The detailed description and analysis of the determinstic version of this model can be found in [3], [4]. The stochastic version of the model will be developed in detail in the next section.

The basic model of the decisionmaker is illustrated in Fig. 2.1 as a two-stage process. The decisionmaker receives an input symbol x from his environment. If the input symbols are generated every τ seconds on the average, then τ, the mean symbol interarrival time, is a description of the tempo of operations. The situation assessment (SA) stage consists of a finite number of well-defined deterministic algorithms that the decisionmaker can choose from to process the measurement x and obtain the assessed situation z. The internal decisionmaking in this stage is the choice of algorithm f_i to process x. Therefore, each algorithm is considered to be active or inactive, depending on the internal decision u. In the response selection (RS) stage, which is similar to the situation assessment stage, one of the deterministic algorithms h_j is chosen according to the response selection strategy $p(v|z)$ to process the situation assessment z into an appropriate response y. Since no learning takes place during the performance of a sequence of tasks, the successive values taken by the variables of the model are uncorrelated, i.e., the model is memoryless. Hence, all information theoretic expressions presented later are on a per symbol basis.

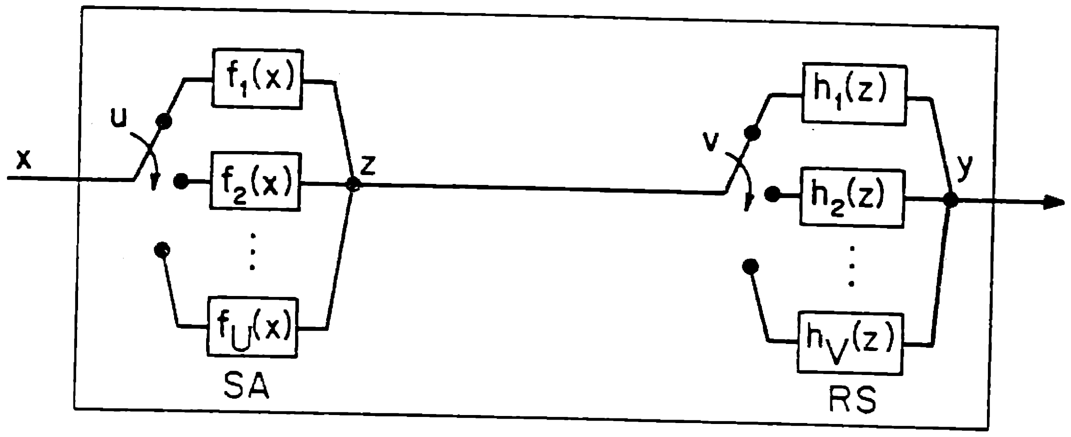

Figure 2.1. Basic Model of Decisionmaking Process

The model of the decisionmaking process shown in Fig. 2.1 may be viewed as a system S consisting of two subsystems, S^{SA} and S^{RS}, that correspond to each one of the two stages. The input to this system S is x and the output is y. Furthermore, let each algorithm f_i contain α_i variables denoted by

$$W^i = \{w_1^i, w_2^i, \ldots, w_{\alpha_i}^i\} \qquad i = 1,2,\ldots,U \tag{2.10}$$

and let each algorithm h_j contain α_j' variables denoted by

$$W^{U+j} = \{w_1^{U+j}, w_2^{U+j}, \ldots, w_{\alpha_j}^{U+j}\} \qquad j = 1,2,\ldots,V \tag{2.11}$$

It is assumed that the algorithms have no variables in common:

$$W^i \cap W^j = \phi \qquad \text{for } i \neq j \quad , \quad i,j = 1,2,\ldots, U+V \tag{2.12}$$

The subsystem S^{SA} is described by a set of variables

$$S^{SA} = \{u, W^1,\ldots,W^U, z\} \tag{2.13}$$

and subsystem S^{RS} by

$$S^{RS} = \{v, W^{U+1},\ldots,W^{U+V}, y\} \tag{2.14}$$

The four quantities stated in the Partition Law of Information for this system S can be evaluated with the following results.

$$G_t = T(x:y) = H(y) - H_x(y) \tag{2.15}$$

$$G_b = T_y(x:u, W^1,\ldots,W^{U+V},z,v) = H(x) - G_t \tag{2.16}$$

In this case, inputs not received or rejected by the system are not taken into account.

$$G_n = H_x(S^{SA}, S^{RS}) = H(u) + H_z(v) \tag{2.17}$$

$$G_c = T(u:w_1^1:\ldots:w_{\alpha_V}^{U+V}:z:v:y) = G_c^{SA} + G_c^{RS} + T(S^{SA}: S^{RS}) \tag{2.18}$$

where

$$G_c^{SA} = \sum_{i=1}^{U} [p_i g_c^i\ (p(x)) + \alpha_i H(p_i)] + H(z) \tag{2.19}$$

$$G_c^{RS} = \sum_{j=1}^{V} [p_j g_c^{U+j}(p(z|v=j)) + \alpha_j' H(p_j)] + H(y) \tag{2.20}$$

$$T(S^{SA} : S^{RS}) = H(z) \tag{2.21}$$

The expression for G_n shows that it depends on the two internal strategies $p(u)$ and $p(v|z)$. In the subsystem coordination expressions G^{SA} and G^{RS}, p_i and p_j are, respectively the probabilities that algorithms f_i and h_j have been selected, i.e., $p_i=p(u=i)$ and $p_j=p(v=j)$. The quantities g_c^k represent the internal coordination of the corresponding algorithms and depend on the probability distributions of their respective inputs. The quantity H is the entropy of a Bernoulli random variable with parameter p:

$$H(p) = -\ p \log p - (1-p) \log (1-p) \tag{2.22}$$

Eq. (2.18) states that the total coordination in the system S can be expressed according to Eq. (2.6) into the sum of the internal coordination within each subsystem given by Eqs. (2.19) and (2.20) and the coordination due to the interaction between the two subsystems given by Eq. (2.21).

The subsystem coordinations consist of terms reflecting the presence of switching due to the internal decision strategies $p(u)$ and $p(v|z)$. If there is no switching, i.e., if for example $p(u=i)=1$ for some i, then H will be identically zero and Eq. (2.19) will reduce to:

$$G^{SA} = g_c^i(p(x)) + H(z) \tag{2.23}$$

Finally, the quantity G, defined in Eq. (2.9) and interpreted as the total information processing activity of the system S, can serve as a measure of the workload of the decisionmaker in carrying out his task.

2.3 THE STOCHASTIC DECISIONMAKING MODEL

A crucial assumption in the basic decisionmaking model is that the algorithms are deterministic. Such an assumption, though restrictive as it seems, has simplified a great deal the evaluation of the various information theoretic expressions, particularly the noise G_n and the coordination G_c. The same basic model will be analyzed in detail in this section but with the assumption of deterministic algorithms removed. First, a stochastic algorithm needs to be analyzed using the information theoretic framework. A stochastic algorithm is different from a deterministic algorithm in that at least one of its variables is stochastic.

Consider an arbitrary algorithm in the decisionmaking model discussed in Section 2.2. The noise g_n and the coordination g_c of the algorithm are the two relevant quantities when the assumption of deterministic algorithms is removed. Suppose this typical algorithm, which receives an input x and yields an output y, is described by n variables, $w_1, w_2, \ldots, w_n$, of which $w_1 \equiv x$ and $w_n \equiv y$ (see Fig.2.2). Then g_n is a measure of the inherent uncertainty in the algorithm's variables when the algorithm's input is fully known. Mathematically, it is defined as

$$g_n = H_x(w_1, w_2, \ldots, w_n) \tag{2.24}$$

The algorithm coordination g_c is a measure of the constraints or interactions existing among all the variables within the algorithm. Mathematically, it is defined as

$$g_c = T(w_1 : w_2 : \ldots : w_n) \tag{2.25}$$

It will be interesting to determine how the two quantities, g_n and g_c of a stochastic algorithm differ from those of a deterministic algorithm.

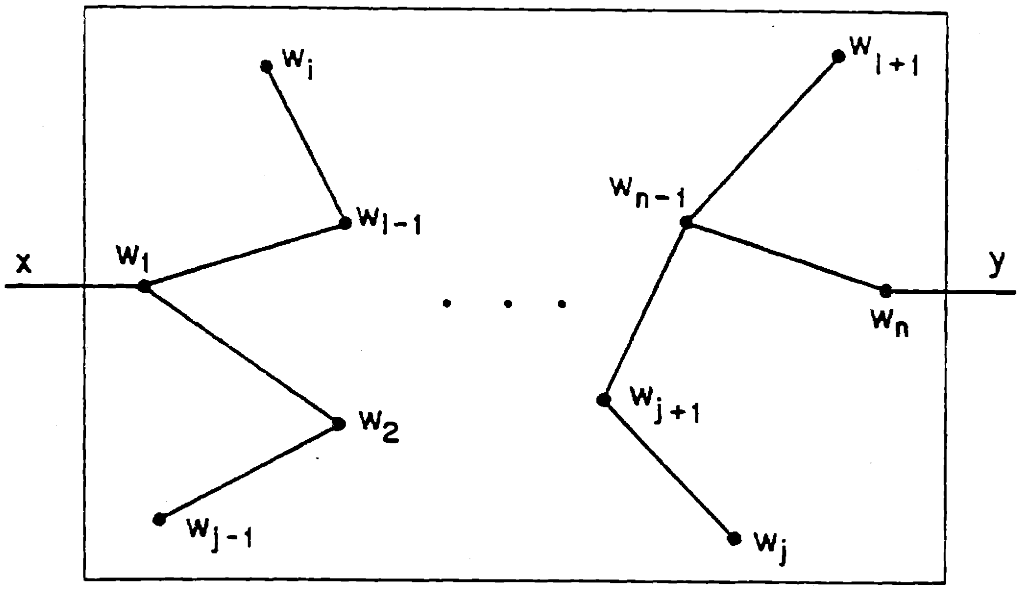

Figure 2.2. A Typical n-Variable Algorithm

<u>Deterministic Algorithm</u>

Since the variables w_i (i=1,2,...,n) of a deterministic algorithm are well-defined without any stochasticity, each of then can be reexpressed as a deterministic function of the input variable x. Hence, when the input value x is known, there is no ambiguity in determining the values of the w_i's. Thus, the g_n in Eq. (2.24) reduces to zero, implying there is no inherent uncertainty within a deterministic algorithm, i.e.,

$$g_n = 0 \tag{2.26}$$

According to the identities in information theory, the g_c in Eq. (2.25) can be rewritten as

$$g_c = \sum_{i=1}^{n} H(w_i) - H(w_1,w_2,\ldots,w_n) \tag{2.27}$$

The second term of Eq. (2.27) can be decomposed into

$$H(w_1,w_2,\ldots,w_n) = H(w_1) + H_{w_1}(w_2) + H_{w_1,\,w_2}(w_3) + \ldots + H_{w_1,w_2,\ldots,w_{n-1}}(w_n) \tag{2.28}$$

Since $w_1 \equiv x$ and the w_i's are all deterministic functions of x, all the terms in Eq. (2.28) except the first one reduce to zero; i.e.,

$$H(w_1,w_2,\ldots,w_n) = H(w_1) \tag{2.29}$$

Substitution of Eq. (2.29) into Eq. (2.27) yields

$$g_c = \sum_{i=2}^{n} H(w_i) \tag{2.30}$$

This result implies that the coordination of a deterministic algorithm is the sum of the marginal entropies of each individual variable within the algorithm except the one duplicating the input value. (In this case, the input variable is w_1).

Stochastic Algorithm

Since a stochastic algorithm has at least one stochastic variable, let the n variables describing the algorithm be partitioned into two mutually exclusive, collectively exhaustive subsets, D and $\bar{D}$, defined and enumerated as follows:

$$\bar{D} \equiv \text{The set of all the stochastic variables within the algorithm}$$

$$= \{w_i,\ w_{i+1},\ldots,w_{j-1},\ w_j\} \quad \text{where } 2 \lesssim i \lesssim j \lesssim n \tag{2.31}$$

$$D \equiv \text{The set of all the deterministic variables within the algorithm}$$

$$= \{w_1,w_2,\ldots,w_{i-1},\ w_{j+1},\ w_{j+2},\ldots,w_n\} \tag{2.32}$$

Since the order of the variables in a joint entropy expression is immaterial, the g_n in Eq. (2.24) can be written as:

$$g_n = H_x(w_1,w_2,\ldots,w_{i-1},\ w_i,w_{i+1},\ldots,w_{j-1},w_j,w_{j+1},\ldots,w_n)$$

$$= H_x(w_1,w_i,w_{i+1},\ldots,w_{j-1},w_j,w_2,\ldots,w_{i-1},w_{j+1},\ldots,w_n)$$

$$= H_x(w_1,\bar{D},\ D-\{w_1\})$$

$$= H_x(w_1) + H_{x,w_1}(\bar{D}) + H_{x,w_1,\bar{D}}(D-\{w_1\}) \tag{2.33}$$

The first term of Eq. (2.33) is equal to zero since $w_1 \equiv x$. Consider the last term of Eq. (2.33), the conditioning of this joint entropy can provide sufficient information to determine the values of the deterministic variables D, hence, this term reduces to zero. The second term of Eq. (2.33) is the joint entropy of all the stochastic variables conditioned on the full knowledge of the input value. Suppose the stochastic variables within the algorithm are defined to be independent of any other variables. Then the conditioning present in the second term is nullified. Consequently, Eq. (2.33) becomes

$$g_n = H_{x,w_1}(\bar{D})$$

$$= H(\bar{D})$$

$$= H(w_i, w_{i+1}, \ldots, w_{j-1}, w_j)$$

$$= H(w_i) + H_{w_i}(w_{i+1}) + H_{w_i,w_{i+1}}(w_{i+2})$$

$$+\ldots+ H_{w_i,w_{i+1},\ldots,w_{j-1}}(w_j) \tag{2.34}$$

Suppose the stochastic variables in $\bar{D}$ are defined to be independent of each other, then their individual marginal entropies will not be affected by the presence of any kind of conditioning. Hence, all the conditional entropies in Eq. (2.34) can equate to their marginal counterparts. As a result, g_n in Eq. (2.34) can be written as

$$g_n = H(w_i) + H(w_{i+1}) + H(w_{i+2}) + \ldots + H(w_j)$$

$$= \sum_{w_\lambda \varepsilon \bar{D}} H(w_\lambda) \tag{2.35}$$

This result shows that the inherent uncertainty of a stochastic algorithm is the sum of all the marginal entropies of each individual stochastic variable within the algorithm.

Accordingly, g_c can be evaluated as follows:

$$g_c = T(w_1:w_2:\ldots:w_{i-1}:w_i:w_{i+1}:\ldots:w_{j-1}:w_j:w_{j+1}:\ldots:w_n)$$

$$= \sum_{w_\lambda \varepsilon D} H(w_\lambda) + \sum_{w_\lambda \varepsilon \bar{D}} H(w_\lambda) - H(D,\bar{D}) \tag{2.36}$$

The last term of Eq. (2.36) can be reduced in a manner similar to that used in reducing g_n.

$$H(D,\bar{D}) = H(w_1,\bar{D}\ ,\ D-\{w_1\})$$

$$= H(w_1) + H_{w_1}(\bar{D}) + H_{w_1,\bar{D}}(D-\{w_1\}) \qquad (2.37)$$

The last term of Eq. (2.37) reduces to zero since the values of the variables in D can be determined unambiguously when the values of input $x \equiv w_1$ and the variables in $\bar{D}$ are known. The second term in Eq. (2.37) is equal to $H(\bar{D})$ since its conditioning can be nullified by the independence assumption made in defining the stochastic variables in $\bar{D}$. Furthermore, $H(\bar{D})$ has been explicitly analyzed in g_n and the result is expressed in Eq. (2.35). Hence, Eq. (2.37) can be written as

$$H(D,\bar{D}) = H(w_1) + \sum_{w_\lambda \varepsilon \bar{D}} H(w_\lambda) \qquad (2.38)$$

Finally, substituting Eq. (2.38) into Eq. (2.36) yields

$$g_c = \sum_{w_\lambda \varepsilon D-\{w_1\}} H(w_\lambda) \qquad (2.39)$$

This result implies that the coordination of a stochastic algorithm is the sum of all the marginal entropies of each individual deterministic variable within the algorithm except the one duplicating the input value. (Here the variable is w_1). It is interesting to note that this conclusion is the same as that of the case where all the variables are deterministic (i.e., $\bar{D} = \emptyset$).

In conclusion, when the Partition Law of Information applies to an algorithm whose stochastic variables, if any, are independent of each other and of any other variable within the algorithm, the inherent uncertainty g_n and the internal coordination g_c are analyzed to be:

g_n = sum of individual marginal entropies of all the stochastic variables within the algorithm.

g_c = sum of individual marginal entropies of all the deterministic variables within the algorithm except the one duplicating the input value.

<u>Analytic Expressions of the Stochastic Decisionmaking Model</u>

In order to find out how the various activity terms of the decisionmaker model are affected by introducing stochastic algorithms, the SA stage of the model is analyzed first. The DM model with its SA stage shown in Fig. 2.3 is the same as the one illustrated in Fig. 2.1 except that the variables within the algorithms can be either deterministic or stochastic.

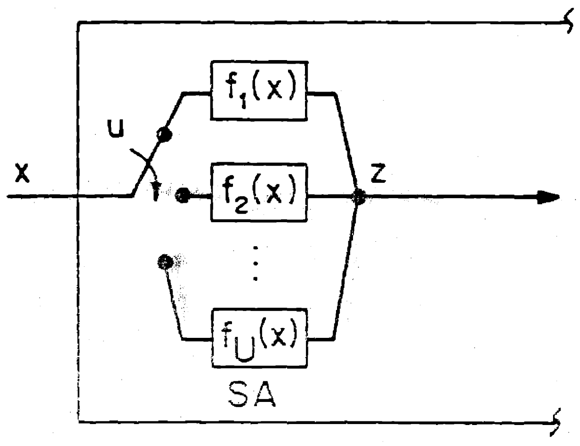

Figure 2.3. Situation Assessment Stage

Since the analytic expressions for the throughput and the blockage of the stochastic DM model are unaffected by the presence of stochastic variables within the model, only the expressions for the noise and the coordination of the model will be derived.

Noise

By definition, the noise present in the SA stage G_n^{SA} is given by

$$G_n^{SA} = H_x(u, W^1, W^2, \ldots, W^U, z) \tag{2.40}$$

which can be written as

$$G_n^{SA} = H_x(u) + H_{x,u}(W^1) + H_{x,u,W^1}(W^2) + \ldots + H_{x,u,W^1,W^2,\ldots,W^{U-1}}(W^U)$$

$$+ H_{x,u,W^1,W^2,\ldots,W^U}(z) \tag{2.41}$$

Since preliminary processing of the input does not exist in the SA stage, the decision strategy p(u) is independent of the input value x, i.e.

$$p(u|x) \equiv p(u);$$

hence the first term of Eq. (2.41) equates to H(u). The last term of Eq. (2.41) reduces to zero because the output value of the SA stage, z, can certainly be determined when all the algorithms' output values are known. Furthermore, given the values of the input variable x and the decision variable u, the knowledge of the values of the ith algorithm's variables W^i will not provide any more information to determine the values of the jth algorithm's variables W^j $(i \neq j)$ in the SA stage. Hence, all the remaining terms in Eq. (2.41) can be simplified to be their corresponding entropies conditioned on the knowledge of only the variables x and u. Equation (2.41) thus becomes

$$G_n^{SA} = H(u) + H_{x,u,}(W^1) + H_{x,u}(W^2) + \ldots + H_{x,u}(W^U)$$

$$= H(u) + \sum_{i=1}^{U} H_{x,u}(W^i) \tag{2.42}$$

By definition, the term $H_{x,u}(W^i)$ can be expressed as

$$H_{x,u}(W^i) = -\sum_{x,u} p(x,u) \sum_{W^i} p(W^i|x,u) \log p(W^i|x,u) \tag{2.43}$$

Since the input variable x and the decision variable u are independent, their joint distribution is

$$p(x,u) = p(x)p(u) \tag{2.44}$$

Substituting Eq. (2.44) into Eq. (2.43) and factoring p(u) out yields

$$H_{x,u}(W^i) = -\sum_{u} p(u) \sum_{x} p(x) \sum_{W^i} p(W^i|x,u) \log p(W^i|x,u) \tag{2.45}$$

when $u \neq i$, the variables W^i are all inactive and so their values are fixed without any uncertainty; the value of the expression in Eq. (2.45) is zero. Hence Eq. (2.45) can be written as

$$H_{x,u}(W^i) = p(u=i) \; [-\sum_{x} p(x) \sum_{W^i} p(W^i|x,u=i) \log p(W^i|x,u=i)] \tag{2.46}$$

The summation expression in the brackets of Eq. (2.46) is equivalent to the inherent uncertainty of the ith algorithm g_n^i when considered as a system with input x and output z according to Eq. (2.24). Therefore, Eq. (2.46) becomes

$$H_{x,u}(W^i) = p(u=i)\ g_n^i(p(x)) \tag{2.47}$$

Finally, substituting Eq. (2.47) into Eq. (2.42) yields

$$G_n^{SA} = H(u) + \sum_{i=1}^{U} p_i g_n^i(p(x)) \tag{2.48}$$

where $p_i = p(u=i)$ as defined in Section 2.2.

The analytic expression G_n^{SA} in Eq. (2.48) shows that the noise within the SA stage of a DM model with stochastic algorithms consists of two parts. The first part is the uncertainty in choosing the algorithm in the SA stage. It accounts for the amount of internal activity due to the decision strategy p(u). The second part is the summation of the inherent uncertainty of each individual algorithm weighted by the relative frequency of use. It measures the amount of stochasticity existed within the SA stage algorithms. Therefore, when the DM model has only deterministic algorithms, the second part of the expression for the noise does not exist.

The expression for the noise for the RS stage G_n^{RS} of the stochastic DM model can be derived in a similar manner with the following result:

$$G_n^{RS} = H_z(v) + \sum_{j=1}^{V} p_j\ g_n^{U+j}(p(z|v=j)) \tag{2.49}$$

where $g_n^k(k=U+1,U+2,\ldots,U+V)$ is the inherent uncertainty of the kth algorithm in the RS stage. The interpretation of the expression in Eq. (2.49) is the same as that of G_n^{SA} in Eq. (2.48).

Coordination

According to the principle of decomposition of coordination (i.e. Eq. (2.6)), the coordination terms within the SA stage and the RS stage as well as between the two stages are needed to evaluate the coordination of the entire stochastic DM model, i.e.,

$$G_c = G_c^{SA} + G_c^{RS} + T(S^{SA}:S^{RS})$$

Since the assumption that the algorithms are deterministic was never used in deriving the analytic expressions of G_c^{SA} and G_c^{RS} for the basic DM model [3], the expressions of G_c^{SA} and G_c^{RS} for the stochastic DM model should be identical to those of the basic model (Eqs. (2.19) and (2.20)).

The last term, coordination between the two stages, is evaluated as follows:

$$T(S^{SA}: S^{RS}) = H(S^{RS}) - H_{S^{SA}}(S^{RS}) \qquad (2.50)$$

Recall that S^{SA} and S^{RS} denote the sets of all the variables within the SA and the RS stages respectively. Since z is the only variable present in the SA stage, which has a direct influence on the variables in the RS stage. Hence, the second term of Eq. (2.50) can reduce to

$$H_{S^{SA}}(S^{RS}) = H_z(S^{RS}) \qquad (2.51)$$

Substituting Eq. (2.51) into Eq. (2.50) yields

$$T(S^{SA}:S^{RS}) = H(S^{RS}) - H_z(S^{RS}) = T(z:S^{RS}) = H(z) - H_{S^{RS}}(z) \qquad (2.52)$$

Since all the algorithms in the model are stochastic algorithms, there always exists a variable w_1^{U+v} duplicating the input value z within the $(U+v)^{th}$ algorithm which is active in the RS stage. Therefore, as w_1^{U+v} is a variable in the RS stage, its knowledge reduces the second term of Eq. (2.52) to zero. Thus Eq. (2.52) becomes

$$T(S^{SA}:S^{RS}) = H(z) \tag{2.53}$$

Hence, the entire expression for the coordination of the DM model with stochastic algorithms is the same as that with deterministic algorithms in Eqs. (2.18) - (2.21). It is interesting to note that under the same input distribution p(x) and decision strategies p(u) and p(v|z), if all the stochastic variables within the algorithms are removed from the model in such a way that the distributions p(z) and p(y) are not affected, then all the activities within the DM model will remain the same except for the noise G_n. This happens because when no stochastic variable exists in an algorithm, its inherent uncertainty g_n^i (i=1,2,...,U+V) will be equal to zero and the G_n expressions in Eqs. (2.48) and (2.49) will thus reduce to

$$G_n^{SA} = H(u) \tag{2.54}$$

$$G_n^{RS} = H_z(v) \tag{2.55}$$

Consequently, the total activity within the entire DM model will decrease.

2.4 BOUNDED RATIONALITY

The qualitative notion that the rationality of a human decisionmaker is not perfect, but is bounded, has been modeled as a constraint on the total activity G:

$$G = G_t + G_b + G_n + G_c \leq F \tau_o \tag{2.56}$$

where τ_o is a critical value of the mean symbol interarrival time and F is the maximum rate of information processing that characterizes the decisionmaker. With such an interpretation for F and τ_o, the total activity G is dependent on the mean symbol processing time t according to the following relation.

$$G = Ft \tag{2.57}$$

Consequently, the constraint in Eq. (2.56) becomes

$$Ft \leq F\tau_o \qquad \text{or} \qquad t \leq \tau_o \tag{2.58}$$

Eq. (2.58) implies that the decisionmaker must process his inputs at a rate that is at least equal to the rate with which they arrive. For a detailed discussion of this particular model of bounded rationality, see Boettcher and Levis [4].

CHAPTER 3

THE PREPROCESSOR MODEL

3.1 INTRODUCTION

A preprocessor (PP) is located between a source and a decisionmaker (DM). The PP may be included as a subsystem in the decisionmaker model (Fig. 3.1), when viewed as an extension of the basic DM model. On the other hand, the PP may be seen as a separate, external decision aid to the decisionmaker (Fig. 3.2). The presence of PP within both contexts is expected to reduce the subsequent processing load.

There are several reasons for including preprocessors in a decisionmaking organization. The fundamental one is to reduce the workload of the decisionmakers (DMs) by doing some of the data processing. Another one is to filter out extraneous data, thus giving the DM more time and resources to process the significant data. Inevitably, the increase in the capability of a PP will necessitate greater complexity in its internal structure, accompanied with higher processing activity, and longer processing time.

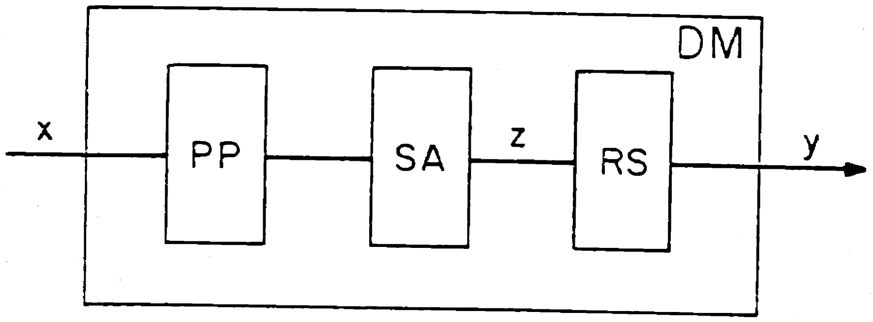

Figure 3.1. Preprocessor as an Internal Subsystem

There are two types of input variables feeding into the basic DM model, the input variable x that describes the arriving inputs and the decision variable u that describes the decision strategy used to process the inputs.

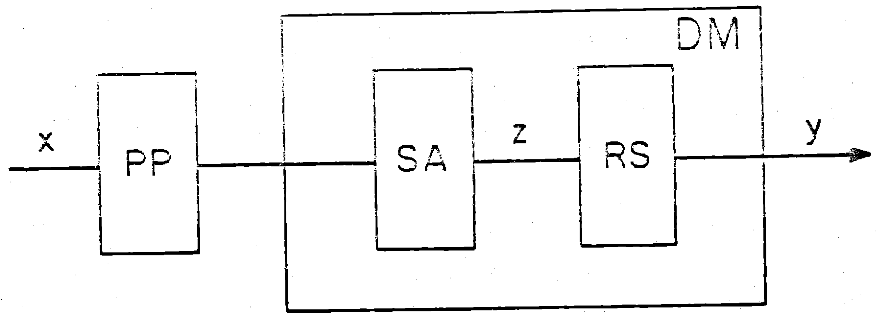

Figure 3.2. Preprocessor as an External Decision Aid

The PP operates on these two variables to achieve its purpose. Hence, the PP simply consists of a single algorithm for preliminary processing of the input. It yields a SA decision strategy, and sometimes a modified input to the DM. Consequently, if the function of a PP is specified, i.e., the algorithm and the interconnections of the internal variables, the activity within the PP is computable, just like for the other algorithms within the SA and RS stages of the DM model. A simple example of a PP is given in the following section to illustrate how the various activity terms can be computed for a specified algorithm with well-defined internal variables.

3.2 THE BASIC PREPROCESSOR

Based on the above discussion, an example of a basic PP model, which yields only one output, the decision strategy u for processing the input in the SA stage of a DM model, is illustrated in Fig. 3.3. From its environment, the PP receives an input datum x, an element of a finite alphabet X. Most of the input elements of the alphabet X can be processed better (i.e., more efficiently with more accurate results) by some subset of the SA algorithms. This special set of algorithms is termed the "desired decision strategy" specific to that particular input element. On receiving an input element, the PP first examines it, then outputs a desired decision strategy to the DM. This decision strategy will determine how the algorithms in the situation assessment stage of the DM will process the input datum x.

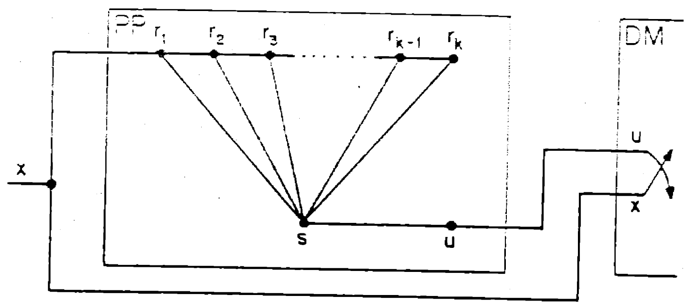

Figure 3.3. Basic Model of Preprocessor

Usually, there are more than one input elements in X with the same desired decision strategy. Hence, the input alphabet X can be divided into a number of mutually exclusive, collectively exhaustive subsets, or partitions, $(X_1,X_2,...,X_k)$ by grouping those input elements with the same desired decision strategy into the same partition. The desired decision strategy can be described by a probability vector whose elements specify the relative frequencies of employing the various algorithms in the SA stage of the DM.

The function of the PP is to identify every arriving input and match it to a desired decision strategy. Mathematically, there exists a many-to-one mapping from the finite set of input alphabet X to the finite probability vector space **P**, or a one-to-one mapping from the finite set of input partitions $\{X_1,X_2,...,X_k\}$ to the finite probability vector space **P**. To model such a mapping (only one-to-one mapping is considered here since it is simpler) within the information theoretic framework, the two sets of interest have to be clearly defined.

Let $\tilde{X}$ be a set of disjoint subsets X_i $(i=1,2,...,k)$ of the input alphabet X, i.e., for each $x \varepsilon X$,

$$x \varepsilon X_i \qquad \text{for some } i$$

and

$$x \notin X_j \qquad \text{for } j \neq i, \quad i,j=1,2,\ldots,k.$$

Hence, the sets X_i can be defined as

$$X = \bigcup_{i=1}^{k} X_i \qquad \text{where } X_i \in \tilde{X}$$

$$X_i \cap X_j = \emptyset \qquad \text{for } X_i, X_j \in \tilde{X},\ i \neq j$$

Let **P** be defined as a set of U-dimensional vectors $\underline{p}$ whose elements describe the relative frequencies of employing the U algorithms in the SA stage of the DM, i.e.,

$$\underline{p} = (p_1, p_2, \ldots, p_U)$$

where

$$0 \leq p_i \leq 1 \quad , \qquad i = 1,2,\ldots,U$$

and

$$\sum_{i=1}^{U} p_i = 1$$

The one-to-one mapping from $\tilde{X}$ to P can be defined now. Let $r_1, r_2, \ldots, r_k$ be the internal variables in the basic PP model that distinguish to which partition an arriving input x belongs. The r_i's (i=1,2,...,k) are boolean variables assuming values "true" and "false" according to the validity of the statement: "The input x belongs to the input partition X_i", i.e.,

Given an input x, then

$$r_i = \begin{cases} \text{"true"} & \text{if } x \in X_i \\ \text{"false"} & \text{otherwise} \end{cases} \qquad i = 1,2,\ldots,k$$

Another internal variable in the PP model is s, a numeric variable, giving explicit account of which partition the arriving input x belongs to. It yields the value i whenever r_i yields a "true" value, $1 \leq i \leq k$.

Specifically,

$$s = i \quad \text{if and only if} \quad r_i = \text{"true"} \quad \text{for some } i$$

$$\text{and} \quad r_j = \text{"false"} \quad \text{where } j \neq i \quad ; \quad i,j=1,2,\ldots,k$$

The output variable of the PP model is a U-dimensional probability vector $\underline{u} \in \mathbf{P}$. This vector $\underline{u}$ gives the DM a specific decision strategy for processing each arriving input x. Its value is determined by the variable s which specifies the input partition. Since each input partition is associated with a desired decision strategy, let $\underline{p}^i$ be the desired decision strategy for the partition X_i, i=1,2,...,k. Hence $\underline{u}$ can be precisely defined as follows:

$$\underline{u} = \underline{p}^i \quad \text{if } s = i \,, \quad \text{for some } \underline{p}^i \in \mathbf{P}.$$

When the internal variables in the PP are defined as above, the one-to-one mapping from the input partition set $\tilde{X}$ to the probability vector space **P** can be modelled. The activity within the PP can now be analyzed within the information theoretic framework.

The addition of the PP to the DM model will necessarily increase the total activity or workload of the system, because of the k+1 extra variables, $r_1,r_2,\ldots,r_k$ and s. The type and amount of additional activity can be determined as follows.

If the preprocessor PP is considered as a separate subsystem of the overall DM system, with input x, internal variables $r_1, r_2, \ldots, r_k$, s and output variable $\underline{u}$, then throughput G_t, blockage G_b, noise G_n, and coordination G_c for the PP can be calculated according to the Partition Law of Information in Section 2.1. (Refer to APPENDIX A for the detailed derivation.)

$$G_t = H(x:\underline{u}) = H(\underline{u}) - H_x(\underline{u}) \tag{3.1}$$

$$G_b = H(x) - G_t \tag{3.2}$$

$$G_n = H_x(r_1, r_2, \ldots, r_k, s, \underline{u}) = H_s(\underline{u}) = H_x(\underline{u}) \tag{3.3}$$

$$G_c = T(r_1 : r_2 : \ldots : r_k : s : \underline{u})$$

$$= H(s) + H(\underline{u}) - H_s(\underline{u}) + T(r_1 : r_2 : \ldots : r_k) \tag{3.4}$$

The noise G_n, which represents the uncertainty in the PP system when the input is known, is computed, as expected, to be $H_x(\underline{u})$ in Eq. (3.3). The dependence of the decision variable $\underline{u}$ on the input variable x shows that the PP has utilized the input information to affect the internal decision strategy in the SA stage of the DM. How the dependence of $\underline{u}$ on x affects the reduction in activity in the DM and hence offsets the increase in processing load due to the presence of PP requires further investigation.

The noise in the whole DM system is the sum of the noise within the PP and the noise within the DM. Since the latter is zero due to the presence of deterministic algorithms, the noise of the whole system is $H_x(\underline{u})$.

The total coordination in the whole DM system S, G_c^S, can be decomposed into the sum of the internal coordination within each subsystem and the coordination due to the interaction between the two subsystems. The latter coordination $T(S^{PP}:S^{DM})$ can be calculated to be (for the derivation, see APPENDIX A).

$$T(S^{PP}:S^{DM}) = H(x) \tag{3.5}$$

The second subsystem of the DM system, other than the PP, is the DM itself. Its coordination expression has been derived in the previous chapter, in Eq. (2.18). Hence, the total coordination of the whole DM system can be found, using Eqs. (3.4), (2.18), (3.5), according to the expression:

$$G_c^S = G_c^{PP} + G_c^{DM} + T(S^{PP}:S^{DM}) \tag{3.6}$$

where G_c^A is the coordination of subsystem A.

With the noise G_n and coordination G_c for the whole DM system defined, the total activity G^S of the system can be determined since

$$G_t^S + G_b^S = H(x). \tag{3.7}$$

By introducing the preprocessor which utilizes input information in order to select better strategies, the activity within the DM may be appropriately reduced. However, the equations show that unless the selected strategy can exploit the behaviour of the arriving inputs and the capacity of the available algorithms to achieve an extremely low processing activity level in the DM on the average, the processing load within the PP will definitely exceed the reduction in activity within the DM. The circumstances under which the presence of the PP will improve the overall system performance subject to various time and resource constraints require precise formulation of performance measures within such a context.

To see how the processing activity of a DM system is affected by the presence of a PP which acts either as an internal subsystem or as an external decision aid to the decisionmaker, the expressions for various activity terms are listed below for the three decisionmakers for comparison and discussion. These DM systems are made up of a basic DM model, as described in the last chapter, and a PP, as described in the last section, which functions either as a subsystem or a decision aid to the DM.

(a) <u>Unaided Basic DM Model</u>

$$G_t + G_b = H(x)$$

$$G_n^{SA} = H(u)$$

$$G_n^{RS} = H_z(v)$$

$$G_c^{SA} = \sum_{i=1}^{U} [p_i g_c^i (p(x)) + \alpha_i H(p_i)] + H(z)$$

$$G_c^{RS} = \sum_{j=1}^{V} [p_j g_c^{U+j} (p(z|v=j)) + \alpha_j' H(p_j)] + H(y)$$

$$T(S^{SA}:S^{RS}) = H(z)$$

(b) <u>DM with PP as an internal Subsystem</u>

$$G_t + G_b = H(x)$$

$$G_n^{SA} = H_x(u)$$

$$G_n^{RS} = H_z(v)$$

$$G_c^{PP} = T(r_1:r_2:\ldots:r_k) + T(s:u) + H(s)$$

$$G_c^{SA} = \sum_{i=1}^{U} [p_i g_c^i (p(x|u=i)) + \alpha_i H(p_i)] + H(z)$$

$$G_c^{RS} = \sum_{j=1}^{V} [p_j g_c^{U+j} (p(z|v=j)) + \alpha_j' H(p_j)] + H(y)$$

$$T(S^{PP}:S^{SA}:S^{RS}) = H(x,u) + H(z)$$

(c) <u>DM with PP as an External Decision Aid</u>

$$G_t + G_b = H(x,u) \qquad (3.8)$$

$$G_n^{SA} = H_x(u)$$

$$G_n^{RS} = H_z(v)$$

$$G_c^{SA} = \sum_{i=1}^{U} [p_i g_c^i (p(x|u=i)) + \alpha_i H(p_i)] + H(z) \qquad (3.9)$$

$$G_c^{RS} = \sum_{j=1}^{V} [p_j g_c^{U+j} (p(z|v=j)) + \alpha_j' H(p_j)] + H(y)$$

$$T(S^{SA}:S^{RS}) = H(z)$$

In order to analyze how the terms in the activity expressions change, it is necessary to know how the addition of a PP to the basic DM system has influenced the probability distributions of individual variables within the system. If a PP affects only the decision variable u of the DM system, then the probability distribution p(u) is not predetermined as in the unaided system, but rather is derived from the desired decision strategy p(u|x) according to the formula

$$p(u) = \sum_{x} p(u|x)\ p(x)$$

where p(x) is the distribution of the arriving inputs and p(u|x) is the given strategy in the SA stage. Since z is the output of the deterministic algorithms in the SA stage, it is a function of u and x. Furthermore, since the presence of a preprocessor affects the distribution p(u), the distribution p(z) will be different as a result. The term p(v=j) is derived from the expression

$$p(v) = \sum_{z} p(v|z)\ p(z)$$

where p(v|z) is the given strategy in the RS stage; so if p(z) has changed, p(v) will no longer be the same. Again, since y is the output of the deterministic algorithms in the response selection stage, it is a function of z and v; hence the distribution p(y) will not remain the same when the distributions p(z) and p(v) have changed. Consequently, the addition of such a simple PP to a DM system affects the distribution of every variable in the above activity expressions except p(x). Therefore, the only terms that will remain unchanged in the above expressions by the introduction of a PP to the system are H(x), α_i, α_j', and g_c^i. It follows that whether a PP will make a significant improvement in the total system activity of the DM system depends on the specific problem, i.e., it requires specification of the input distribution, the decision strategies and the algorithmic structures and functions within the PP as well as within the DM model.

3.3 PREPROCESSOR WITH FILTERING

Sometimes the PP modifies the input to the DM system in order to reduce the processing activity within the DM. For instance, when the arriving inputs contain much extraneous data which should be ignored by the DM, the PP can filter them out by blocking them so that no such input will be transmitted to the DM. Such a function may be realized in the previous example by adding two more internal variables, r_0 and x_0. (See Figure 3.4).

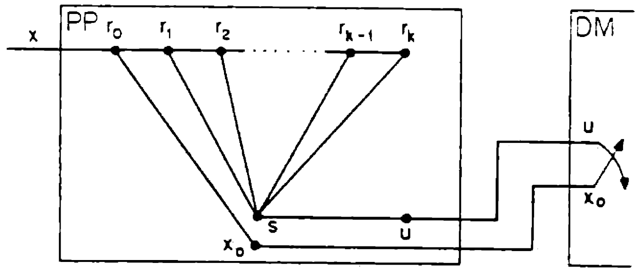

Fig. 3.4. Preprocessor Model with Filtering

The boolean variable r_0, like the other boolean variables, r_i, is to distinguish whether the input x belongs to a specific input partition X_0 of the input alphabet X. The partition X_0 contains all those extraneous input elements in the input alphabet X. The variable x_0 will duplicate the input value x and transmit it to the DM process accordingly, if it is identified as not belonging to X_0. Otherwise, the variable x_0 will remain inactive so that the input, identified to be one of those irrelevant data, will be blocked within the PP. Mathematically, r_0 and x_0 are defined as follows.

Given an input x,

$$r_0 = \begin{cases} \text{"true"} & \text{if } x \in X_0 \\ \text{"false"} & \text{otherwise} \end{cases} \tag{3.10}$$

$$x_o = \begin{cases} \square & \text{if } x \in X_o \\ x & \text{otherwise} \end{cases} \qquad (3.11)$$

The addition of these two variables, r_o and x_o, to the PP requires some modification in the definition of the numeric variable s and the output variable $\underline{u}$.

$$s = i \quad \text{if and only if} \quad r_i = \text{"true"} \quad \text{for some } i$$
$$\text{and } r_j = \text{"false"} \text{ where } j \neq i \;;\quad i,j=0,1,2,\ldots,k \qquad (3.12)$$

$$\underline{u} = \begin{cases} \underline{p}^i & \text{if } s = i, \quad 1 \leq i \leq k \quad , \quad \text{or some } \underline{p}^i \varepsilon P \\ \square & \text{if } s = 0 \end{cases} \qquad (3.13)$$

In other words, no decision strategy will process an irrelevant input datum which is blocked within the PP. When the PP has internal variables defined as above, the various kinds of activity within the PP can be analyzed with the following results. (For the derivations, see APPENDIX B).

$$G_t = T(x:\underline{u},x_o) = H(\underline{u},x_o) \qquad (3.14)$$

$$G_b = H(x) - G_t$$

$$G_n = H_x(r_o,r_1,\ldots,r_k,x_o,s,\underline{u}) = H_s(\underline{u}) \qquad (3.15)$$

$$G_c = T(r_o:r_1:\ldots:r_k:x_o:s:\underline{u})$$

$$= T(r_o:r_1:\ldots:r_k) + H(r_o) + H(s) + H(\underline{u}) - H_s(\underline{u}) \qquad (3.16)$$

Hence, the total activity of the PP can be computed as

$$G^{PP} = G_t + G_b \div G_n + G_c$$

$$= H(x) + H(s) + H(\underline{u}) + H(r_o) + T(r_o:r_1:\ldots:r_k) \tag{3.17}$$

Again, the effect of introducing such a PP to a DM system can be analyzed by listing the activity expressions for comparison. But since the presence of the PP will certainly affect all the probability distributions, including the input distribution p(x) to the DM, to discover under what conditions will the addition of such a PP improve the DM system is again problem-dependent.

Although no general result about the design of a PP to ascertain the reduction of processing activity within the DM model is known, the above example of the PP model with filtering guarantees a more preferable DM system when the DM exhibits bounded rationality behaviour. In previous work, [3], [4], the bounded rationality constraint on the DM is modelled as a maximum allowable rate of total processing activity F. Mathematically, the constraint is expressed as

$$G \leq F\tau \tag{3.18}$$

where G is the total processing activity within the DM model and τ is the mean interarrival time of input symbols to the DM model. Equation (3.18) can be re-expressed as

$$G \leq G_r \qquad \text{where} \qquad G_r = F\tau$$

Since F is a parameter specific to a decisionmaker with bounded rationality, the threshold of total processing activity G_r within the DM depends on τ, the mean symbol interarrival time. The retention of extraneous data within the PP model with filtering reduces the rate of arriving inputs to the DM, thus increases the mean symbol interarrival time τ, resulting in a higher threshold of total processing activity G_r for the DM. The increase of the threshold value G_r implies a less constrained DM. Therefore, the introduction of a PP with filtering to a basic DM system with bounded rationality behavior will not degrade performance but may actually improve it.

3.4 A NUMERICAL EXAMPLE

In order to demonstrate how a preprocessor can assist a decisionmaker, a numerical example of a PP with filtering is constructed below. The objective is to illustrate how a preprocessor reduces the activity level and increases the processing threshold of a stochastic decisionmaking system. The internal structure of the PP as well as the associated DM are shown in Fig. 3.5. For simplicity, only the SA stage of the DM model is considered here.

The elements of the input alphabet $X = \{x_1, x_2, x_3, x_4\}$ arrive at the PP in an equally likely manner at a rate of λ symbols per second. Altogether, three partitions are defined in X. They are

$$X_0 = \{x_1, x_2\} \quad , \quad X_1 = \{x_3\} \quad , \quad X_2 = \{x_4\}.$$

Each of these partitions has its own desired decision strategy defined as follows. Again for simplicity, only pure strategies are considered.

$$p(u|X_0) = \begin{cases} 1 & , \quad u = \square \\ 0 & , \quad \text{otherwise} \end{cases} \tag{3.19}$$

$$p(u|X_1) = \begin{cases} 1 & , \; u = 1 \\ 0 & , \; \text{otherwise} \end{cases} \tag{3.20}$$

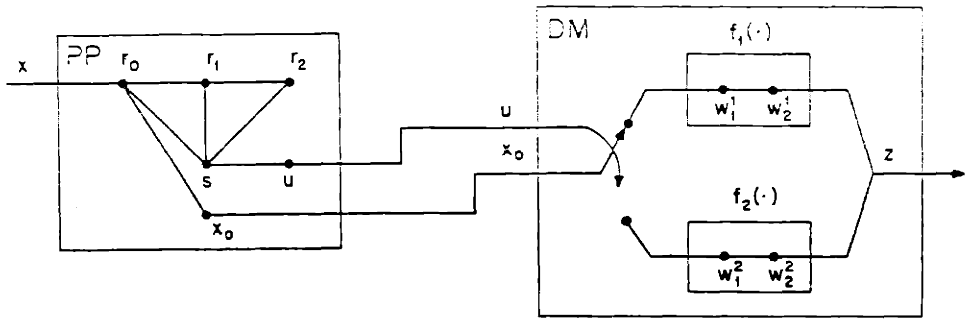

Figure 3.5. Detailed Internal Structure of a PP with a DM System

$$p(u|X_2) = \begin{cases} 1 & , \ u = 2 \\ 0 & , \ \text{otherwise} \end{cases} \tag{3.21}$$

The notation u = [] means that no further processing is required for the elements in the partition X_0. The job of the PP is to identify to which partition the arriving input belongs, then yield the appropriate desired decision strategy and input data to the DM for processing.

The two stochastic algorithms in the DM can be specified as

$$f_1(x) = \begin{cases} x + 1 & \text{w.p. } 0.5 \\ x & \text{w.p. } 0.5 \end{cases} \tag{3.22}$$

$$f_2(x) = \begin{cases} x - 1 & \text{w.p. } 0.5 \\ x & \text{w.p. } 0.5 \end{cases} \tag{3.23}$$

With the functions of the PP and the two algorithms f_1 and f_2 defined, the internal variables and the interconnections among them can be defined. Hence, once p(x) is given, the activities within the PP as well as the DM can be evaluated.

According to Eq. (3.17) the activity within the PP is

$$G^{PP} = H(x) + H(s) + H(u) + H(r_0) + T(r_0:r_1:r_2) \tag{3.24}$$

According to Eqs. (3.8), (2.48), and (3.9), the activity within the DM system is

$$G^{DM} = H(x_0,u) + H_x(u) + \sum_{i=1}^{2} p_i g_n^i (p(x|u=i)) + p_i g_c^i(p(x|u=i))$$

$$+ \alpha_i H(p_i) + H(z) \tag{3.25}$$

Based on the PP and the DM defined above, Eqs. (3.24) and (3.25) can be used to compute the total activities with the following results. (Refer to APPENDIX C for the detailed computation).

$$G^{PP} = 7.12$$

$$G^{DM} = 7$$

Now consider the overall activity of the DM system as well as its system performance. If the PP is considered as a subsystem of the DM system (Fig. 3.6), then the total activity of the DM system will be

$$G^1 = G^{PP} + G^{DM} = 14.12$$

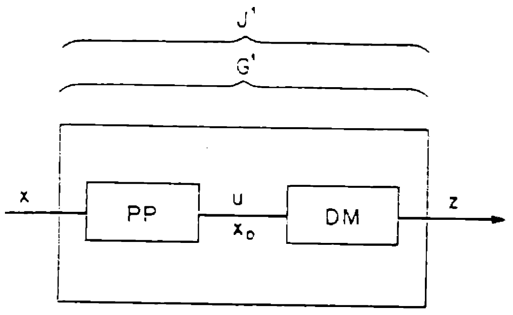

Figure 3.6. PP as a Subsystem of the DM system

If the PP is considered as an external decision aid of the DM system (Fig. 3.7), then the total activity of the DM system will not include the activity within the PP. The system activity is

$$G^2 = G^{DM} = 7$$

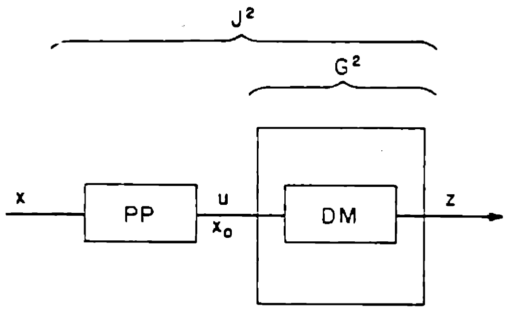

Figure 3.7. PP as an External Decision Aid of the DM System

It is obvious that the system with PP as a decision aid has reduced the system's activity level by moving some of the processing work outside of the system. Here in this example, it is assumed that the moving of the boundary line of the DM system does not create any interface problem to increase the system activity. As for the system performance, since the same task is processed with the same input and output behavior, i.e., same p(x) and p(z), both realizations shown in Figs. 3.6 and 3.7 will produce the same system performance. However, this is not true when the DM exhibits bounded rationality. Two reasons account for this. First, the activity of the DM system with PP as a subsystem is generally greater than that with PP as a decision aid. Hence, the workload of the former is more likely to exceed the system threshold G_r. Second, the addition of the PP as a decision aid reduces the input arrival rate of the DM system from λ to $\lambda/2$. As a result, the interarrival time τ and thus the system threshold $G_r = F\ \tau$ are doubled. An increase in the threshold G_r implies a smaller possibility of leading to overload in the DM. The two facts can be demonstrated by a typical performance-workload (i.e., J-G) plot for a DM system with non-deterministic decision strategy (Fig. 3.8).

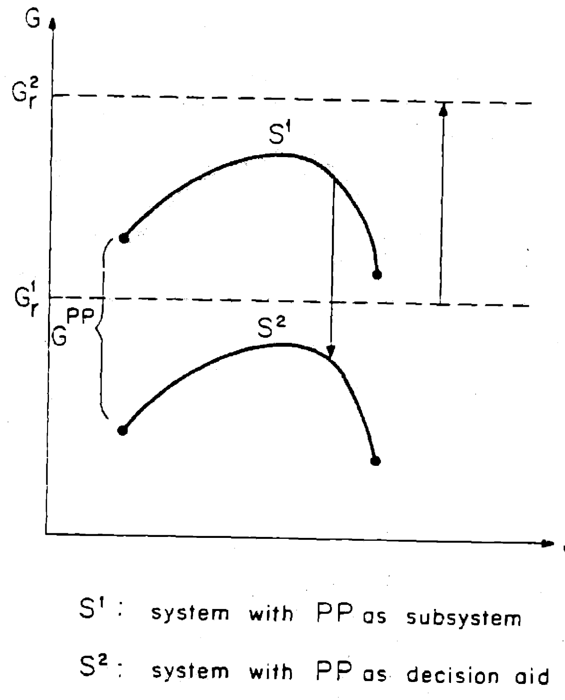

Figure 3.8. A typical J-G Plot for Systems with Preprocessor

Two features are shown on the plot when the PP's role in the DM system has changed from subsystem to decision aid. First, the moving of the performance-workload locus down by the amount G^{PP} is due to the removing of some of the processing load out of the system. Second, the translation of the threshold line up is due to the less frequent arrival of input symbols.

In general, when a DM exhibits bounded rationality, the unaided system (Fig. 3.6) will have more decision strategies leading to overload than the aided system (Fig. 3.7).

On the other hand, there are times when a decision aid may not be appropriate. It is because in reality, decision aids are extensions of the decisionmaker's ability to gather, process and disseminate information. If they are not designed properly, they may make serious mistakes, impairing the performance of the decisionmaker. In the PP model developed so far, such a phenonmenon can be illustrated by including stochastic variables to describe possible error generation within the decision aid. Under such circumstances, the input entropy and the input arrival rate to the DM are likely to increase. Thus, the processing load within the DM may increase up to the point of exceeding the decreasing system threshold G_r, indicating a degradation of performance. Hence, a decision aid does not always improve system performance.

Finally, a stochastic DM system has a greater need of a properly designed PP than a deterministic DM system to improve performance. This is due to the greater activity within a stochastic DM system resulting from the stochasticity within the algorithms of the system (i.e., including the increase of the number of stochastic variables as well as the increase in the entropy of these individual variables).

CHAPTER 4

APPLICATIONS

4.1 INTRODUCTION

At the end of the last chapter, a numerical example illustrating the use of a preprocessor was presented. It was stated in the discussion that it is not always advantageous to introduce a PP as a decision aid because the PP may not be properly designed and error-free. In this chapter, two generic examples of preprocessors with well-defined purposes are described as well as their corresponding decisionmaking systems. The first PP, which will act as a decision aid, processes the input data, while the second one aggregates the input data.

So far, the criteria for determining whether a PP is a subsystem or a decision aid for a DM system have not yet been specified. They often depend on the usage of the PP, and on the evaluation of system activity and system performance. Since it is assumed that no filtering occurs in both examples, the input arrival rate for the preprocessors will remain constant and thus the bounded rationality constraint will remain unchanged. Hence, the system performance depends entirely on the system activity. The total activity of the DM system with and without a preprocessor in the two examples that follow will then be evaluated to determine the role of the PP in the system. As a result, guidelines for determining the role of a PP in a DM system will evolve.

In the two examples, the aspect of determining the decision strategy conditioned on the knowledge of the input values in the PP is ignored. The preprocessor's function is to modify the input data to affect reduction in the processing workload within the DM.

4.2 PREPROCESING OF DATA

The function of the preprocessor in this case is to reduce the workload of the decisionmaker by processing some of the data before they are received by the DM. In order to evaluate the utility of the PP, two different decisionmaking systems, shown in Figs. 4.1 and 4.2, are considered.

The first one represents an unaided DM while the second one represents a DM with a preprocessor. The DMs in both systems have identical structures and algorithms. The PP in the second system is an algorithm that performs the same function as the SA stage of the first DM system. Since the PP is just an algorithm without any decisionmaking capability, the SA stage of the first DM system has to employ a pure strategy, i.e., the same algorithm is always selected to process the arriving inputs. Under such conditions, the decisionmaking process in both systems are identical. In order to allow the two systems to do the same task with the same performance, so that the activity or workload can be properly compared, the algorithms within the processes are assumed to be deterministic and their output distributions p(y) identical. The latter result will be ensured, if the RS stage in the DM of both systems has the same input distribution p(z) and decision strategy p(v|z). Since the second system has executed the SA job within its PP, its SA stage simply transmits its input to an identity algorithm so that the distribution p(z) within both systems will be identical.

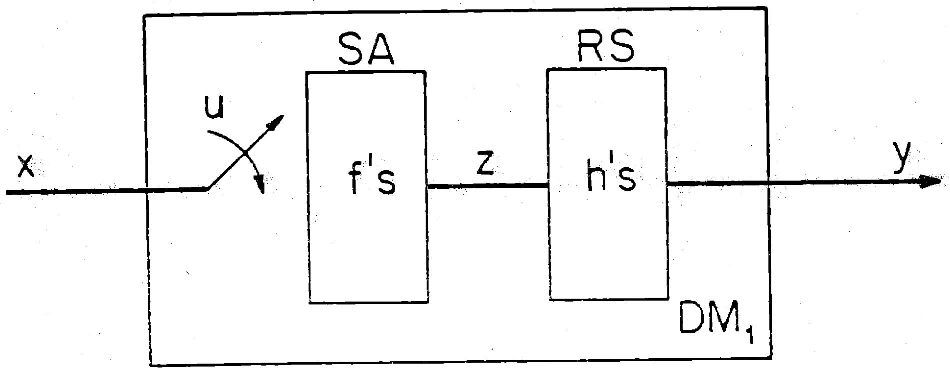

Figure 4.1. Basic DM System

With such a processing setup, the two systems are capable of doing the same task with the same performance. To demonstrate the utility of the PP in the second system, the total acivity within each system has to be evaluated.

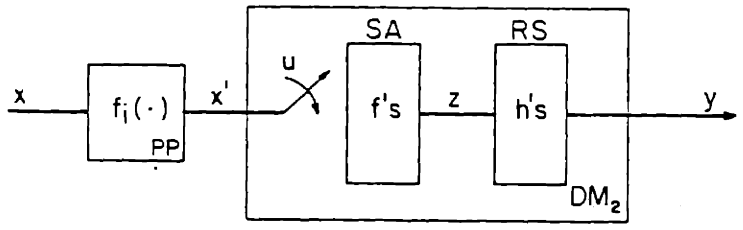

Figure 4.2. Aided DM System

Define:

$$\text{Activity within First System} = G_{DM_1} \tag{4.1}$$

$$\text{Activity within Second System} = G_{DM_2} + G_{PP} \tag{4.2}$$

where G_i is the total activity with subsystem i.

In accordance with the results presented in Chapter 2, the general expression for a decisionmaker is given by:

$$G_{DM} = H(x) + H(u) + \sum_{i=1}^{U} p_i g_c^i + \sum_{i=1}^{U} \alpha_i H(p_i) + H(z) + H_z(v)$$

$$+ \sum_{j=1}^{V} p_j g_c^{U+j}(p(z|v=j)) + \sum_{j=1}^{V} \alpha_j' H(p_j) + H(y) + H(z) \tag{4.3}$$

Since the response selection process, input distribution p(z) and decision strategy $p(v|z)$ of the RS stage in both systems are assumed identical, the activity within the RS stage of the two system is the same, denoted by G_{RS}.

$$G_{RS} = H_z(v) + \sum_{j=1}^{V} p_j g_c^{U+j}(p(z|v=j)) + \sum_{j=1}^{V} \alpha_j' H(p_j) + H(y) + H(z) \quad (4.4)$$

Assume that the pure strategy in the SA stage of the first system is such that the ith algorithm is always selected (i.e., $p(u=i)=1$) whereas in the second system the first algorithm, which is an identity algorithm, is always selected. Then the activity of each decisionmaker in the two systems is obtained from the expressions in Eqs. (4.3) and (4.4)

$$G_{DM_1} = H(x) + g_c^i + H(z) + G_{RS} \quad (4.5a)$$

$$G_{DM_2} = H(x') + g_c^1 + H(z) + G_{RS} \quad (4.5b)$$

The preprocessor in the second system consists of a single algorithm without any decisionmaking capability. The algorithm in this case is identical to the ith algorithm of the SA stage, hence the activity within the PP is simply

$$G_{PP} = H(x) + g_c^i \quad (4.6)$$

The function of the identity algorithm $f_1(\cdot)$ is to transmit its input to its output without any internal processing. Therefore, there is only one output variable z which duplicates the input value x and there are no internal variables. (See Fig. 4.3)

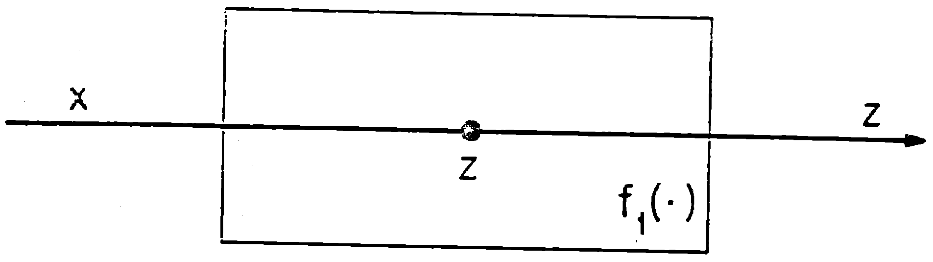

Figure 4.3. The Identity Algorithm

Hence, its internal coordination is

$$g_c^1 = T(\phi:z) = 0 \tag{4.7}$$

Since the internal coordination of any nontrivial algorithm (i.e., other than the identity algorithm) is greater than zero, $g_c^i > 0$, it follows that

$$g_c^i > g_c^1 \tag{4.8}$$

Given that the PP contains a deterministic algorithm without any decisionmaking capability, then

$$H(x) \geq H(x') > 0. \tag{4.9}$$

According to Eqs. (4.5a) to (4.9), the following two inequalities will always be valid.

$$G_{DM_1} > G_{DM_2} \tag{4.10}$$

$$G_{DM_1} < G_{DM_2} + G_{PP} \tag{4.11}$$

From the expressions of the total activity within the two DM systems given in Eqs. (4.1) and (4.2), the inequalities in Eqs. (4.10) and (4.11) show that even though the workload of the first system is less than that of the second system, the presence of the PP makes the workload of the second decisionmaker to be less than that of the first one. Such a result verifies the utility of the PP in terms of its ability to reduce the activity within a decisionmaker.

This result illustrates the utility of moving some processing function out of the situation assessment stage of a decisionmaker by designing preprocessors that carry out mechanistic tasks, e.g., signal processing.

4.3 AGGREGATION OF DATA

In a decisionmaking process, information aggregation is necessary when the data complexity is high and the time available for decision is short. An aggregation process reduces the input entropy by grouping the input data into several zones of indifference [9]. All the data in the same zone of indifference are recognized as the same super datum in the decisionmaking process. As a result, the modified input alphabet with a reduced entropy will affect the processing workload of the DM. This concept will be illustrated by a simple example.

In this example, the aggregation process in the PP is a truncation operation that converts any rational number into the greatest integer equal to or less than the original number. For instance, any number x in the semi-closed interval [n,n+1) will be truncated to the integer n. Such an operation can be denoted by the operator $\lfloor \cdot \rfloor$, i.e., $\lfloor x \rfloor = n$.

The interval [n,n+1) is interpreted as a zone of indifference in the input alphabet since the truncation algorithm in the PP, upon receiving any input from the interval, will yield the same integer n as its output. (See Fig. 4.4)

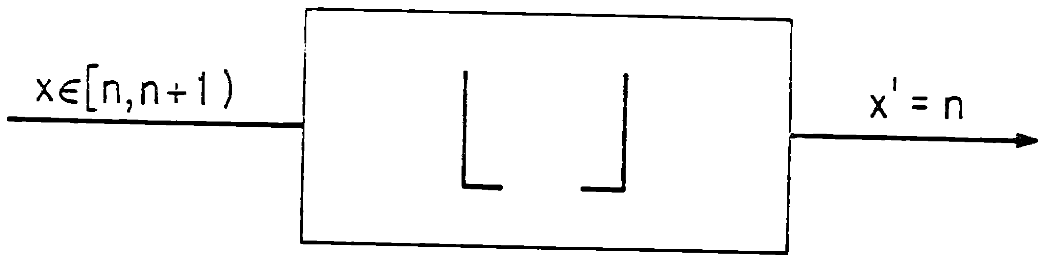

Figure 4.4. The Truncation Algorithm

To illustrate how the input entropy is reduced by such an algorithm, consider an input alphabet X consisting of 2n equally probable elements, in the form of

$$X = \{0, 0.5,\ 1,\ 1.5,\ 2,\ 2.5, \ldots, n-1,\ n-0.5\}$$

After the truncation operation in the PP, the output alphabet X' of the PP, which is the new input alphabet to the DM, consists of n equally probable elements in the form of

$$X' = \{0,\ 1,\ 2, \ldots, n-1\} \tag{4.12}$$

The entropy of the original alphabet, is

$$H(x) = -\sum_{x} p(x)\ \log_2 p(x) = -\ 2n \left[\frac{1}{2n}\ \log_2 \frac{1}{2n}\right] = \log_2 2n$$

$$= \log_2 n + 1 \tag{4.13}$$

while the entropy of the second alphabet is

$$H(x') = -\ n \left[\frac{1}{n}\ \log_2 \frac{1}{n}\right] = \log_2 n \tag{4.14}$$

Clearly, the entropy of the input alphabet is reduced by the truncation operation.

In general, if the input alphabet consists of nt equally probable elements (where n and t are both positive integers), then

$$X = \{0, \frac{1}{t}, \frac{2}{t}, \frac{3}{t}, \ldots, \frac{t-1}{t}, 1, 1+\frac{1}{t}, 1+\frac{2}{t}, \ldots, n-\frac{1}{t}\}.$$

After the truncation operation in the PP, the new input alphabet X' will be in the same form as in Eq. (4.12).

The entropy of the original input alphabet X,

$$H(x) = -\,nt\left[\frac{1}{nt}\log_2\frac{1}{nt}\right] = \log_2 n + \log_2 t \tag{4.15}$$

From Eqs. (4.14) and (4.15), it follows that the truncation operation reduces the input entropy by the amount $\log_2 t$, i.e.,

$$H(x) = H(x') + \log_2 t \tag{4.16}$$

where X and X' are the original and the new input alphabets to the DM respectively, and t is the number of intervals within two consecutive integers defined in the original input alphabet X.

The function of the PP in this example is to perform the aggregation task within the SA algorithms of the DM and thus reduce the workload of the DM In order to evaluate the utility of such a PP, two different DM systems are given, as shown Figs. 4.5 and 4.6, to do the same task.

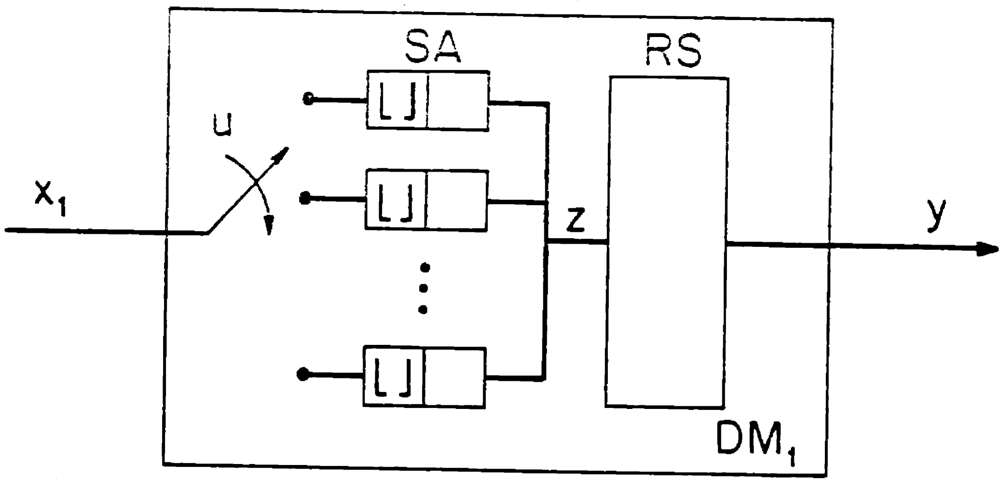

Figure 4.5. Basic DM System

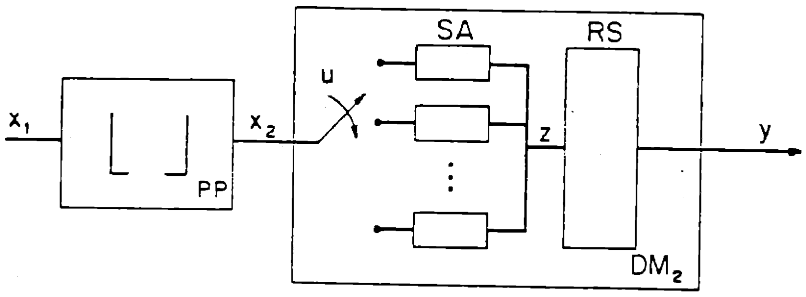

Figure 4.6. Aided DM System

As in Section 4.2, the first system consists of a DM only while the second system has a DM aided by a preprocessor. The decisionmakers in both systems have identical internal structure except that each of the SA algorithms in the first system has a truncation operation whereas that in the second system does not. The truncation operation in the second system is done by a PP, which consists of a single truncation algorithm. When identical decision strategies are employed by the two systems, both will perform the same task equally well. It follows from the discussion on entropies, that the presence of such a PP in the second system renders the entropy of the input to its DM to be less than that to the DM in the first system, i.e., $H(x_1) > H(x_2)$. To understand how the PP affects the processing load of each DM as well as the activity of the whole system, it is necessary to investigate the activity expression for the DMs of the two systems. The general activity expression of a DM was given in Chapter 2 as

$$G_{DM} = H(x) + H(u) + \sum_{i=1}^{U} p_i g_c^i + \sum_{i=1}^{U} \alpha_i H(p_i) + H(z) + H_z(v)$$

$$+ \sum_{j=1}^{V} p_j g_c^{U+j}(p(z|v=j)) + \sum_{j=1}^{V} \alpha_j' H(p_j) + H(y) + H(z) \tag{4.17}$$

Since the two processes have identical internal structures for the RS stage, the number of internal variables in the RS algorithms, α_j', $j = U+1, U+2,\dots,U+V$, will be the same. Furthermore, when deterministic algorithms are employed, the same task will be executed by the identical decision strategy in the two systems with the same performance. These facts imply identical α_j', $p(u)$, $p(v|z)$ and $p(y)$ in the two processes. If the distributions $p(z)$ in both processes are assumed to be identical, then a large part of the G_{DM} expression in Eq. (4.17) will have the same value, G_o, where

$$G_o = H(u) + H(z) + H_z(v) + \sum_{j=1}^{V} p_j g_c^{U+j}(p(z|v=j))$$

$$+ \sum_{j=1}^{V} \alpha_j' H(p_j) + H(y) + H(z) \tag{4.18}$$

Hence, the expressions for the workload of each DM are

$$G_{DM_1} = H(x_1) + \sum_{i=1}^{U} p_i g_{c_1}^{i} + \sum_{i=1}^{U} \alpha_{i_1} H(p_i) + G_o \tag{4.19}$$

$$G_{DM_2} = H(x_2) + \sum_{i=1}^{U} p_i g_{c_2}^{i} + \sum_{i=1}^{U} \alpha_{i_2} H(p_i) + G_o \tag{4.20}$$

where the subscripted variables of x, g_c^i and α_i refer to the different input distributions and SA algorithms of the two DMs.

If the input alphabet modelling the task is finite and evenly distributed with t intervals between two consecutive integers, then the following relation between the two input alphabets holds according to (Eq. (4.16)

$$H(x_1) = H(x_2) + \log_2 t \tag{4.21}$$

To compare the coordination terms in both systems, a typical SA algorithm is chosen from each DM for analysis, say $f_i^1(\cdot)$ and $f_i^2(\cdot)$, respectively. The algorithm $f_i^1(\cdot)$ receives input x_1 and yields output z while the algorithm $f_i^2(\cdot)$ receives input x_2 and yields output z. Since each of the SA algorithms in the first system has a truncation operation, when the algorithm $f_i^1(\cdot)$ processes its input x_1, its truncation operation always yields an internal variable within the algorithm $f_i^1(\cdot)$ which has the same distribution as x_2, the input variable of the corresponding algorithms $f_i^2(\cdot)$ in the second system. Hence the algorithm $f_i^1(\cdot)$ can be viewed as consisting of two subsystems. The first, the truncation operation, has internal coordination $g_{\lfloor\rfloor}$ and $\alpha_{\lfloor\rfloor}$ internal variables. The other is just identical to the algorithm $f_i^2(\cdot)$ in the second system. (See Fig. 4.7)

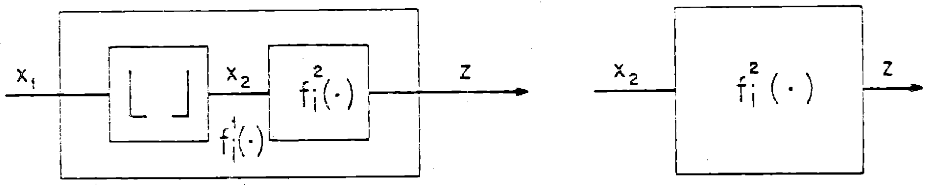

Figure 4.7. Typical algorithms from the Two DM Systems

By the decomposition property of coordination, the internal coordination $g^i_{c_1}$ of the algorithm $f^1_i(\cdot)$ can be expressed as follows:

$$g^i_{c_1} = g_{\lfloor\rfloor} + g^i_{c_2} + H(x_2) \tag{4.22}$$

where $g^i_{c_2}$ is the internal coordination of the algorithm $f^2_i(\cdot)$. The last term of Eq. (4.22), $H(x_2)$, is due to the intersystem coordination (assuming that $f^2_i(\cdot)$ is a deterministic algorithm).

As the algorithm $f^1_i(\cdot)$ is made up of two subsystems as shown in Fig. 4.7, the number of internal variables α_{i_1} within the algorithm $f^1_i(\cdot)$ is the sum of the internal variables within each individual subsystem

$$\alpha_{i_1} = \alpha_{\lfloor\rfloor} + \alpha_{i_2} \tag{4.23}$$

where α_{i_2} is the number of internal variables of the algorithm $f^2_i(\cdot)$.

The following inequalities are direct results of Eqs. (4.21) to (4.23).

$$H(x_1) > H(x_2) \tag{4.24}$$

$$g^i_{c_1} > g^i_{c_2} \qquad (4.25)$$

$$\alpha_{i_1} > \alpha_{i_2} \qquad (4.26)$$

From Eqs. (4.19) and (4.20) and the above inequalities, it can be concluded that the workload of the first DM, the unaided one, is greater than that of the second one, i.e.,

$$G_{DM_1} > G_{DM_2} \qquad (4.27)$$

The utility of the PP is thus shown since it reduces the processing workload of the second DM.

Consider now the workload of the two systems. The equations (4.1) and (4.2) of the example in Section 4.2 also hold for this example.

$$\text{Activity within First System} = G_{DM_1} \qquad (4.28)$$

$$\text{Activity within Second System} = G_{DM_2} + G_{PP} \qquad (4.29)$$

Since the PP in the second system is a truncation algorithm with input x_1, the activity within the PP is simply

$$G_{PP} = H(x_1) + g_{\lfloor\rfloor} \qquad (4.30)$$

where $g_{\lfloor\rfloor}$ is the internal coordination of the truncation algorithm and has the same value as the internal coordination of the subsystem responsible for the truncation operation within every SA algorithm of the first system.

To investigate the relationship between the workloads of the two systems, substitute Eqs. (4.22) and (4.23) into Eq. (4.19).

$$G_{DM_1} = H(x_1) + \sum_{i=1}^{U} p_i(g_{\lfloor\rfloor} + g^i_{c_2} + H(x_2)) + \sum_{i=1}^{U} (\alpha_{\lfloor\rfloor} + \alpha_{i_2})H(p_i) + G_o$$

$$= [H(x_1) + g_{\lfloor\rfloor}] + \left[H(x_2) + \sum_{i=1}^{U} p_i g^i_{c_2} + \sum_{i=1}^{U} \alpha_{i_2} H(p_i) + G_o\right]$$

$$+ \alpha_{\lfloor\rfloor} \sum_{i=1}^{U} H(p_i) \tag{4.31}$$

According to Eqs. (4.30) and (4.20), the first and second brackets of Eq. (4.31) are G_{PP} and G_{DM_2} respectively. Therefore, Eq. (4.31) becomes

$$G_{DM_1} = G_{PP} + G_{DM_2} + \alpha_{\lfloor\rfloor} \sum_{i=1}^{U} H(p_i) \tag{4.32}$$

If $\alpha_{\lfloor\rfloor} > 0$ and a nondeterministic decision strategy is assumed, i.e., $0 < p_i < 1$, for some i, then the last term of Eq. (4.32) is strictly positive. It can be concluded from Eq. (4.32) that

$$G_{DM_1} > G_{PP} + G_{DM_2} \tag{4.33}$$

The inequality in (4.33) states that the total processing workload of the first system is greater than that of the second system. The difference of the activities between the two systems, as revealed in the last term of Eq. (4.32) is reasonable because moving the truncation operation out from every

SA algorithm within the DM to the PP outside the DM will definitely reduce the workload due to each algorithm in the SA stage. In other words, the DM in the second system no longer needs extra resources to reinitialize the variables required for the truncation operation in each SA algorithm.

In this example, the introduction of the PP in the DM system not only reduces the workload of the DM but also reduces the activity within the whole DM system. In general, the kind of preprocessors that satisfy the inequalities Eqs. (4.27) and (4.33) can be implemented as external decision aids. They also suggest an internal structure within the SA stage of a DM that leads to reduced workload.

4.4 CONCLUSION

The two examples analyzed in the previous sections lead to the following remarks. The utility of a PP in a DM system can be evaluated by:

(1) Investigating the presence of any improvement in system performance while the DM's workload remains unchanged.

(2) Investigating any reduction in the DM's workload when the system performance remains the same.

In the two examples, when the system is evaluated to demonstrate the utility of the respective PP, the internal structures (i.e., the internal variables and their interconnections) of the PP's are never defined. The only constraint required is the inequality of the input and output entropies of the preprocessor i.e., $H(x_{in}) \geq H(x_{out})$. Such a condition can be made possible by satisfying one of the following three requirements:

(1) The preprocessor contains deterministic algorithms

(2) No stochastic variables exist in the PP

(3) No decisionmaking capability is allowed in the PP.

Therefore, the evaluation of the effectiveness of the two DM systems is totally independent of the implementation or modelling of the PP. One may try to generalize along this line to determine the system effectiveness of any DM system with a PP given that $H(x_{in}) > H(x_{out})$ to be a condition of the PP.

The role of a PP played in a DM system depends on how the boundaries of the system are drawn which in turn is determined by how much the PP can reduce the workload in the system. If

$$G_{DM_1} > G_{DM_2}$$

and

$$G_{DM_1} < G_{DM_2} + G_{PP}$$

then the boundaries of the DM system should not enclose the PP because the inclusion will increase the system activity. Consequently, the PP can function as a decision aid but not as a subsystem. On the other hand, if

$$G_{DM_1} > G_{DM_2}$$

and

$$G_{DM_1} > G_{DM_2} + G_{PP}$$

then the boundaries of the DM system can enclose the PP since including the PP still reduces the system activity. So in this case, the PP can function as a subsystem as well as a decision aid. (The system activity is further reduced when the PP functions as a decision aid).

The example in the last section justifies the need of having a PP as a subsystem in a DM system; i.e., it indicates that redefinition of the algorithms or procedures in such a way that identical processing is done in one place reduces the overall workload.

CHAPTER 5

DUAL-TASK PROBLEM

5.1 INTRODUCTION

Another application of the preprocessor is to facilitate a decisionmaker in handling dual-task processing. Depending on how the dual-task problem is formulated, the system performance of the dual-task case can be compared with that of single-task case when the measure of performance is properly defined. The difference of performance may account for the presence of switching between the two tasks during the execution of the dual-task.

Two variants of the dual-task problem will be presented in this chapter. One is the sequential dual-task problem in which inputs of two different tasks arrive one after another in a sequential manner. The other is the parallel dual-task problem in which inputs of the two tasks arrive in parallel in a synchronous manner. The two tasks in both variants are non-synergistic, i.e., they are dissimilar and do not reinforce each other, and so their performance may decline from what it should be in the single-task case. For the single-task case, the input arrival rate will be identical to the rate of the dual-task case in order to compare properly the levels of workload in the performance-workload space.

In the subsequent sections, models of the two dual-task problems will be formulated in the context of a single decisionmaker aided by a preprocessor and then analyzed to determine the effect that executing two non-synergistic tasks can have on system performance. In the process of demonstrating dual-task behavior, various graphs of single-task performance and activity or workload against several parameters are required to be drawn. All of these will be used in constructing the performance curves for single-task as well as dual-task processing.

5.2 SEQUENTIAL DUAL-TASK PROBLEM

The model of the sequential dual-task processing shown in Fig. 5.1 consists of a preprocessor and a decisionmaker.

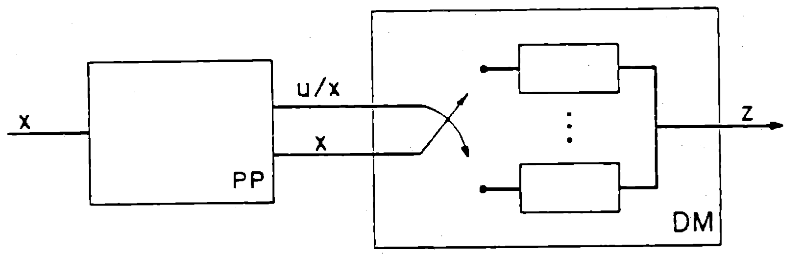

Figure 5.1. Sequential Dual-Task Processing Model

The input x of the system may be from task A with input alphabet X_A, or task B with input alphabet X_B. The PP examines each individual input to distinguish the task to which the particular input belongs. If it is a task A input, then the PP will yield a decision strategy u_A pertaining to task A to the DM for processing the input; otherwise strategy u_B pertaining to task B will be genrated. Essentially, the preprocessor acts as a matching algorithm yielding decision strategies as described in Chapter 3.

The decision strategies are the variables determining system activity and system performance. A pure or deterministic strategy means that a specific algorithm is always chosen to process the inputs from that particular task. A mixed or stochastic strategy is one obtained as a convex combination of pure strategies. In order to simplify the analysis that follows, which demonstrates how the system performance is affected by dual-tasking, it is assumed that there are only two pure strategies available for each individual task. When the four pure strategies of the two tasks are denoted by u_A^1, u_A^2, u_B^1, u_B^2, the mixed strategies u_A and u_B of the two tasks can be written as functions of decision strategy parameters δ_A or δ_B which correspond to the probabilities of employing pure strategies u_A^1 and u_B^1 respectively.

$$u_A = \delta_A u_A^1 + (1-\delta_A) u_A^2 \qquad 0 \leq \delta_A \leq 1 \tag{5.1}$$

$$u_B = \delta_B u_B^1 + (1-\delta_B) u_B^2 \qquad 0 \leq \delta_B \leq 1 \tag{5.2}$$

Again, for the sake of simplicity, only the decision strategies of the SA stage are considered. The situation assessment variable z is the system output.

Given that z' is the desired decision response, the performance of a decisionmaking task can be defined as the probability of error in determining z which is presented by J. Since two tasks are being performed, their measures of performance J_A and J_B are defined as the probabilities of error in executing task A and task B respectively.

$$J_A = p(z \neq z') | x \varepsilon X_A) \tag{5.3}$$

$$J_B = p(z \neq z' | x \varepsilon X_B) \tag{5.4}$$

If the performance measure evaluated when a pure strategy is in effect is denoted by J^1 and J^2, respectively, then by the linearity of probability, the quantities J_A and J_B can be rewritten as functions of the parameters δ_A and δ_B that specify the decision strategies $u_A$and u_B as in Eqs. (5.1) and (5.2):

$$J_A(\delta_A) = \delta_A J_A^1 + (1-\delta_A) J_A^2 \tag{5.5}$$

$$J_B(\delta_B) = \delta_B J_B^1 + (1-\delta_B) J_B^2 \tag{5.6}$$

Typical plots of performance versus the parameter δ of individual tasks may look as shown in Fig. 5.2.

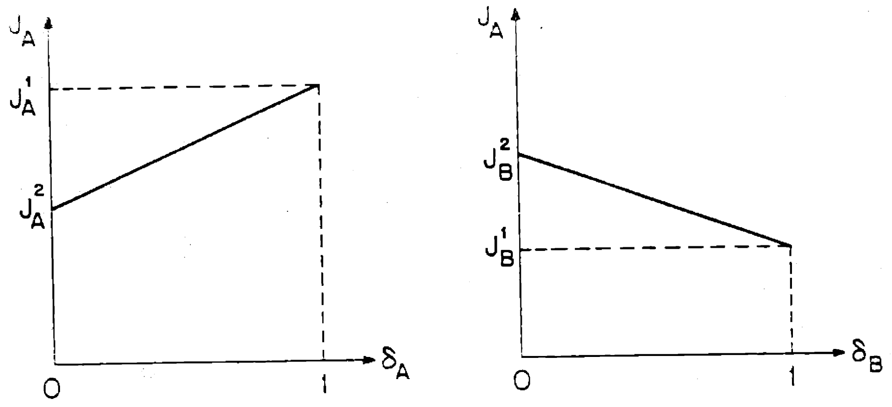

Figure 5.2. Performance Versus Decision Strategy Parameter δ For Each Task

Now the measure of overall performance for the system can be defined by the probability of making a task A error or a task B error. If α is the probability of the system processing task A inputs, then by assuming errors on both tasks to be equally detrimental, and by the linearity of probability the system performance J can be written as

$$J(\alpha) = \alpha\, J_A(\delta_A) + (1-\alpha) J_B(\delta_B) \qquad 0 \le \alpha \le 1 \qquad (5.7)$$

Graphically, for a fixed α, the system performance J can be plotted as a tilted plane with boundaries at the planes $\delta_A=0$ and 1 and $\delta_B=0$ and 1. The plane in the 3-dimensional space (J,δ_A,δ_B) is shown in Fig. 5.3.

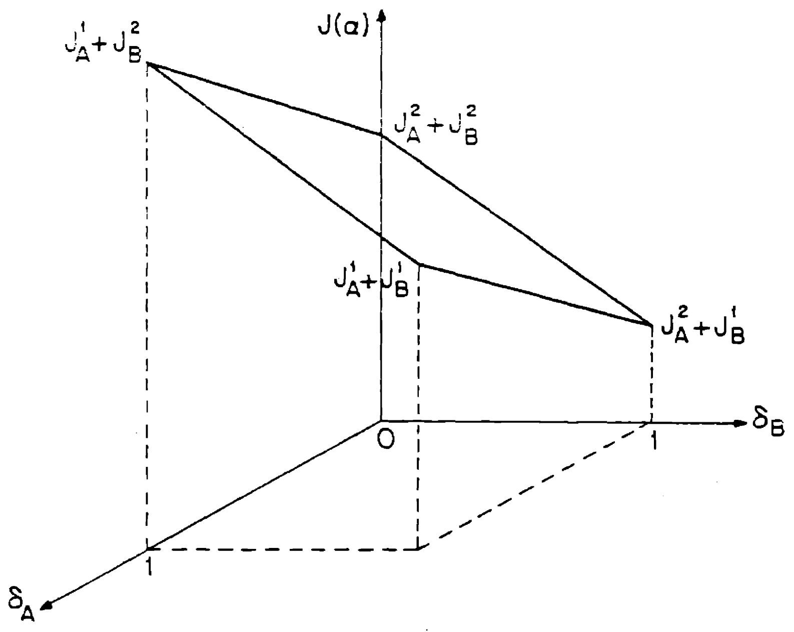

Figure 5.3. Total Performance Versus Decision Strategy Parameters δ_A and δ_B

The total activity of the system for single-task processing is a convex function of the decision strategy parameters δ_A and δ_B, whereas that of dual-task processing is a convex function of the task division parameter α as well as the decision strategy parameters δ_A and δ_B. The covexity of the system activity G in α as well as in δ_A and δ_B had been demonstrated [10], i.e.,

$$G_A(\delta_A) \geq \delta_A G_A^1 + (1-\delta_A)G_A^2 \tag{5.8}$$

$$G_B(\delta_B) \geq \delta_B G_B^1 + (1-\delta_B)G_B^2 \tag{5.9}$$

$$G(\alpha) \geq \alpha G_A(\delta_A) + (1-\alpha)G_B(\delta_B) \tag{5.10}$$

Typical graphs of G_A, G_B and G for a fixed α are shown in Figs. 5.4 and 5.5.

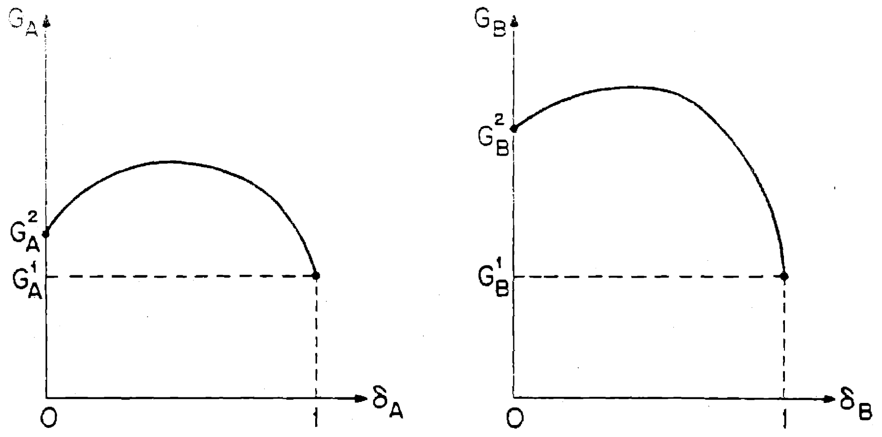

Figure 5.4. Workload Versus Decision Strategy Parameter δ For Each Task

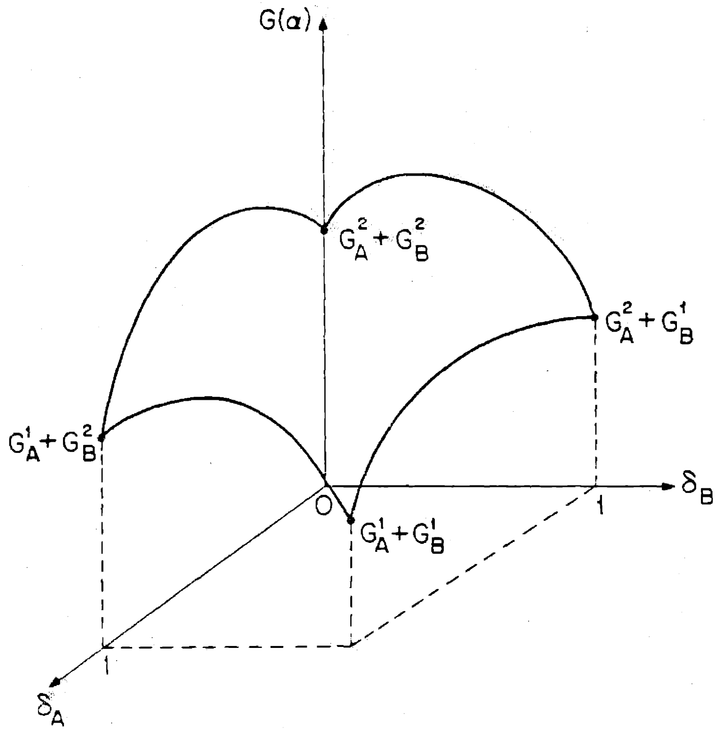

Figure 5.5. Total Workload Versus Decision Strategy Parameters δ_A and δ_B

The system activity or workload in Fig. 5.5 is a curved bounded surface with four corners (somewhat like a tent). When, for a fixed α, J is ploted against G, parametrically with respect to (δ_A,δ_B) the performance-workload locus for the whole system can be constructed.

To simplify the construction of the locus, but without any loss of generality, it is assumed that $\delta_A = \delta_B = \delta$, i.e., both task A and task B employ the same rule in decisionmaking (but the content of the decisions may be different). For example, if there are four algorithms f_1, f_2, f_3, and f_4 in the DM, the pure strategies of the two tasks can be defined as employing different algorithms. For instance, for task A: u_A^1 means u=1; u_A^2 means u=2; while for task B: u_B^1 means u=3 and u_B^2 means u=4. Then the mixed strategies, u_A and u_B, will have the same form but different content in the decisionmaking process.

$$u_A = \delta\, u_A^1 + (1-\delta)u_A^2 \tag{5.11}$$

$$u_B = \delta\, u_B^1 + (1-\delta)u_B^2 \tag{5.12}$$

With such a formulation, Fig. 5.3 and Fig. 5.5 will degenerate into the 2-dimensional plots shown in Figs. 5.6 and 5.7.

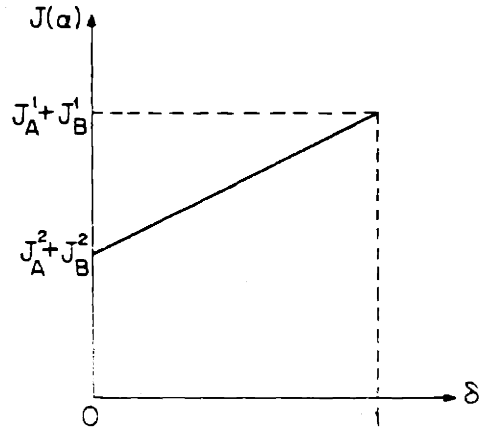

Figure 5.6. Performance Versus Decision Strategy Parameter δ

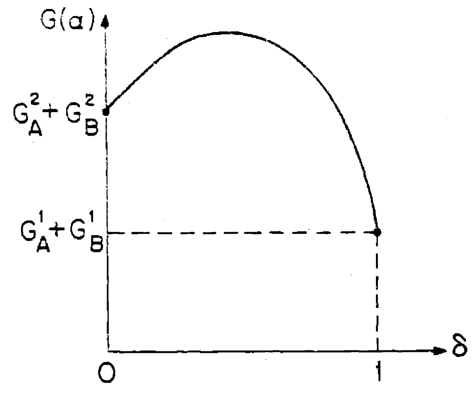

Figure 5.7. Workload Versus Decision Strategy Parameter δ

Therefore, the J–G plot of the system for a fixed α can be drawn parametrically with respect to δ (Fig. 5.8).

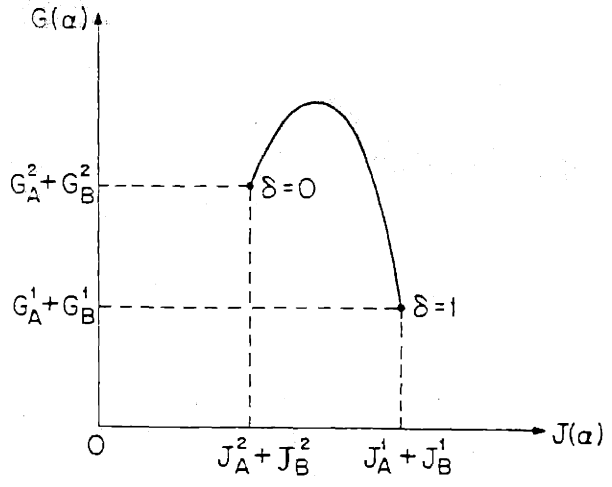

Figure 5.8. System Workload Performance Locus

If for all the different ways of task division, i.e., $0 \leq \alpha \leq 1$, the assumption $\delta_A = \delta_B = \delta$ is always kept, then similar J–G plots like Fig. 5.8 can be drawn for all values of α. To display all these J–G plots in such a format that the performance of single-task processing and dual-task processing can be compared, G can be plotted in a polar coordinate plane with its radial coordinate being the system performance J and the angular coordinate being proportional to the task division parameter α, (Fig. 5.9) i.e.,

$$\theta \varepsilon [0, \frac{\pi}{2}] \quad \alpha \varepsilon [0,1]$$

where

$$\alpha = \frac{\theta}{\pi/2}.$$

Therefore, a typical J-G plot of the system for all possible values of α may look like the curved surface in Fig. 5.10 which resembles part of the surface area of a toroid with curved edges. Its edges at $\delta=0$ and $\delta=1$ are curved upward with respect to G because for a particular decision strategy, the system activity G is a convex function of the task division parameter α. The other two edges of the curved surface in Fig. 5.10 are the J-G plots for the single task processing. The projection of the surface will be a curved area in the shape of a quarter toroid as shown in Fig. 5.11 on a J-α plane.

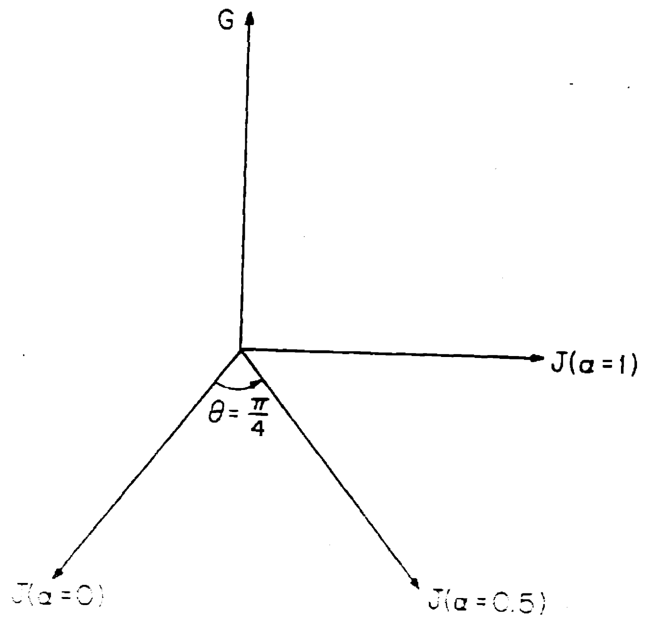

Figure 5.9. Coordinate System for Plotting J-G Curves of Varying α

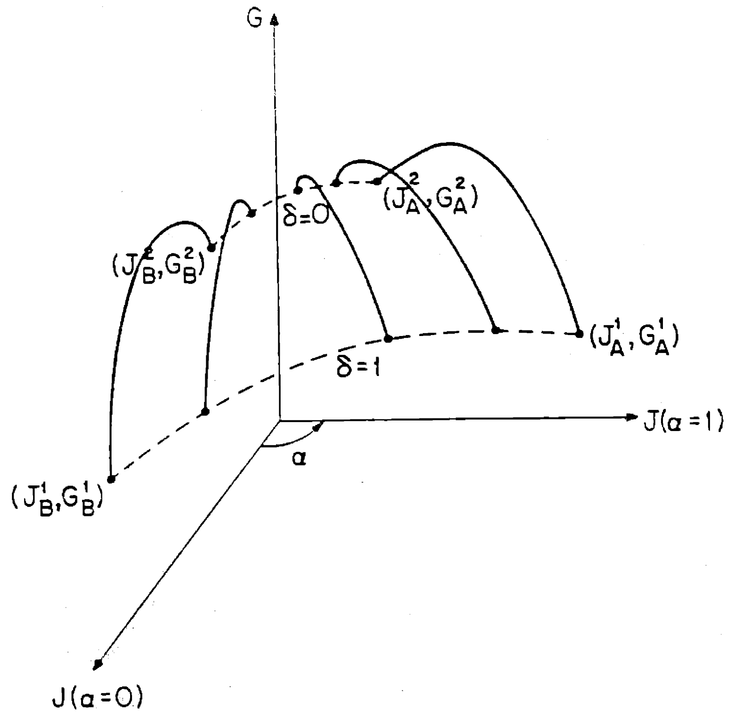

Figure 5.10. System Workload Performance Loci of Changing Task Division Parameter α.

When the system exhibits bounded rationality behavior, the plane of the activity threshold G may cut through the curved surface in Fig. 5.10. The surface above the cut corresponds to the region where the workload exceeds the bounded rationality constraint. The remaining surface represents the region with admissible strategies. The projection of the remaining surface on the J-α plane will allow comparison of performance between single-task processing and dual-task processing. Depending on the numerical specifications of the problem, the projection of the region that results from admissible strategies may look like Fig. 5.12.

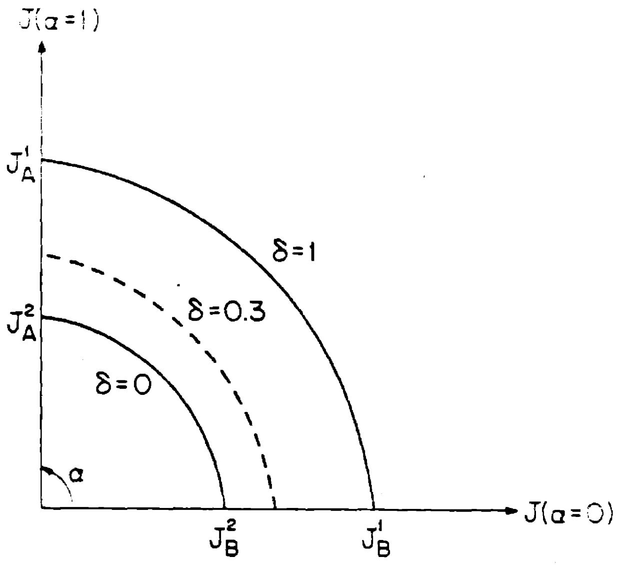

Figure 5.11. Projection of Fig. 5.10 on J-α Plane

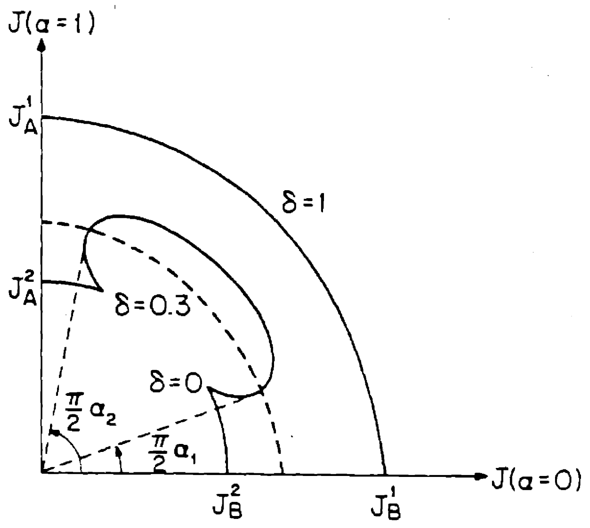

Figure 5.12. Projection of Locus Due to Admissible Strategies on the J-α plane.

The locus in Fig. 5.12 is obtained by considering a plane of constant G in Fig. 5.10 that is above the δ=1 boundary of the (G,J_A,J_B) surface and intersects the δ=0 boundary. This is illustrated in Fig. 5.13. For δ=0.3, a large central segment of the G–α curve exceeds the threshold level G_r. Because of the convexity of the G–α curves, the eliminated segment is defined by the interval (α_1, α_2). Similarly, a segment of the $\delta=0$ curve, the $\delta=0.6$ curve, and the ones for intermediate values of δ, are eliminated. This results in the shape shown in Fig. 5.12.

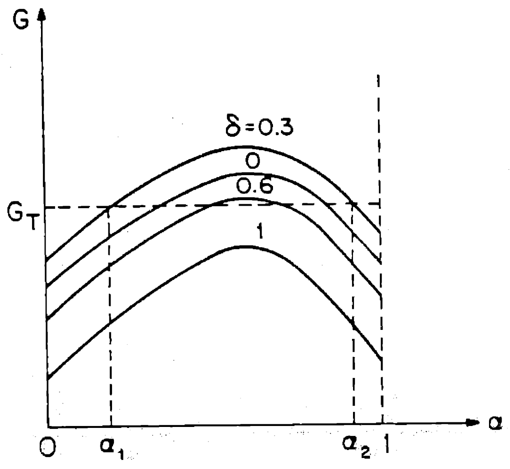

Figure 5.13. G–α curves for various values of δ

It can be observed in Fig. 5.12 that under the constraint of bounded rationality ($G \leq G_r$), the decisionmaker doing a single, or almost a single task, achieves a wider range of performance values, including the optimum ones. This is evidenced by the locus near the axes $\alpha=0$ and $\alpha=1$. When both tasks have to be carried out, the range of values of the performance is smaller; many of the better values are not achievable. Such a deficiency can be explained by the need for extra time and energy for a decisionmaker to adjust in handling a different task. Hence, in general, a decisionmaker performs better in processing a single task rather than processing two different tasks in sequence, even through the incoming task rate for both cases are the same.

5.3 PARALLEL DUAL-TASK PROBLEM

The model of parallel dual-task processing is similar to that of the sequential dual-task processing except that it has two input ports where the input elements from task A and task B are arriving simultaneously in a synchronous manner (see Fig. 5.14).

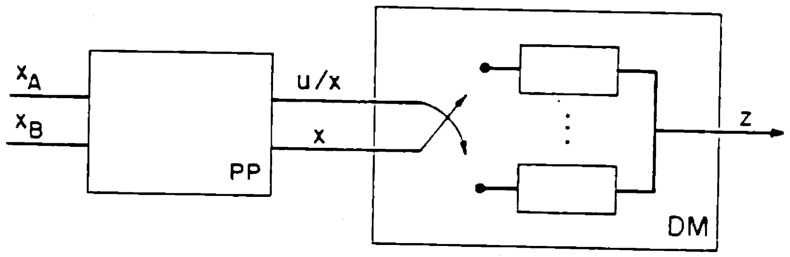

Figure 5.14. Parallel Dual-Task Processing Model

It is assumed that a human decisionmaker can handle only one task at a time even though two inputs, two different tasks, arrive at the same time. Naturally, he will select the input of the highest priority to process, while the other input data are either stored for later processing when resources are available, or simply ignored. So, the parallel task problem considered is a psuedo-sequential task problem, since tasks are still processed one after another rather than simultaneously. Consequently, in order to transform a parallel task problem into a sequential one, a properly designed preprocessor is needed. Depending on how the PP handles the parallel tasks, different versions of the parallel dual-task problem result. In the one defined here, no buffer is available in the PP to store the unprocessed task input. The decisionmaker selects one of the two tasks and ignores the other one.

If the two inputs ports always receive inputs from task A and task B respectively, and it is assumed that task A always has a higher priority than task B, then the internal structure of the PP model is such that task A inputs always get processed and task B inputs are ignored. This is a trivial case that deserves no further discussion.

To allow processing of the lower priority task, input B, null variables ϕ are added to the input alphabet X_A of task A where ϕ does not need to be processed at all. In other words, task B will get processed when the task A input is a null variable to the PP. This dual-task formulation has been motivated by an experiment conducted at the Navy Personnel Research and Development Center [11], [12]. An air defense task and a communication task were performed concurrently, with the former task having a higher priority than the latter. The result of the experiment was that the performance of the air defense task was not affected by the presence of the communication task, whereas the performance of the communication task was degraded under the dual-task conditions. Moreover, the performance decrement in the communication task was linearly related to the track load in the air defense task. These phenonmena may be explained by formulating the parallel dual-task problem as follows.

Of the two input alphabets, X_A and X_B, only the alphabet of the higher priority task, X_A in this case, possesses the null variable ϕ. So there are altogether two forms of synchronous parallel inputs to the PP: $x_A \neq \phi, x_B \varepsilon X_B$ and $x_A = \phi$, $x_B \varepsilon X_B$, where x_A and x_B are elements of X_A and X_B respectively. For each type of synchronous parallel inputs the PP receives, a definite output x will be generated by the PP for the decisionmaker to process:

$$\text{If} \quad x_A \neq \phi \quad \text{then} \quad x = x_A$$

$$\text{If} \quad x_A = \phi \quad \text{then} \quad x = x_B$$

In addition to the output x, a decision strategy u_A and u_B will be generated by the PP for processing the output of x_A or x_B, respectively. When the PP functions as described, its internal variables and their interconnections can be structured as shown in Fig. 5.15.

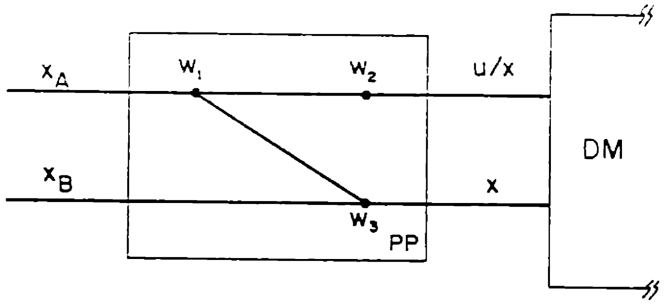

Figure 5.15 A Typical Parallel Dual-Task Preprocessor

Here, w_1 is a boolean variable whose function is to identify whether the value of x_A is a non-null variable or not. If γ is the frequency of x_A not being null, then w_1 can be defined as follows:

$$w_1 = \begin{cases} \text{true} & \text{w.p. } \gamma & \text{if } x_A \neq \phi \\ \text{fagse} & \text{w.p. } 1-\gamma & \text{if } x_A = \phi \end{cases}$$

Both w_2 and w_3 are the output variables u and x, respectively; their values depend on whether x_A is a non-null variable, or whether w_1 is true nor not. So they can be defined in a similar way.

$$w_2 = \begin{cases} u_A & \text{w.p. } \gamma & \text{if } w_1 = \text{true} \\ u_B & \text{w.p. } 1-\gamma & \text{if } w_1 = \text{false} \end{cases}$$

$$w_3 = \begin{cases} x_A & \text{w.p. } \gamma & \text{if } w_1 = \text{true} \\ x_B & \text{w.p. } 1-\gamma & \text{if } w_1 = \text{false} \end{cases}$$

From the definitions of the internal variables and their interconnections it is very clear that the three variables in the PP will have the same entropy, denoted by $H(\gamma)$. Thus the total activity of the PP is $3H(\gamma)$.

Again, as in the previous section, given that there are only two pure strategies available for each individual task, the mixed decision strategies of each task , u_A and u_B, can be rewritten as linear combinations of the pure strategies with decision strategy parameters δ_A and δ_B, respectively.

Since not all the inputs of the two parallel tasks are processed in the decisionmaker, the task division parameter for dual-task processing has to be re-interpreted. In this case, it is the proportion of x_A's and x_B's in the processed input x. In transforming parallel dual-task processing into equivalent sequential processing through the use of a preprocessor, only a fraction γ of task A input data and a fraction $(1-\gamma)$ of task B input data are sent to the decisionmaker to be processed. Hence, the task division parameter for the decisionmaker's processed input is γ, the frequency of non-null task A inputs to the preprocessor.

In order to compare the individual task performance as the task input distribution changes, J-δ plots of the two tasks have to be drawn for various values of γ. Since the inputs from the two tasks arrive at the PP simultaneously and not all the input data are processed by the decisionmaker, the measures of performance J_A and J_B for task A and task B respectively have to be defined precisely. Given x_A' and x_B' to be the desired output responses of inputs x_A and x_B respectively, J_A and J_B can be defined as follows.

For task A, the desired response x_A' can only apply to non-null input elements since all the null elements ϕ are retained within the PP and thus their desired responses are undefined. As a result, only non-null variables are significant in defining the performance measure of task A. Therefore,

$$J_A = p(x_A \neq x_A' | x_A \neq \phi) \tag{5.13}$$

For task B, the desired response x_B' is well-defined for all the elements in alphabet X_B. Due to the task's lower priority status, the inputs x_B have to be retained within the PP when $x_A \neq \phi$, resulting in definite errors or cost for inputs x_B at these times. Mathematically

$$J_B = p(x_B \neq x_B') = p(x_A = \phi)\, p(x_B \neq x_B' | x_A = \phi)$$

$$+ p(x_A \neq \phi)\, p(x_B \neq x_B' | x_A \neq \phi)$$

Since $p(x_A = \phi) = 1-\gamma$; $p(x_A \neq \phi) = \gamma$ and $p(x_B \neq x_B' | x_A \neq \phi) = 1$

$$J_B = (1-\gamma)\, p(x_B \neq x_B' | x_A = \phi) + \gamma \qquad (5.14)$$

Rewriting Eq. (5.14) yields

$$J_B = p(x_B \neq x_B' | x_A = \phi) + \gamma\, p(x_B = x_B' | x_A = \phi) \qquad (5.15)$$

In accordance with expressions (5.13) and (5.15), the individual task performance J_A and J_B can be plotted against the task division parameter γ: these will be straight lines as shown in Fig. 5.16.

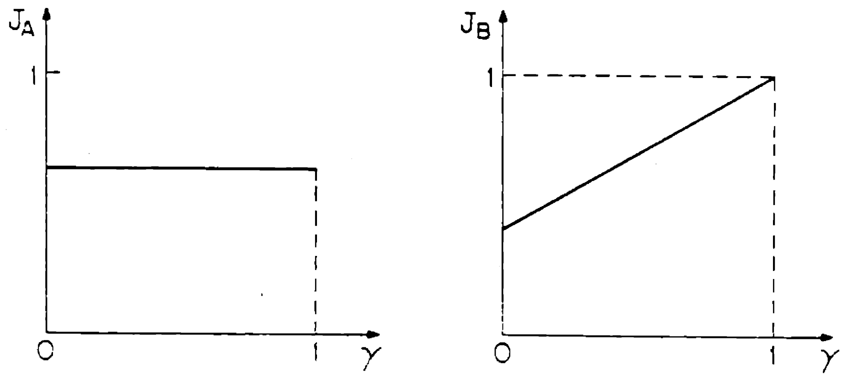

Figure 5.16. Performance Versus Parameter γ For Each Task

Given the two parameters δ_A and δ_B that specify the mixed decision strategies, it is known that the task performance J_A (or J_B) is independent of δ_B (or δ_A). Moreover, according to Eqs. (5.5) and (5.6), the relationships between J_A and δ_A, and between J_B and δ_B, are both linear. Hence, the J_A-δ_A plot and J_B-δ_B plot should both be straight lines as shown in Figure 5.2.

Finally, in order to investigate how the individual task performance varies with the task input distribution for all the decision strategies for task A, J_A is plotted against γ and δ_A in Fig. 5.17. The locus is a plane tilted in one direction.

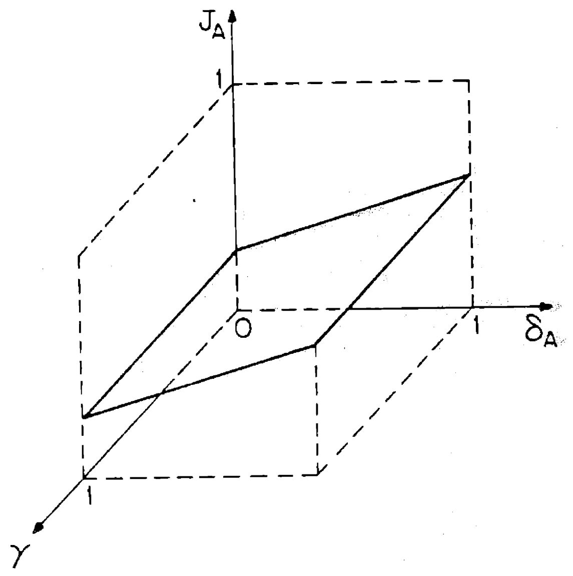

Figure 5.17. Task A Performance Versus Parameters γ And δ_A

A similar locus is drawn for task B in Fig. 5.18. It is a plane tilted in two directions.

The projections of the loci in Figs. 5.17 and 5.18, drawn on the J_A-γ plane and J_B-γ plane, respectively (Figs. 5.19 and 5.20) show how, for all the decision strategies the individual task performance varies with the task input distribution.

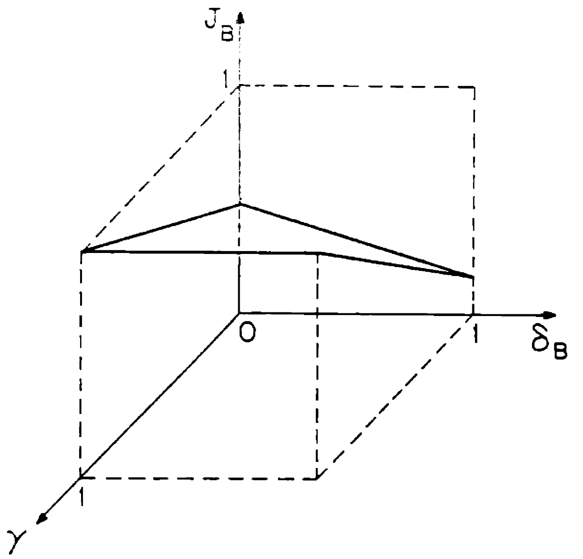

Figure 5.18. Task B Performance Versus Parameter γ And δ_B

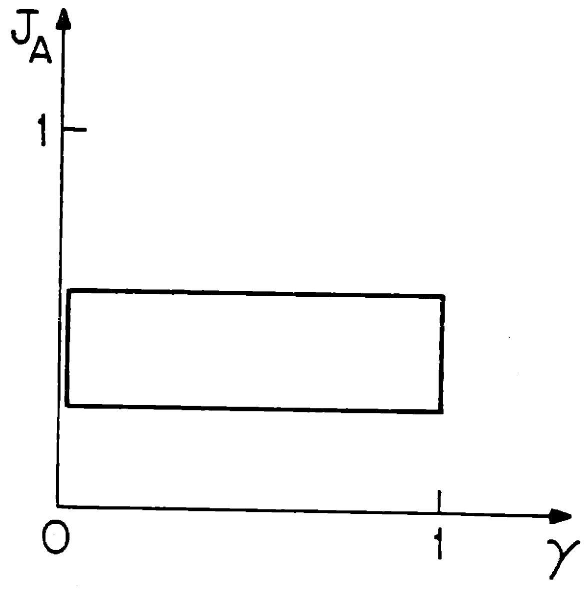

Figure 5.19. Task A Performance Versus Task Division Parameter

If task A and task B model the air defense task and the communication task, respectively mentioned at the beginning of this section, then in Fig. 5.19, the constant range of task A performance throughout the whole range of γ implies that the performance of the air defense task is unaffected by the presence of any amount of communication task. In the special case of $\gamma=0$, it is noted that all the task A inputs are null variables, and therefore, task A performance is undefined.

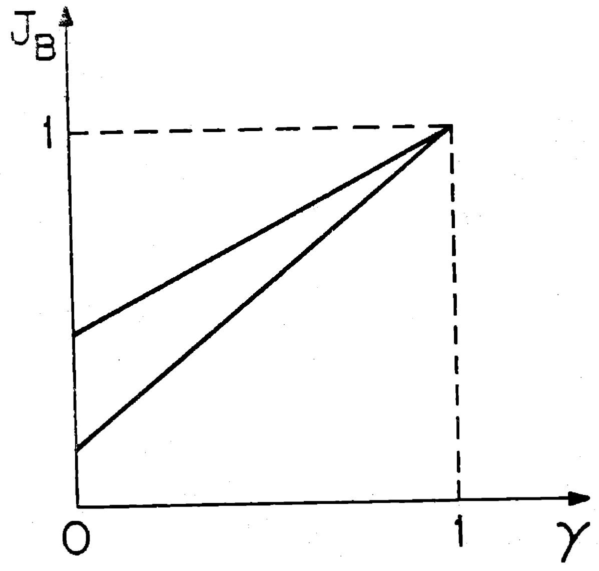

Figure 5.20. Task B Performance Versus Task Division Parameter

On the other hand, in Fig. 5.20, with increasing value of γ, the increasing value and decreasing range of task B performance imply that when the communication task occupies a smaller share between the two tasks, its performance degrades with a smaller variability. The value $J_B=1$ at $\gamma=1$ means that when all the task A inputs are non-null, all the task B inputs will be retained in the PP without any output response at all. Since γ is the fraction of non-null variables in the task A inputs and only non-null variables will be processed by the decisionmaker, γ can also be interpreted as the amount of processing load in the task A input channel. Hence, in Eq. (5.15), the presence of the coefficient γ in the J_B expression indicates that the performance decrement (i.e., the increase of J_B value) in the communication task is linearly related to the track load in the air defense task.

Basically, the above results justify the model of the parallel dual-task problem developed so far for the air defense task and communication task experiment [12].

CHAPTER 6

CONCLUSIONS

The main results developed in the thesis are summarized in this section. The basic preprocessor model employs a matching algorithm to partition the input alphabet so that its input elements can be processed by their respective desired decision strategies. The utility of the preprocessor in reducing the workload of a decisionmaker is found to depend on the input uncertainty, the decision strategies and the algorithmic structures in the preprocessor. The filtering preprocessor model upgrades a decisionmaking system with bounded rationality by withholding the irrelevant inputs and reducing the input rate to the system. As a result, more time will be available to the decisionmaker to process the relevant inputs. The latter phenomenon is illustrated by a numerical example.

One of the uses of a preprocessor is to reduce the processing load in the situation assessment stage of a decisionmaker by preprocessing the input data. Another use is to aggregate the input data, thus reducing the uncertainty in the input. In both cases, the workload of the decisionmaker is reduced.

A preprocessor can also be used to facilitate a decisionmaker in handling a dual-task. Such a preprocessor functions as a matching algorithm to yield appropriate decision strategies for processing two kinds of inputs. It has been shown that a decisionmaker carrying out a dual-task performs at best as well as when he executes a single task. This is due to the readjustment effort required by the decisionmaker to handle the different tasks. In the case of parallel dual-tasks, the preprocessor accepts two inputs of different task priority simultaneously and yields one input to the decisionmaker with its associated decision strategy. In this case, the individual task performance is investigated to verify the performance of the human decisionmaker in a combined air-defense communication task experiment.

CHAPTER 7

REFERENCES

[1] J. G. Wohl, "Force Management Decision Requirements for Air Force Tactical Command and Control," IEEE Trans. on Systems, Man and Cybernetics, SMC-11, No. 9, September 1981, pp. 618-639.

[2] D. Brick, Private Communication, 1981.

[3] K. L. Boettcher, "An Information Theoretic Model of the Decision Maker," S.M. Thesis, LIDS-TH-1096, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, June 1981.

[4] K. L. Boettcher and A. H. Levis, "Modeling the Interacting Decisionmaker with Bounded Rationality," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-12, No. 3, May/June 1982.

[5] C. E. Shannon and W. Weaver, The Mathematical Theory of Communication, University of Illinois, Urbana, IL, 1949.

[6] R. G. Gallager, Information Theory and Reliable Communication, John Wiley and Sons, New York, 1968.

[7] R. C. Conant, "Laws of Information Which Govern Systems," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-6, No. 4, April 1976.

[8] W. J. McGill, "Multivariate Information Transmission," Psychometrika, Vol. 19, No. 2, June 1954.

[9] R. F. Drenick, "Organization and Control," in Directions in Large Scale Systems, Y. C. Ho and S. K. Mitter, Eds., Plenum, New York, 1976.

[10] S. A. Hall, "Information Theoretic Models of Storage and Memory," S. M. Thesis, LIDS-TH-1232, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, September 1982.

[11] R. T. Kelley, F. L. Greitzer, and R. L. Hershman, "Air Defense: A Computer Game for Research in Human Performance," NPRDC-TR-81-15, Navy Personnel Research and Development Center, San Diego, CA, July 1981.

[12] R. T. Kelley, , F. L. Greitzer, "Effects of Track Load on Decision Performance in Simulated Command and Control Operations," NPRDC-TR-82-21, Navy Personnel Research and Development Center, San Diego, CA, January 1982.

APPENDIX A

DERIVATION OF BASIC PREPROCESSOR EXPRESSIONS

The analytic expressions of the noise, coordination and intersystem coordination for the basic preprocessor in Section 3.2 are evaluated in this appendix, using the information theory identities.

Noise

The noise G_n was defined in Eq. (3.3)

$$G_n = H_x(r_1,r_2,\ldots,r_k,s,\underline{u})$$

By repeatedly applying the identity

$$H(x,y) = H(x) + H_x(y) \tag{A.1}$$

G_n can be written as

$$G_n = H_x(r_1,r_2,\ldots,r_k) + H_{x,r_1,r_2,\ldots,r_k}(s) + H_{x,r_1,r_2,\ldots,r_k,s}(\underline{u}) \tag{A.2}$$

Since the variables $r_1,r_2,\ldots,r_k$ are all well-defined boolean functions of x, the first term of Eq. (A.2) is zero. The variable s is a well-defined function of $r_1,r_2,\ldots,r_k$, so the second term of Eq. (A.2) is zero again. As the value assumed by the variable $\underline{u}$ depends only on the value of s, the last term of Eq. (A.2) reduces to $H_s(\underline{u})$. Hence, Eq. (A.2) becomes

$$G_n = H_s(\underline{u})$$

By definition,

$$H_s(\underline{u}) = -\sum_s p(s) \sum_{\underline{u}} p(\underline{u}|s) \log p(\underline{u}|s) \tag{A.3}$$

The variables in the basic PP model are defined in such a way that, given an input x belonging to the input alphabet X,

$$s = i \quad \text{if and only if} \quad x \in X_i \quad \text{for some } i$$
$$\text{and } x \notin X_j, \quad i \neq j, \quad i,j=1,2,\ldots,k$$

Hence, Eq. (A.3) can be written as

$$H_s(\underline{u}) = -\sum_{i=1}^{k} p(x \in X_i) \sum_{\underline{u}} p(\underline{u}|x \in X_i) \log p(\underline{u}|x \in X_i) \tag{A.4}$$

Since the distribution $p(\underline{u}|x)$ is the same for all $x \in X_i$ and the union of all the partitions X_i's is the input alphabet X, Eq. (A.4) is equivalent to

$$H_s(\underline{u}) = -\sum_x p(x) \sum_{\underline{u}} p(\underline{u}|x) \log p(\underline{u}|x) = H_x(\underline{u})$$

Thus,

$$G_n = H_s(\underline{u}) = H_x(\underline{u})$$

Coordination

The coordination G_c was defined in Eq. (3.4),

$$G_c = T(r_1 : r_2 : \ldots : r_k : s : \underline{u})$$

According to the definition of n-dimensional coordination, G_c can be written as

$$G_c = \sum_{i=1}^{k} H(r_i) + H(s) + H(\underline{u}) - H(r_1, r_2, \ldots, r_k, s, \underline{u}) \qquad \text{(A.5)}$$

Consider the last term of Eq. (A.5). Using the identity in Eq. (A.1),

$$H(r_1, r_2, \ldots, r_k, s, \underline{u}) = H(r_1, r_2, \ldots, r_k) + H_{r_1, r_2, \ldots, r_k}(s)$$

$$+ H_{r_1, r_2, \ldots, r_k, s}(\underline{u}) \qquad \text{(A.6)}$$

The second and the third terms of Eq. (A.6) reduce to zero and $H_s(\underline{u})$, respectively, because of the presence of appropriate conditioning in the terms. Hence, Eq. (A.6) is simplified into

$$H(r_1, r_2, \ldots, r_k, s, \underline{u}) = H(r_1, r_2, \ldots, r_k) + H_s(\underline{u}) \qquad \text{(A.7)}$$

Substituting Eq. (A.7) into Eq. (A.5) yields

$$G_c = \sum_{i=1}^{k} H(r_i) + H(s) + H(\underline{u}) - H(r_1, r_2, \ldots, r_k) - H_s(\underline{u})$$

$$= H(s) + H(\underline{u}) - H_s(\underline{u}) + T(r_1 : r_2 : \ldots : r_k)$$

Intersystem Coordination

The intersystem coordination was defined in Eq. (3.5),

$$T(S^{PP}:S^{DM}) = H(S^{DM}) - H_{S^{PP}}(S^{DM}) \tag{A.8}$$

Since the preprocessor of the example is a deterministic algorithm, the input variable x determines every other variable in the algorithm. The second term of Eq. (A.8) is thus equivalent to $H_x(S^{DM})$; hence Eq. (A.8) becomes

$$\begin{aligned} T(S^{PP}:\ S^{DM}) &= H(S^{DM}) - H_x(S^{DM}) \\ &= H(x:S^{DM}) \\ &= H(x) - H_{S^{DM}}(x) \end{aligned} \tag{A.9}$$

If the input value x to the decisionmaker is duplicated by some variable within the DM model, then the second term of Eq. (A.9) reduces to zero. Hence, Eq. (A.9) becomes

$$T(S^{PP}:\ S^{DM}) = H(x)$$

APPENDIX B

DERIVATION OF FILTERING PREPROCESSOR EXPRESSIONS

This appendix presents the derivation of the throughput, noise and coordination expressions for the preprocessor model with filtering discussed in Section 3.3.

Throughput

The throughput G_t of this preprocessor is defined in Eq. (3.14) as

$$G_t = T(x:\underline{u},x_o) = H(\underline{u},x_o) - H_x(\underline{u},x_o) \tag{B.1}$$

Since the preprocessor with filtering is a deterministic algorithm, knowledge of its input x will determine all the variables within the algorithm. Therefore, the second term of Eq. (B.1) equates to zero. Thus, Eq. (B.1) becomes

$$G_t = H(\underline{u},x_o)$$

Noise

The noise G_n of the preprocessor in Eq. (3.15) is defined as

$$G_n = H_x(r_o,r_1,\ldots,r_k,x_o,s,\underline{u})$$

By applying the identity in Eq. (A.1) repeatedly, G_n can be written as

$$G_n = H_x(r_o,r_1,\ldots,r_k) + H_{x,r_o,\ldots,r_k}(x_o) + H_{x,r_o,\ldots,r_k,x_o}(s)$$

$$+ H_{x,r_o,\ldots,r_k,x_o,s}(\underline{u}) \qquad \text{(B.2)}$$

Due to the presence of appropriate conditioning in the first, second and third terms of Eq. (B.2), all of them are zero. The last term reduces to $H_s(\underline{u})$ because the value of $\underline{u}$ depends only on the value of s. Therefore, Eq. (B.2) is simplified into

$$G_n = H_s(\underline{u})$$

<u>Coordination</u>

The coordination G_c of the preprocessor, Eq. (3.16), is

$$G_c = T(r_o:r_1:\ldots:r_k:x_o:s:\underline{u})$$

According to the definition of n-dimensional coordination, G_c can be written as

$$G_c = \sum_{i=0}^{k} H(r_i) + H(x_o) + H(s) + H(\underline{u}) - H(r_o,r_1,\ldots,r_k,x_o,s,\underline{u}) \qquad \text{(B.3)}$$

Consider the last term of Eq. (B.3). Using the identity in Eq. (A.1)

$$H(r_o,r_1,\ldots,r_k,x_o,s,\underline{u}) = H(r_o,r_1,\ldots,r_k) + H_{r_o,r_1,\ldots,r_k}(x_o)$$

$$+ H_{r_o,r_1,\ldots,r_k,x_o}(s) + H_{r_o,r_1,\ldots,r_k,x_o,s}(\underline{u})$$

(B.4)

Again due to the presence of appropriate conditioning, the third term of Eq. (B.4) is zero while the last term reduces to $H_s(\underline{u})$. The second term reduces to $H_{r_o}(x_o)$ since the variables $r_1,\ldots,r_k$ give no information about x_o that r_o does not give. Hence, Eq. (B.4) becomes

$$H(r_o,r_1,\ldots,r_k,x_o,s,\underline{u}) = H(r_o,r_1,\ldots,r_k) + H_{r_o}(x_o) + H_s(\underline{u}) \qquad \text{(B.5)}$$

Substituting Eq. (B.5) into Eq. (B.3) yields

$$G_c = \sum_{i=0}^{k} H(r_i) + H(x_o) + H(s) + H(\underline{u}) - H(r_o,r_1,\ldots,r_k) - H_{r_o}(x_o) - H_s(\underline{u})$$

$$= T(r_o:r_1:\ldots:r_k) + T(r_o:x_o) + H(s) + H(\underline{u}) - H_s(\underline{u}) \qquad \text{(B.6)}$$

Consider the term $T(r_o:x_o)$; by definition,

$$T(r_o:x_o) = H(r_o) - H_{x_o}(r_o) \qquad \text{(B.7)}$$

The variables r_o and x_o are defined in Eqs. (3.10) and (3.11) in such a way that r_o will assume a "true" value if x_o is inactive, otherwise r_o will assume a "false" value. Hence, the full knowledge of x_o leaves no uncertainty in determining the value of r_o. The second term of Eq. (B.7) is zero and Eq. (B.7) becomes

$$T(r_o:x_o) = H(r_o) \tag{B.8}$$

Substituting Eq. (B.8) into Eq. (B.6) yields

$$G_c = T(r_o:r_1:...:r_k) + H(r_o) + H(s) + H(\underline{u}) - H_s(\underline{u})$$

APPENDIX C

COMPUTATION OF TOTAL ACTIVITIES FOR NUMERICAL EXAMPLE

This appendix shows all the steps in computing the total activities of the preprocessor and the decisionmaker of the numerical example discussed in Section 3.4.

G^{PP}:

To evaluate G^{PP}, the numerical value for each of the terms in Eq. (3.24) is computed as follows:

Given $p(x_i) = 0.25$, $i=1,2,3,4.$

then $H(x) = 2$ (C.1)

According to the definition of s in Eq. (3.12), its distribution is

$$p(s) = \begin{cases} 0.5, & s = 0 \\ 0.25, & s = 1 \\ 0.25, & s = 2 \end{cases}$$

Hence, $H(s) = 1.5$ (C.2)

Similarly, according to Eq. (3.13), the distribution of variable u is

$$p(u) = \begin{cases} 0.5, & u = \square \\ 0.25, & u = 1 \\ 0.25, & u = 2 \end{cases}$$

So, $H(u) = 1.5$ (C.3)

According to Eq. (3.10), the distribution for variable r_o is

$$p(r_o) = \begin{cases} 0.5, & r_o = \text{true} \\ 0.5, & r_o = \text{false} \end{cases} \quad \text{(C.4)}$$

Thus, $H(r_o) = 1$ (C.5)

To compute $T(r_o:r_1:r_2)$, the distributions of r_o, r_1, r_2 as well as the joint distribution of the r_i's are needed.

The distribution $p(r_o)$ is given in Eq. (C.4). In order to compute $p(r_1)$, $p(r_1|r_o)$ is needed. According to the internal structure of the preprocessor in Fig. 3.4,

$$p(r_1|r_o = \text{true}) = \begin{cases} 1, & r_1 = \text{false} \\ 0, & r_1 = \text{true} \end{cases}$$

$$p(r_1|r_o = \text{false}) = \begin{cases} 0.5, & r_1 = \text{true} \\ 0.5, & r_1 = \text{false}. \end{cases} \quad \text{(C.6)}$$

Since $p(x) = \sum_y p(x|y)\, p(y)$ (C.7)

it follows that

$$p(r_1) = \begin{cases} 0.25, & r_1 = \text{true} \\ 0.75, & r_1 = \text{false}. \end{cases} \tag{C.8}$$

Then, $p(r_2|r_0,r_1)$ is needed to compute $p(r_2)$.

$$p(r_2|r_0 \text{ or } r_1 = \text{true}) = \begin{cases} 1, & r_2 = \text{false} \\ 0, & r_1 = \text{true} \end{cases}$$

$$p(r_2|r_0 \text{ and } r_1 = \text{false}) = \begin{cases} 1, & r_2 = \text{true} \\ 0, & r_2 = \text{false}. \end{cases}$$

Since $r_0 = \text{true}$ and $r_1 = \text{true}$ are mutually exclusive but not independent events, it follows from Eqs. (C.4), (C.6) and (C.8) that

$$p(r_0 \text{ or } r_1 = \text{true}) = p(r_0 = \text{true}) + p(r_1 = \text{true}) = 0.5 + 0.25 = 0.75$$

and

$$p(r_0 \text{ and } r_1 = \text{false}) = p(r_1 = \text{false}|r_0 = \text{false})\ p(r_0 = \text{false})$$

$$= 0.5 \times 0.5 = 0.25.$$

Hence, using Eq. (C.7),

$$p(r_2) = \begin{cases} 0.25, & r_2 = \text{true} \\ 0.75, & r_2 = \text{false} \end{cases}$$

By computing the various entropies from the above distributions and applying the identity of Eq. (A.1) on the term $H(r_0, r_1, r_2)$,

$$T(r_0:r_1:r_2) = H(r_0) + H(r_1) + H(r_2) - H(r_0, r_1, r_2)$$

$$= H(r_1) + H(r_2) - H_{r_0}(r_1) - H_{r_0,r_1}(r_2)$$

$$= 3.5 - 1.5 \log_2 3$$

Therefore, according to Eq. (3.24) and using Eqs. (C.1), (C.2), (C.3), (C.5) and (C.8), the total G^{PP} is obtained:

$$G^{PP} = 2 + 1.5 + 1.5 + 1 + 3.5 - 1.5 \log_2 3 = 7.12.$$

$\underline{G^{DM}}$:

To evaluate G^{DM}, the numerical value for each of the terms in Eq. (3.25) is computed as follows. In order to compute $H(x_0,u)$, it is necessary to know the distributions $p(x_0)$ and $p(u|x_0)$, since

$$p(x_0,u) = p(u|x_0)\ p(x_0). \tag{C.9}$$

According to the definition of x_0 in Eq. (3.11),

$$p(x_0) = \begin{cases} 0.5, & x_0 = \square \\ 0.25, & x_0 = x_3 \\ 0.25, & x_0 = x_4 \end{cases}$$

Since only deterministic decision strategies are employed in this example,

$$p(u|x_o) = \begin{cases} 1 , & u = \square , & x_o = \square \\ 1 , & u = 1 , & x_o = x_3 \\ 1 , & u = 2 , & x_o = x_4 \end{cases}$$

Hence, by Eq. (C.9)

$$p(x_o,u) = \begin{cases} 0.5 , & u = \square , & x_o = \square \\ 0.25 , & u = 1 , & x_o = x_3 \\ 0.25 , & u = 2 , & x_o = x_4 \end{cases}$$

and so

$$H(x_o,u) = 1.5 \tag{C.10}$$

Since u can be determined entirely by x,

$$H_x(u) = 0. \tag{C.11}$$

In the term

$$\sum_{i=1}^{2} p_i g_n^i(p(x|u=i))$$

where

$$p_i = p(u=i) = \begin{cases} 0.25 , & i = 1 \\ 0.25 , & i = 2 \\ 0.5 , & i = \square \end{cases} \tag{C.12}$$

g_n^i is the inherent uncertainty of the i^{th} algorithm given that the algorithm f_i is chosen to process the input. To evaluate g_n^i, the probability distributions of the internal variables within the ith algorithm are required. The stochastic algorithm with function $f_1(x)$ defined in Eq. (3.22) has several internal variables interconnected as shown in Fig. C.1.

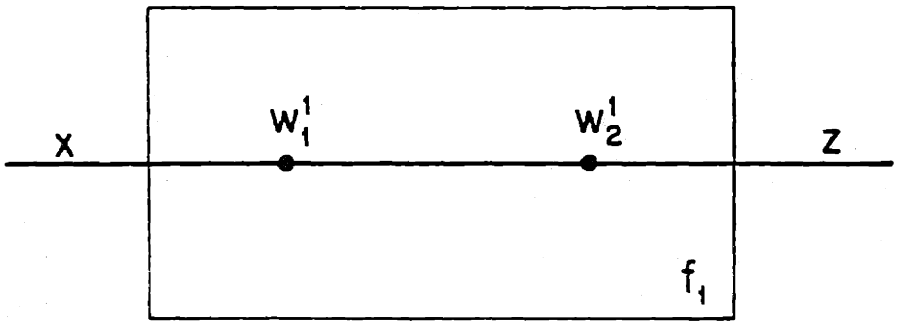

Figure C.1. The Stochastic Algorithm f_1

The probability distributions of the two internal variables in the algorithm f_1 are

$$p(w_1^1) = \begin{cases} 1 & , \quad w_1^1 = x \\ 0 & , \quad \text{otherwise} \end{cases} \tag{C.13}$$

$$p(w_2^1) = \begin{cases} 0.5 & , \quad w_2^1 = w_1^1 + 1 \\ 0.5 & , \quad w_2^1 = w_1^1 \end{cases} \tag{C.14}$$

According to Eq. (2.35), g_n of $f_1(\cdot)$ is

$$g_n = \sum_{w_\lambda \varepsilon \bar{D}} H(w_\lambda).$$

In this algorithm, $\bar{D} = \{w_2^1\}$, therefore

$$g_n^1 = H(w_2^1)$$

It follows from the distribution of w_2^1 defined in Eq. (C.14) that

$$g_n^1 = 1$$

Similarly,

$$g_n^2 = g_n^1 = 1 \tag{C.15}$$

Therefore, from Eqs. (C.12) and (C.15)

$$\sum_{i=1}^{2} p_i g_n^i(p(x|u=i)) = p_1 g_n^1 + p_2 g_n^2 = 0.25(1) + 0.25(1) = 0.5 \tag{C.16}$$

To compute the term

$$\sum_{i=1}^{2} p_i g_c^i(p(x|u=i))$$

it is necessary to compute g_c. According to Eq. (2.39), g_c of $f_1(\cdot)$ is

$$g_c^1 = \sum_{w_\lambda \varepsilon D-\{w_1\}} H(w_\lambda).$$

In this case, $D = \{w_1^1\}$ and $w_1 = w_1^1$, hence

$$D - \{w_1\} = \phi$$

and

$$g_c^1 = 0.$$

Similarly, $g_c^2 = g_c^1 = 0$. Therefore,

$$\sum_{i=1}^{2} p_i g_c^i(p(x|u=i)) = p_1 g_c^1 + p_2 g_c^2 = 0 \tag{C.17}$$

Since there are only two internal variables in each of the two stochastic algorithms, $\alpha_1 = \alpha_2 = 2$. With $H(p)$ defined in Eq. (2.22), and the values of p_1 and p_2 defined in Eq. (C.12), the term

$$\sum_{i=1}^{2} \alpha_i H(p_i) = 2\ [2(-0.25\ \log_2 0.25 - 0.75\ \log_2 0.75)] = 3.25 \tag{C.18}$$

Finally, by enumerating all the possible outputs of the two algorithms,

$$p(z) = \begin{cases} 0.5 & , z = \square \\ 0.25 & , z = x \\ 0.125 & , z = x + 1 \\ 0.125 & , z = x - 1 \end{cases}$$

and so

$$H(z) = 1.75 \tag{C.19}$$

Therefore, according to Eq. (3.25), and using Eqs. (C.10), (C.11), (C.16), (C.17), (C.18) and (C.19), the total G^{DM} is obtained:

$$G^{DM} = 1.5 + 0 + 0.5 + 0 + 3.25 + 1.75 = 7$$