INFORMATION SCIENCES **57–58**, 319–345 (1991)

Computer Scheduling Algorithms: Past, Present, and Future*

K. M. BAUMGARTNER

Digital Equipment Corp., Maynard, Massachusetts 01754-2571

and

B. W. WAH

Coordinated Science Laboratory, University of Illinois, Urbana, Illinois 61801-3082

ABSTRACT

Efficient scheduling techniques of computing resources are essential for achieving satisfactory performance for users as computer systems and their applications become more complex. In this paper, we survey research on scheduling algorithms, review previous classifications of scheduling problems, and present a broader classification scheme. Using a uniform terminology for scheduling strategies and the new classification scheme, previous work on scheduling strategies is reviewed and trends in scheduling research are identified. Finally, a methodology for developing scheduling strategies is presented.

1. INTRODUCTION

Early computer systems were centralized due to the cost of replicating hardware and additional staffing. As hardware costs dropped, it became possible for smaller organizations to own computer systems. Consequently, several computer installations could be present on a college or industrial campus, and local area networks (LANs) evolved to allow communication among the computer installations. The resulting collection of resources and the communications medium are distributed computer systems (DCS's). This trend is even more prevalent now as networks of personal computers and workstations such as Sun, Apollo, and MicroVAX are common in the work place. The difference between workstations and personal computers is the order of

* This research was supported by the National Aeronautics and Space Administration under grant NCC 2-481 and the National Science Foundation under grant MIP 88-10584.

development of communication facilities. Personal computers were designed as independent computers, and later networking capability was added (this is similar to communication among mainframe computers). The communication among workstations was developed simultaneously with the computing power, allowing communication services to be more easily integrated. Now computing power is literally distributed from desktop to desktop.

There are three distinguishing characteristics of local area networks: they are comprised of autonomous systems so that control is not limited to one location; there is a physical distribution of resources (typically on the order of one kilometer); and the speed of communications ranges from approximately one to 20 megabits per second. Enslow [18] more formally specifies five requirements for a DCS: a multiplicity of general purpose resources (physical and logical), a physical distribution of these resources, a high-level operating system to integrate control, system transparency so that services may be requested by name, and operation of the resources characterized by cooperative autonomy.

The networks which connect computers and workstations allow communication, but they also have capability to allow efficient sharing of resources. Since the demands for computing power are continually increasing, the network can be used for scheduling tasks during time when it is otherwise idle. DCS's can provide a cost-effective means to increase the computing power available to a single computer user if jobs can be scheduled to exploit potential parallelism. Livney and Melman [33] have shown that in a system of n independent processors modeled as M/M/1 systems [29], the condition where a job is waiting for service at one processor while another processor is idle occurs 70% of the time for traffic intensities (the ratio of the arrival rate to the service rate) ranging from 0.5 to 0.8. This idle-while-waiting condition indicates the possibility of reducing the average job delay. With a global scheduling strategy for a DCS, the occurrence of the idle-while-waiting condition can be reduced and consequently the overall performance of the DCS can be improved.

1.1. DEFINITION OF THE SCHEDULING PROBLEM

The scheduling problem is shown in Figure 1. It is a mapping of a set of jobs or tasks to a set of processors (or machines). The job characteristics (processing time and precedence constraints), machine environment (number of processors, interconnection, power of processors), and performance objectives are the input to the scheduler, and the mapping, or schedule, is the output. There are five components of the problem: the events, the environment, the requirements, the scheduler, and the schedule. The scheduler allocates resources to events.

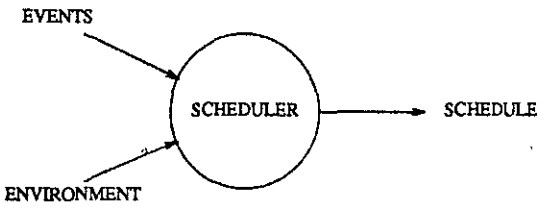

Fig. 1. The scheduling problem

Events are the smallest indivisible schedulable entity. The environment refers to all characteristics and capabilities of the surroundings that will impact the schedule such as physical capabilities of the processors and the communication mechanism. Requirements are also input to the scheduler and may range from a real time deadline to the requirement of determining if an improvement in performance is possible. The scheduler takes the events, environment, and requirements as input, and produces a schedule. In the precisely specified case, the schedule will be a set of ordered pairs of events and times. Frequently it is not possible to precisely state a schedule, in which case the schedule is specified by a set of rules. These rules will specify dynamic correcting actions when the distribution of events in the environment is such that the correcting action will result in an improvement in some system performance parameter.

In order to fully investigate the problem of scheduling, it is beneficial to consider a larger perspective as shown in Figure 2. The starting point is a *problem*. The problem can be *deterministic, non-deterministic,* or *undecidable*. A *deterministic* problem is clearly defined and tractable in terms of both time and space. Traditionally, deterministic scheduling problems refer to those scheduling problems that have all information required for generating a schedule specified *a priori*. This excludes any data dependent tasks (i.e., decision points). An example would be matrix multiplication. A *non-deterministic* problem may be clearly defined but too complex to find an exact solution with finite resources, or it may be clearly defined but require too much space to find an exact solution with finite resources, or it may require too much time and space. Alternatively, a non-deterministic problem has in-

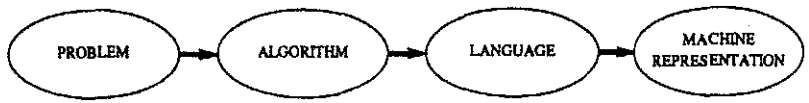

Fig. 2. Mapping from problem to machine representation

puts which are not exactly specified. For example, a scheduling problem with resource requirements specified as distributions is a deterministic scheduling problem. There is no algorithm to solve an *undecidable* problem even with infinite processing and storage resources. An undecidable problem cannot be solved directly, nor can it be represented directly as a deterministic problem. Only a deterministic problem can be solved, so the non-deterministic problem is solved using a deterministic algorithm that approximates using heuristics. The *algorithm* is the specified in a *language* which is in turn given a *machine representation*.

During the 1960s and early 1970s, a great deal of attention was focused on scheduling problems that emerged in a manufacturing environment [13]. These problems were predominantly deterministic as the arrival time of jobs requiring service and the duration of service were exactly known. Scheduling problems that emerge in computer systems are non-deterministic because exact information about resource requirements is rarely available. Deterministic and non-deterministic scheduling problems are discussed further in Section 2.

Scheduling resources on distributed systems has two aspects: intracomputer scheduling and intercomputer scheduling. Intracomputer scheduling concerns scheduling within a computer (local scheduling) while intercomputer concerns scheduling tasks among computers (global scheduling). Intracomputer scheduling occurs at many levels within the processor such as through the memory hierarchy, at the device and functional unit level, and also scheduling processes. When a computer is composed of multiple processors, scheduling among them is another level. Intercomputer scheduling is a level above processor scheduling, and involves communication among independent computers. The levels of scheduling are shown in Figure 3. At each level, scheduling is a mapping of events to the environment.

1.2. ORGANIZATION

This paper is organized as follows. In Section 2 previous classifications of scheduling problems are reviewed. These previous classifications are for deterministic scheduling problems, which are problems with all resource requirements specified exactly. A new classification of scheduling problems which incorporates both deterministic and stochastic scheduling problems is presented in Section 3. Using the new classification, trends in research on scheduling problems are investigated. Terminology for scheduling strategies is also introduced in that section. Section 4 reviews previous work on scheduling. A methodology for developing scheduling strategies is discussed in Section 5, and Section 6 contains concluding remarks.

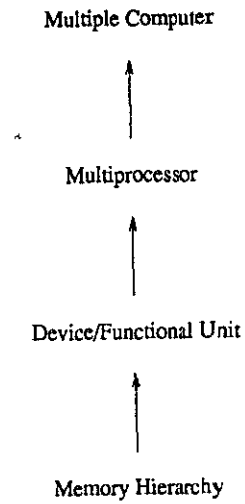

Fig. 3. The levels of scheduling

2. PREVIOUS CLASSIFICATIONS OF SCHEDULING PROBLEMS

Before discussing classifications of scheduling problems, it is useful to state the reasons why a classification is needed, and to identify the desirable characteristics of a classification. The goal of developing a classification is to increase and organize overall knowledge about a class of problems. This goal is realized through two steps: specification of a problem and showing relationship among problems. A classification implies developing a taxonomy, which addresses both the above steps, since a taxonomy implies a specification by categorization. Categorization in turn shows relationships since problems with attributes in the same category will be related. Thus the taxonomy helps organize knowledge about a class of problems.

There are at least four attributes of a classification that are desirable. First is to identify the significant characteristics of the problem since these will contribute to an efficient solution. Next is to clearly show the relationship among problems. This is beneficial because the solution to one problem may indicate the solution to a closely related problem. Conversely if a problem has no solution, that may indicate a related problem also has no solution. Expandability and contractability of a classification are useful because they allow important features of the problem to be focused on and unimportant details of the problem can be eliminated to reduce the complexity of the problem representation. Finally, it is desirable for the classification to separate the problem specification from the solution. The separation allows a clearer

comparison of scheduling strategies for a given problem, and avoids confusing a strategy and a problem.

Three classifications of scheduling problems are discussed in this section. First is Conway et al.'s classification [13], a four-parameter scheme where the categories are A: the job arrival process which for static arrivals indicates the number, and for dynamic arrivals indicates the probability distribution of the time between arrivals; B: the number of machines in the shop; C: the flow pattern in the shop; and D: objective function. This classification has two of the desirable characterics stated earlier: it identifies significant problem features and separates the problem from the solution, but lacks expandability and contractability. Much of the problem is unspecified, such as whether the machines are homogeneous or heterogeneous, if pre-emption is allowed, and details about the kinds of tasks to be scheduled. There is no capacity for specifying stochastic problems with this classification.

Graham et al [23] classified deterministic problems with requirements of optimal solutions. The classification uses three sets of attributes: job characteristics (α), machine environment (β), and objective function (γ). These sets of attributes are subdivided into numerous components. In effect, Graham and Lawler's classification combined Conway et al.'s [13] B and C categories into one field specifying the machine environment, and provided a more detailed specification for the job specification and machine environment. The $\alpha/\beta/\gamma$ classification allows precise specification of problems. In fact, by enumerating the possible values of the categories, a finite set of problems can be named. Identification of significant features, grouping sets of related characteristics and separating the problem from the solution are also advantages of this classification. This classification is especially good at showing the relationship among problems using reducibility. As an example, the reducibility for task structure is shown in the attribute task precedence. Scheduling tasks governed by general precedence constraints is more difficult than scheduling tasks governed by tree-like precedence which in turn is more difficult than scheduling independent tasks. Solution to a more difficult problem implies the solution of the easier problem. Unfortunately, this classification does not include representations for stochastic scheduling problems or representations for problems with non-optimal objective function requirements. Since each attribute has values that are explicitly specified, a uniform representation of stochastic problems using this representation would require a large expansion of the representation. The set of stochastic problems is extremely large, and an expanded version of the $\alpha/\beta/\gamma$ representation would be unwieldy.

Gonzalez [21] uses nine classification categores: the number of processors, task duration (either equal or unequal), precedence graph structure, task interruptibility, processor idleness (whether this is allowed or not), job periodicity, presence or absence of deadlines, resource limited schedules, and

homogeneous versus heterogeneous processors. While this classification identifies nine key features, there is no grouping of related characteristics, no relationship among problems shown, and the classification is not easily expandable. Also, this classification is not designed for problem specification.

3. A NEW CLASSIFICATION FOR SCHEDULING PROBLEMS

In this section, a new classification of scheduling problems is presented. Additionally, terminology for describing scheduling strategies is also described.

3.1. THE ESR CLASSIFICATION SCHEME

In the process of developing a new classification, it was beneficial to use some of the methods of the previous classifications, specifically the grouping techniques. Consequently, the classification uses three groups corresponding to the input components of the scheduling problem noted in Section 1. These three categories (the events, the environment (or surroundings), and the requirements) comprise the new ESR classification scheme. These correspond in part to the categories of the $\alpha/\beta/\gamma$ classification. There is some rearrangement, and this classification is not as explicit. The categories and the attributes of the ESR classification are summarized in Table 1.

The first category, E, is the event. The three attributes of events considered are their relationship with other events, arrival patterns, and the available information about the resource requirements of each event. Events may be independent, in which case the execution time is not constrained by other events. Conversely, events may have precedence constraints, where an event must be executed before its successors. Another relationship among events is communication, where two events exchange information during or between their execution. This relationship is especially important when communication costs are considered.

Availability indicates whether events, or jobs, all arrive simultaneously prior to execution, whether they arrive periodically, or whether their arrival is governed by a stochastic distribution. The first case is of simultaneous arrival is also called static arrivals. Periodic arrivals are similar to static arrivals in the sense that they can be considered to be a repetition of static arrivals if arrivals are sufficiently far apart to allow execution of the previous set of jobs to complete before the next set arrives. Stochastic arrivals have interarrival times that are governed by a probability distribution.

TABLE I
The ESR Classification

Category	Attribute	Values
Event	Relation to other events	independent
		precedence
		communication
	Availability (arrivals)	static
		periodic
		stochastic
	Resource Requirements	deterministic resource requirements
		stochastic resource requirements
Environment	Number	1, k, n
	Classes of Resources	homogenous resources (1 class)
		heterogeneous resources (>1 class)
	Physical Characteristics	speed
		size of main memory
		special capabilities, etc.
	Paths Connecting Resources	*toplogy*
	Communication Overhead	none
		deterministic
		stochastic
	Communication Mechanism	flow
		broadcast
		message passing
		other
Requirements	Performance	any solution
		deadlines (real time)
		good (improved) solution
		optimal solution

The final attribute of events is the resource requirements, which may be completely specified as constants and so be deterministic, or may be governed by a probability distribution. Clearly these three attributes are important in the scheduling process. The available information about resource requirements is especially important, however frequently in multiprocessor systems there is limited information available.

The second category, S, is the environment or surroundings. Any environmental characteristics that will impact a schedule, such as the number of resources, the number of classes of resources, the physical characteristics of each class, the topology of the communications medium, the communication delay, and the communication mechanism, are included in this category. The number of resources can be one, implying scheduling on one machine, a specific number such as two or three (specified as k), or an arbitrary number, n. The resources may be homogeneous (all are identical) or homogeneous (all are unique or multiple classes of processors). The physical characteristics of these classes can also be indicated if they are significant. Interconnection among resources is called the path among them. There are infinite possibilities for this category including bus, ring, multistage networks, and arbitrary point-to-point connections. Communication overhead may be zero, in which case events flow among resources at no cost, or may have a constant (deterministic) or varying (stochastic) cost. Last, the communication mechanism may be a flow of events, broadcast of information, message passing, or take some other form. The scheduler uses information about the surroundings to make predictions about the relative costs of different schedules.

The final category is the requirements, R, of the schedule. In the simplest case, the requirements are for any solution or schedule. Other cases include decreasing response time (or another performance metric) resulting from a previous schedule, meeting a fixed deadline, finding the schedule that uses the minimum number of processors, and so on.

A key point about this classification is that it is high level, meaning that attributes and their values can be specified in general terms or in more detail. When a scheduling problem is specified using this classification, the values of the attributes are expanded if they reflect a characteristic that will be used in creating the schedule. Unlike the deterministic scheduling problems considered with the $\alpha/\beta/\gamma$ classification, stochastic scheduling problems are not enumerable; it is impossible to list every problem. With this classification, it is possible to specifically note those values the scheduler uses and leave less important parameters expressed in more general terms. Problem specification is indicated by the three categories using the following notation:

$$\text{E:}\{\bullet\} - \text{S:}\{\bullet\} - \text{R:}\{\bullet\}$$

The sets correspond with each category, and include as much detail as is desired. If the sets are empty, it is presumed that there is no useful information available about that category. For example, deterministic problems can be specified as

$$\text{E:}\left\{\begin{array}{c}\text{arbitrary acyclic}\\ \text{event precedence}\end{array}\right\}\left\{\begin{array}{c}\text{static or periodic}\\ \text{arrivals}\end{array}\right\}\left\{\begin{array}{c}\text{deterministic}\\ \text{resource}\\ \text{requirements}\end{array}\right\} - \text{S:}\left\{\begin{array}{c}\text{n processors}\end{array}\right\}\left\{\begin{array}{c}\text{heterogeneous}\\ \text{resources}\end{array}\right\}\left\{\begin{array}{c}\text{deterministic}\\ \text{communication}\\ \text{overhead}\end{array}\right\} - \text{R:}\left\{\begin{array}{c}\text{optimize some}\\ \text{performance}\\ \text{parameter}\end{array}\right\}.$$

Stochastic scheduling problems, in contrast, can be specified as

$$\text{E: } \left\{ \begin{array}{c} \text{arbitrary acyclic} \\ \text{event precedence} \\ \text{stochastic arrivals} \\ \text{stochastic resource} \\ \text{requirements} \end{array} \right\} - \text{S: } \left\{ \begin{array}{c} \text{n processors} \\ \text{heterogeneous} \\ \text{resources} \\ \text{stochastic} \\ \text{communication} \\ \text{overhead} \end{array} \right\} - \text{R: } \left\{ \begin{array}{c} \text{reduce} \\ \text{response time} \end{array} \right\}.$$

3.2. *SCHEDULING TERMINOLOGY*

There is a considerable conflict in the literature regarding terminology used to describe attributes of scheduling strategies. This section discusses terminology for classifying scheduling strategies. First, some previous classifications of scheduling strategies are reviewed. Using the results of these studies, terminology used in this paper is described.

Wang and Morris developed a classification of scheduling algorithms [54]. The criteria for classification is whether the strategy is source-initiated or sink-initiated meaning whether overloaded resources look to alleviate their load, or lightly loaded resources actively pursue more work. Additionally, the level of information dependency is a factor. Information dependency refers to the level at which a resource has information about the current state (workload) of other resources.

The terminology used in global scheduling (or load balancing) [5, 12, 15, 19, 38, 52] is varied and conflicting. Some features commonly discussed (using different names) are whether the scheduling intelligence is centralized or distributed, whether the rule basis is static, meaning independent of the current state of the system, or dynamic, meaning decisions are state-dependent. This characteristic is also referred to as being deterministic versus probabilistic, or adaptive versus non-adaptive (adaptability also refers to a different attribute discussed below). These features are useful for comparing scheduling strategies. The comparison has indicated that these characteristics are related to the potential of an algorithm, as will be discussed in Section 4.

The informal classification used for load balancing problems above was formalized by Casevant and Kuhl in an attempt to unify the diverse notation used [9]. Their classification was designed for distributed computing systems and consists of two parts: a hierarchical classification and a flat classification. The hierarchical classification is used to show where some characteristics are exclusive. The flat classification gives definitions of attributes that are not exclusive. Several observations about this taxonomy are:

(a) Adaptability refers to long-term algorithm state-dependency rather than short-term state dependency for rule basis. Adaptability is available with both a static and dynamic rule basis since the algorithm can be static (or dynamic) for one time interval, and then change for the next time interval.

(b) Load balancing and optimality are considered strategy characteristics rather than problem requirements (elements of the set R).

(c) One time reassignment and dynamic reassignment (which correspond to pre-emption in the one processor case) are considered strategy characteristics rather than environment capabilities (elements of the set S).

(d) Bidding in the flat portion of the classification and cooperation in the hierarchical portion of the classification are not distinct.

The terminology summarized in Table 2 is not exclusively adopted from any of these sources for the following reasons. Wang and Morris' classification [54] focuses on only two aspects of the strategy (initiation location and information dependency), so it not extensive enough. Casevant and Kuhl's taxonomy is not used exclusively because there is overlap between their classification of solutions and the classification of problems described in Section 2.3 [9].

The first characteristic specified is the level of scheduling as described in Section 1. The rule basis is included next and may be static or dynamic. Note this is different than the static versus dynamic sets used to describe event arrivals. Next is the location of control. If control has a degree of distribution, the controlling processors can negotiate to make scheduling decisions, or function independently. Initiation and adaptability are the two final characteristics. The tradeoffs of these different characteristics are discussed in the next section.

The relationship among the six characteristics is shown in Figure 4. Each path through the relationship graph indicates a set of attributes a schedule may have. Some paths and the corresponding combination of sets of attributes are not possible. For example it is not possible to have a centralized strategy with cooperation. Centralized implies one decision point. Static strategies are considered centralized since although each participating strategy may make different decisions, the choice was determined *a priori* by one scheduling intelligence. In Section 4 this terminology is used to discuss the results obtained in previous work on scheduling problems.

Previous classifications of scheduling problems are limited to deterministic scheduling problems. The ESR classification includes both deterministic and stochastic problems, and specifies problems using three sets of characteristics, events, environment (or surroundings), and requirements. The ESR scheme specifies problems in a flexible manner, so that attributes of the three sets can be indicated with varying degrees of completeness allowing attention

TABLE 2
Revised Classification of Scheduling Strategies

Terminology		
Characteristic	Values	Explanation
Level of Scheduling	Intra-resource Inter-resource	Refers to scheduling within a node or resource as opposed to among nodes or resources. There may be multiple sub-levels of intra-resource scheduling. Also called global versus local.
Rule Basis	Static Dynamic	Refers to the flexibility of schedule rules to react to the current state of the system. A static schedule bases rules on unchanging system characteristics. A dynamic schedule bases rules on the current state of the system. Also called state-dependency.
Location of Control	Distributed Hierarchical Centralized	Describes where the responsibility for scheduling decisions lies. This applies primarily to dynamic timing since static timing implies a centralized decision. Hybrids are also possible.
Cooperation	Negotiated Independent	Describes the interaction among locations of control (distributed or hierarchical only). This applies to distributed or hierarchical control since centralized does not have separate modules to cooperate.
Initiation	Source Sink Both	Which processor initiates job movement, the overloaded processor (source initiated) or the underloaded processor (sink initiated).
Adaptability	Adaptable Non-adaptable	Refers to flexibility of the algorithm, and whether the algorithm changes execution based on the arrivals to the system.

to be focused on important features of the problem, and less important details to be left in general terms.

4. STATE OF THE ART IN SCHEDULING RESEARCH

In this section, trends in scheduling problems that research is focused upon are identified. This information indicates the current importance of scheduling problems on DCS's. The trends also show the progression of understanding of scheduling problems. Table 3 chronologically lists studies on scheduling

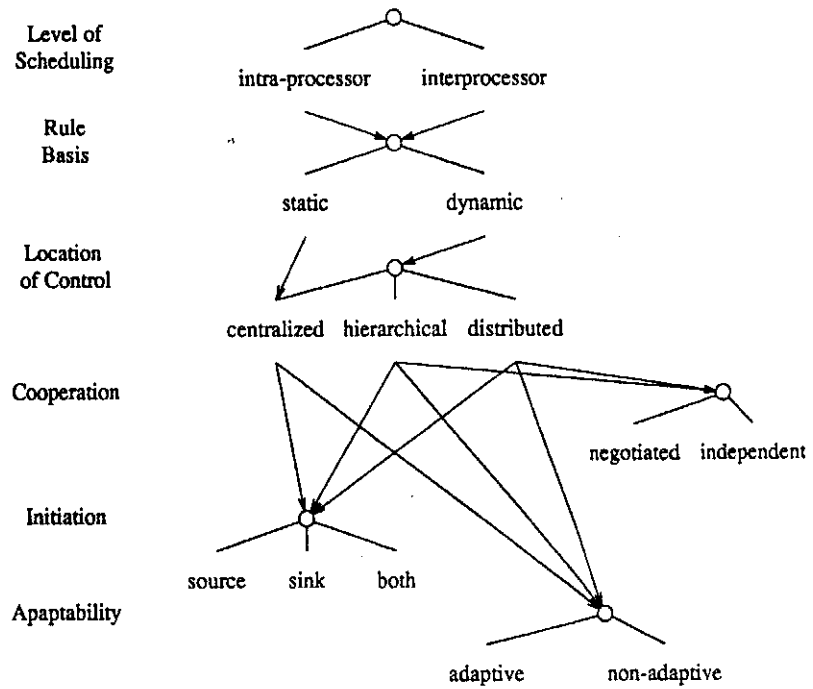

Fig. 4. Relationship among strategy characteristics

problems. Problems are uniformly specified in this table. Three categories indicating trends are communication overhead, resource requirements, arrival characteristics. Figure 5 shows a graphical representation of the trends. The progression is toward the general problem of scheduling tasks on distributed computing systems. This is not unexpected; the earlier work is on the more specific problems associated with deterministic scheduling, and later work focuses on the more general problems associated with stochastic scheduling problems.

The ESR classification scheme can be used to specify a problem, and to make comparisons among similar problems. It becomes difficult to show relationships among problems using all the attributes for a wide range of problems because it is possible for a given problem to be less general than another problem for one attribute, while being more general for another attribute. Because of this, the results of previous studies are shown by grouping problems by the characteristic that dominated the trends shown in the previous section (i.e., whether processing requirements are specified stochastically or deterministically).

TABLE 3
Summary of Scheduling Problems Studied

References	Event								Environment						
	Relationship			Availability			Resoure Requirements		Number			Resources		Communication Overhead	
	I	P	C	Static	Periodic	Stoch	Determ	Stoch	1	k	n	Homog	Heter	None	Some
[CoM67]	x			x	x	x	x		x	x	x	x		x	
[GoR72]	x	x		x			x		x	x	x	x		x	
[RaC72]	x	x		x	x		x	x	x	x	x	x		x	
[Bak74]	x			x	x		x		x	x	x	x	x	x	
[Sto77]	x	x		x	x		x		x	x	x	x	x	x	x
[LiH77]	x	x		x	x		x		x	x		x	x	x	
[GrL77]	x	x		x	x		x		x	x	x	x	x	x	
[Gon77]	x	x		x			x		x	x	x	x	x	x	
[StB78]	x	x		x	x		x		x	x		x	x	x	x
[Sto78]	x	x		x	x		x		x	x		x	x	x	x
[Apl78]	x	x		x				x	x			x		x	
[Gon79]		x		x			x		x	x	x		x		x

continued

TABLE 3
Summary of Scheduling Problems Studied (*continued*)

References	Event								Environment						
	Relationship			Availability			Resoure Requirements		Number			Resources		Communication Overhead	
	I	P	C	Static	Periodic	Stoch	Determ	Stoch	1	k	n	Homog	Heter	None	Some
[ChK79]	x					x		x	x	x	x	x	x	x	
[RaS79]	x	x		x	x		x		x	x		x	x	x	x
[OuS80]	x	x				x		x	x	x	x	x		x	x
[Weg80]	x			x			x		x	x	x	x		x	x
[EnH80]	x			x	x		x		x			x		x	
[ChH80]	x	x		x	x		x		x	x		x	x	x	x
[KrH80]	x			x			x		x	x	x	x	x	x	
[JaK80]		x		x			x		x	x	x		x	x	
[NiH81]	x					x		x	x	x	x	x	x	x	x
[LoL81]	x	x		x			x		x	x	x	x	x		x
[Efe82]	x	x		x			x		x	x		x	x	x	
[WaH82]	x					x		x	x	x		x	x	x	x
[LiM82]	x					x		x	x	x	x	x		x	x

continued

TABLE 3
Summary of Scheduling Problems Studied (*continued*)

References	Event								Environment						
	Relationship			Availability			Resoure Requirements		Number			Resources		Communication Overhead	
	I	P	C	Static	Periodic	Stoch	Determ	Stoch	1	k	n	Homog	Heter	None	Some
[HwC82]	x					x		x	x	x	x	x	x		x
[ChA82]	x	x		x				x			x	x	x		x
[Mon82]		x		x			x		x	x	x	x		x	
[Leu82]		x		x			x		x	x		x		x	
[Mal82]		x		x			x			x	x		x		x
[Bal83]		x		x			x		x			x		x	
[Wil83]		x		x			x		x	x	x		x		x
[Pin83]	x			x				x	x	x	x	x	x	x	
[WaP83]	x					x		x	x	x	x	x			x
[WaJ83]	x					x		x	x	x	x	x			x
[MeT84]	x	x			x	x		x	x	x	x	x	x	x	x
[KeL84]	x					x		x	x	x	x	x			x
[RaS84]	x			x	x	x	x		x	x	x	x	x		x

continued

TABLE 3
Summary of Scheduling Problems Studied (*continued*)

References	Event								Environment						
	Relationship			Availability			Resoure Requirements		Number			Resources		Communication Overhead	
	I	P	C	Static	Periodic	Stoch	Determ	Stoch	1	k	n	Homog	Heter	None	Some
[BaW85]	x					x		x	x	x	x	x			x
[BaH85]			x	x			x		x	x		x			x
[JeB85]			x	x			x		x	x		x			x
[TaT85]	x					x		x	x	x	x	x	x		x
[WaM85]	x					x		x	x	x	x	x		x	x
[WaJ85]	x					x		x	x	x	x	x			x
[NiH85]	x	x		x			x			x			x		x
NiX85]	x					x		x	x	x	x	x			x
[Sta85]	x					x		x	x	x	x	x			x
[JuW86]	x					x		x	x	x	x	x			x
[Ezz86]	x					x		x	x	x	x	x			x
[EaL86]	x					x		x	x	x	x	x			x
[Alo86]	x					x		x	x	x	x	x			x
[BaW87]	x					x		x	x	x	x	x			x

4.1. *PROBLEMS WITH DETERMINISTIC RESOURCE REQUIREMENTS*

A great deal of research was done for scheduling tasks with exact *a priori* knowledge of execution requirements during the 1960s and 1970s. This work has been described in several books [7, 13, 14] and survey papers [21, 23]. The problems are represented as follows.

$$\text{E: } \left\{\begin{array}{c}\text{arbitrary acyclic}\\ \text{precedence}\\ \text{static arrival}\\ \text{deterministic}\\ \text{resource}\\ \text{requirements}\end{array}\right\} - \text{S: } \left\{\begin{array}{c}\text{n processors}\\ \text{heterogeneous}\\ \text{resources}\\ \text{open shop,}\\ \text{flow shop paths}\\ \text{communication}\\ \text{by flow}\end{array}\right\} - \text{R: } \left\{\begin{array}{c}\text{optimal}\\ \text{near-optimal}\\ \text{approximate}\end{array}\right\}$$

Note that this specification includes less general problems (i.e., events that are independent tasks). When specifying a set of problems with the ESR classification, a more general problem encompasses less general problems. This is a class of problems that frequently occurs in a manufacturing environment and is scheduling at a inter-resource level. Scheduling strategies for this type of problem have been discussed extensively [7, 23]. Solutions are optimal, near optimal, or approximate. Schedules may be explicit time, event pairs or static rules such as schedule the shortest job first. Scheduling intelligence in this case is centralized.

As DCS's evolved, scheduling problems related to computing environment received more attention. The ESR specification of these problems is

$$\text{E: } \left\{\begin{array}{c}\text{arbitrary acyclic}\\ \text{precedence}\\ \text{static arrivals}\\ \text{deterministic}\\ \text{resource}\\ \text{requirements}\end{array}\right\} - \text{S: } \left\{\begin{array}{c}\text{n processors}\\ \text{heterogeneous}\\ \text{resources}\\ \text{deterministic}\\ \text{communication}\\ \text{overhead}\end{array}\right\} - \text{R: } \left\{\begin{array}{c}\text{optimal}\\ \text{near-optimal}\\ \text{approximate}\end{array}\right\}$$

Some early scheduling problems for the multiprocessor environment considered scheduling tasks with acyclic precedence requirements with no communication among tasks [20, 43]. These studies assumed that a reasonable estimate of resource requirements could be obtained with a preprocessing phase of a program. Results of these studies included a method for determining the minimum computation time, and a method to determine the minimum number of processors to achieve the minimum computation time [43]. Also, a comparison of a centralized versus a decentralized algorithm indicated that the decentralized algorithm performed better.

Scheduling tasks with intertask communication is a more difficult and a more realistic problem. Improving performance requires limiting excessive communication and evenly distributing the work load among processors. If all the tasks are scheduled on one processor, there is no communication cost, but there is no benefit from a multiplicity of processors. If the tasks are distributed such that processor utilization is completely uniform, full advantage of the multiplicity of processors is realized; however, communications costs may be so large that the benefits of concurrent execution are eliminated causing processing to be more expensive than with one processor. Since the goals of limiting communication and balancing load are directly conflicting, obtaining maximum performance is a tradeoff.

Several studies were performed by Stone et al. [44, 46, 48, 49] in which jobs were represented by graphs. The nodes represent tasks with execution requirements, the edges are labeled with communication costs. The representation is actually serial since the program will be executing on only one node at a time. The communication is information that one module sends to its successor on completion. An assignment of tasks is specified by a cut that divideds the graph into as many sections as there are processors. The cost of the assignment is equal to the sum of the execution costs plus the sum of the communication costs between tasks that are not assigned to the same processor (i.e., the sum of the weights of the edges on the cut). Consequently, the minimum cost assignment corresponds to the minimum cutset. The time requirements of this method may allow it to be used dynamically for two or three processors, but not efficiently for larger numbers of processors. A similar mapping problem for larger systems was explored by Bokhari [8] in which communicating modules are placed as much as possible on adjacent processors. Other approaches to this problem include graph matching [11, 44], mathematical programming [11, 56], a branch and bound algorithm [35], and heuristics [2, 11, 16, 30, 34]. An optimal solution for the n processor case was found by Chou and Abraham [10].

The solutions to these problems are for static sets of jobs, and the scheduling strategies are static as well. Their execution time is too long for them to be effective for dynamically arriving tasks. Their utility is then for the planning phase of a system (prediction of minimum or maximum execution time), or for real-time systems where timing is critical.

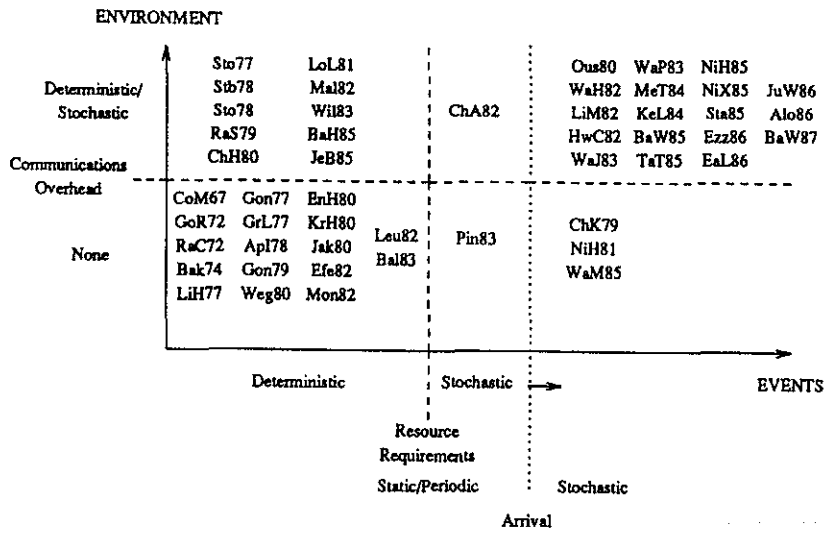

Fig. 5. Trends in scheduling research

4.2. *PROBLEMS WITH STOCHASTIC RESOURCE REQUIREMENTS*

The studies reviewed in the previous subsection used exact information about the execution and communication requirements of the jobs being scheduled. Frequently in scheduling problems on computer systems, this information is not known and only probabilistic information may be available. Further, dynamically arriving sets of jobs are more common in computer systems than static sets of jobs. This change in available information changes the methods and expectations of the scheduler. The problem is easier in some cases because with exact *a priori* information, there may be much information that it cannot be used efficiently [42]. The class of problems discussed in this section have an ESR representation as follows:

$$\text{E: } \left\{ \begin{array}{c} \text{independent tasks} \\ \text{dynamic arrivals} \\ \text{stochastic resource requirements} \end{array} \right\} - \text{S: } \left\{ \begin{array}{c} \text{n processors} \\ \text{homogeneous resources} \\ \text{bus or point-to-point interconnection} \end{array} \right\} - \text{R: } \left\{ \begin{array}{c} \text{evenly distribute load} \\ \text{minimize communication} \end{array} \right\}$$

The following issues are important when developing strategies for dynamic

scheduling problems with stochastic resource requirements: balanced load versus minimized communication tradeoff, location of control, the status information used for the scheduling decisions, and the initiation point. The balanced load versus communication tradeoff was discussed above.

Location of control is also a tradeoff. A centralized location of control may allow the scheduling strategy to be simpler. However, the decision point has the potential of becoming a bottleneck and a critical failure point. If a distributed decision is made, the overhead of distributing status information can be so high as to eliminate the benefits of scheduling. A comparison of centralized versus distributed strategies using a trace driven simulation was performed by Zhou [58]. The results indicate that neither strategy is always superior, and that the overhead for communicating information is important for both. The third consideration is what status information to use for scheduling decisions. Several studies on this topic have indicated that excessive status information is not only unnecessary, but can be detrimental [33, 50, 54].

Several static scheduling strategies have been proposed. Proportional branching is a static, sink-initiated strategy where jobs are routed to processors with a probability determined by the relative power of the processors [12]. Ni and Hwang found optimal static sink initiated strategies for single and multiple job classes again with centralized control [38]. An optimal static source-initiated strategy was found by Tantawi and Towsley for scheduling jobs that are modeled as independent tasks [50].

Dynamic strategies have more potential than static strategies because they can react to changes in the system state. Chow and Kohler [12] proposed three dynamic centralized strategies with a job dispatcher (sink-initiated) and found the one that maximizes throughput gives the best performance.

Distributed dynamic strategies are more complex than scheduling strategies using centralized control because of the added task of coordinating independent actions. Typically, distributed and adaptive strategies involve negotiation among participating processors. This negotiation involves communication of status information, and the selection of processors involved in the scheduling decision. The simplest method is to maintain a centralized table with load information, and processors can consult the table before sending jobs for remote execution [24]. This method has the similar problems as the centralized dispatcher in the sense of reliability and bottleneck. Another technique is to have each processor broadcast its load and keep track of the loads of the other processors, in effect duplicating the table at each processor in the system [33, 58]. Alternatively, only significant load changes can be broadcast resulting in a decrease in network overhead [33, 58]. Other methods of exchange involve nearest neighbors [28, 57]. Stankovic and Ramamritham have proposed a strategy that includes a bidding phase where negotiation takes place [45, 47]. Some significant results of these previous studies are as follows.

(a) Load balancing is beneficial since load imbalance occurs frequently in a system with 10 or more processors.

(b) Excessive state information to make a load-balancing decision is not necessary and may be detrimental.

(c) Sink-initiated strategies have the potential for improved performance over source-initiated.

(d) Dynamic strategies have greater potential than non-adaptive strategies.

(e) Centralized strategies may create reliability and bottleneck problems.

(f) Scheduling communication should not interfere with regular message transfer.

(g) There is a mismatch between the capabilities of the network and the communications required for scheduling operations.

5. A METHODOLOGY FOR DEVELOPING SCHEDULING STRATEGIES

There are three components of a scheduling strategy where improvement is possible: representation, procedure, and evaluation. Representation involves the identification of features significant to the problem. The procedure is the steps performed in the strategy. Finally, the evaluation is the measurement of parameters associated with performance. Evaluation may then involve measurement of system parameters for use in procedure decisions, and measurement of system parameters for performance evaluation.

The methodology for developing a scheduling strategy uses the following three assumptions: the system in question is a distributed computer system as discussed in Section 1, the arrival rate is stochastic, and the jobs resource requirements are expressed stochastically.

The overall flow of the methodology is shown in Figure 6. There are eight steps and each is made considering the key results noted in Section 4.3. The first step is to develop a problem representation for the three components: environment, events and requirements. In this step, environmental attributes of the system which may aid in the scheduling process are identified. Underlying communication network capabilities, such as broadcast or multicast capability and bandwidth, are important at this point as they may be helpful during later steps.

The procedure specification is next and consists of identifying system states that will allow improvement. Such states are those in which the idle-while-waiting condition arises, or those states which make the idle-while-waiting condition more likely. The next step is to determine feasible rearrangements of events or Δs that redistributed the jobs to correct the condition. Clearly the redistribution must use the environment capabilities indicated when the

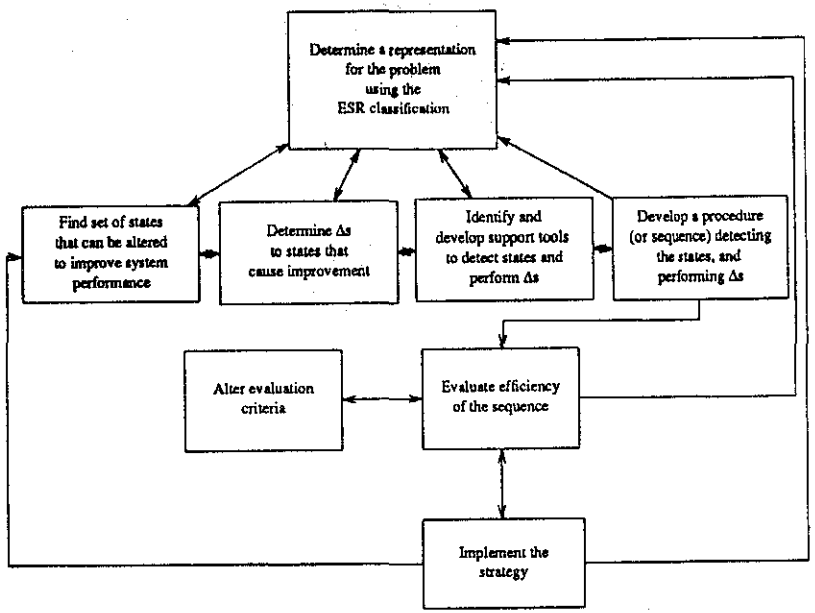

Fig. 6. The methodology for developing scheduling strategies

problem representation was specified. Consequently, it may be necessary at this point to iterate to the first step and redefine the representation to include more features to aid in identifying the states where improvement is possible, and the redistribution to effect the improvement.

The next step in procedure development is to determine the support tools for state detection and implementing the Δs. It is important to assure that these operations can be performed efficiently. If this is not the case, iteration may again be necessary. Candidates for change will be both the representation, where it may be possible to identify additional capabilities that can be utilized, and determining new states and Δs. It may be necessary to adjust expectations because network limitations preclude complex state detection and rearrangement. Conversely, care should be taken to assure that network capabilities are fully exploited.

With tools developed to identify states and Δs, the final step in the procedure development is to determine the overall sequence of operations. During the formulation of this sequence, other system responsibilities need to be considered. The sequence development may require adjustment to each of the four steps mentioned previously.

After the sequence has been generated, performance evaluation is necessary. This consists of two steps: determining the evaluation criteria (i.e.,

delay or throughput), and measuring the performance. Obviously, these steps must consider the set R. Also important is assuring the overhead of the scheduling strategy does not impede other system missions.

At the conclusion of the preliminary study, the strategy can be implemented. This may require iteration through all of the five steps of the methodology. Again, performance measurement will be necessary.

6. CONCLUDING REMARKS

Previous classifications of scheduling problems are limited to deterministic scheduling problems. In this paper, we have presented the ESR classification which includes both deterministic and stochastic problems. The ESR scheme specifies problems in a flexible manner, so that attributes of the three sets can be indicated with varying degrees of completeness allowing attention to be focused on important features of the problem, and less important details to be left in general terms. Using the attributes of the ESR classification, previous research on scheduling problems has been discussed and trends in scheduling problems that have been studied have been identified. Additionally, terminology for discussing scheduling strategies has been defined. Finally, a methodology for developing scheduling strategies has been described. It uses the ESR representation of the problem to focus key aspects that contribute to an efficient solution.

REFERENCES

1. R. Alonso, The Design of Load Balancing Strategies for Distributed Systems, *Future Directions in Computer Architecture and Software Workshop*, May, 1986.
2. W. F. Appelbe and M. R. Ito, Scheduling Heuristics in a Multiprogramming Environment, *IEEE Transactions on Computers*, July, 1978, pp. 628–637.
3. J. Barhen and E. C. Halbert, ROSES: An Efficient Scheduler for Precedence-Constrained Tasks of Concurrent Multiprocessors, *Proceedings of the SIAM First Conference on Hypercube Computers*, August, 1985, pp. 123–147.
4. K. R. Baker, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, Preemptive Scheduling of a Single Machine to Minimize Maximum Cost Subject to Release Dates and Precedence Constraints, *Operations Research* 31:381–386 (1983).
5. K. M. Baumgartner and B. W. Wah, The Effects of Load Balancing on Response Time for Local Computer Systems with a Multiaccess Network, *Proceedings of the International Conference on Communications*, June, 1985, pp. 10.1.1–10.1.5.
6. K. M. Baumgartner and B. W. Wah, Window Protocols for Load Balancing on a System with a Local Multiaccess Network, *Proceedings of the International Conference on Parallel Processing*, August, 1987, pp. 851–858.
7. K. A. Baker, *Introduction of Sequencing and Scheduling*, John Wiley and Sons, New York, 1974.

8. S. H. Bokhari, On the Mapping Problem, *IEEE Transactions on Computers*, March, 1981, pp. 207–214.
9. T. Casevant and J. Kuhl, A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Transactions of Software Engineering* 14:141–154 (1988).
10. T. C. K. Chou and J. A. Abraham, Load Balancing in Distributed Systems, *IEEE Transactions on Software Engineering*, Vol. SE-8:401–412 (1982).
11. W. W. Chu, L. J. Holloway, M. T. Lan, K. Efe, Task Allocation in Distributed Data Processing, *IEEE Computer*, November, 1980, pp. 57–68.
12. Y. C. Chow and W. Kohler, Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System, *IEEE Transactions on Computers*, C-28:354–361 (1979).
13. R. W. Conway, W. L. Maxwell, L. W. Miller, *Theory of Scheduling*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1967.
14. E. G. Coffman, Jr., *Computer and Job-Shop Scheduling Theory*, John Wiley and Sons, New York, 1976.
15. D. L. Eager, E. D. Lazowska, J. Zahorjan, Adaptive Load Sharing in Homogeneous Distributed Systems, *IEEE Transactions on Software Engineering* SE-12:662–675 (1986).
16. K. Efe, Heuristic Models of Task Assignment Scheduling in Distributed Systems, *IEEE Computer*, June, 1982, pp. 50–56.
17. G. P. Engleburg, J. A. Howard, D. A. Mellichamp, Job Scheduling in a Single-Node Hierarchical Network for Process Control, *IEEE Transactions on Computers*, August, 1980, pp. 710–719.
18. P. H. Enslow, Jr., What is a "Distributed" Data Processing System? *IEEE Computer*, January, 1978, pp. 13–21.
19. A. K. Ezzat, Load Balancing in Nest: A Network of Workstations, *Distributed Computing Conference*, 1986, pp. 1138–1149.
20. M. J. Gonzalez, Jr. and C. V. Ramamoorthy, Parallel Task Execution in a Decentralized System, *IEEE Transactions on Computers*, c-21:1310–1322 (1972).
21. M. J. Gonzalez, Jr., Deterministic Processor Scheduling, *Computing Surveys*, 9:173–204 (1977).
22. T. Gonzalez, A Note on Open Shop Preemptive Schedules, *IEEE Transactions on Computers*, October, 1979, pp. 782–786.
23. R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling, *Proceedings of Discrete Optimization*, August, 1977, pp. 47, 98.
24. K. Hwang, W. Croft, B. W. Wah, F. A. Briggs, W. R. Simmons, C. L. Coates, A Unix-Based Local Computer Network With Load Balancing, *IEEE Computer*, 15:55–66 (1982).
25. B. Jayaraman and R. M. Keller, Resource Control in a Demand-Driven Data-Flow Model, *Proceedings of the International Conference on Parallel Processing*, 1980, pp. 118–127.
26. D. Jefferson and Beckman, B, Virtual Time and Time Warp on the JPL Hypercube, *Proceedings of the SIAM First Conference on Hypercube Computers*, August, 1985, pp. 111–122.
27. J. Y. Juang and B. W. Wah, Channel Allocation in Multiple Contention Bus Networks, *INFOCOM*, July, 1986, pp. 34–42.
28. R. M. Keller and F. C. H. Lin, Simulated Performance of Reduction-Based Multiprocessors, *IEEE Computer*, July, 1984. pp. 70–82.
29. L. Kleinrock, *Queueing Systems Volum I: Theory*, John Wiley and Sons, New York, 1975.

30. A. Kratzer and D. Hammerstrom, A Study of Load Leveling, *Proceedings of the IEEE COMPCON*, Fall, 1980. pp. 647–654.
31. J. Y. T. Leung, On Scheduling Independent Tasks with Restricted Execution Times, *Operations Research*, Vol. 30, 1982.
32. H. F. Li, Scheduling Trees in Parallel/Pipelined Processing Environments, *IEEE Transactions on Computers*, November, 1977, pp. 1101–1112.
33. M. Livney and M. Melman, Load Balancing in Homogeneous Broadcast Distributed Systems, *Proceedings of the ACM Computer Network Performance Symposium*, 1982, pp. 47–55.
34. V. Lo and J. W. S. Liu, Task Assignment in Distributed Multiprocessor Systems," *International Conference on Parallel Processing*, August, 1981, pp. 358–360.
35. P. Y. R. Ma, E. Y. S. Lee, M. Tshchiya, A Task Allocation Model for Distributed Computing Systems, *IEEE Transactions on Computers*, C-31:41–47 (1982).
36. R. Mehrotra and S. N. Talukdar, Scheduling Tasks for Distributed Processors, *The 11th Annual International Symposium on Computer Architecture*, June, 1984, pp. 263–270.
37. C. L. Monma, Linear-Time Algorithms for Scheduling on Parallel Processors, *Operations Research*, 30:124–116 (1982).
38. L. M. Ni and K. Hwang, Optimal Load Balancing Strategies for a Multiple Processor System, *Proceedings of the International Conference on Parallel Processing*, August, 1981, pp. 352–357.
39. L. M. Ni and K. Hwang, Optimal Load Balancing in a Multiple Processor System with Many Job Classes, *IEEE Transactions on Software Engineering*, SE-11:491–496 (1985).
40. L. M. Ni, C. W. Xu, T. Gendreau, A Distributed Drafting Algorithm for Load Balancing, *IEEE Transactions on Software Engineering*, SE-11:1153–1161 (1985).
41. J. K. Ousterhout, D. A. Scelza, P. S. Sindhu, Medusa: An Experiment in Distributed Operating System Structure, *Communications of the ACM*, 23:92–105 (1980).
42. M. Pinedo, Stochastic Scheduling with Release Dates and Due Dates, *Operations Research*, Vol. 31, 1983.
43. C. V. Ramamoorthy, K. M. Chandy, M. J. Gonzalez, Jr., Optimal Scheduling Strategies in a Multiprocessor System, *IEEE Transactions on Computers*, C-21:137–146 (1972).
44. G. S. Rao, H. S. Stone, T. C. Hu, Assignment of Task in a Distributed Processor System with Limited Memory, *IEEE Transactions on Computers*, C-28:291–299 (1979).
45. K. Ramamritham and J. A. Stankovic, Dynamic Task Scheduling in Hard Real-Time Distributed Systems, *IEEE Software*, 1:65–75 (1984).
46. H. S. Stone and S. H. Bokhari, Control of Distributed Processes, *IEEE Computer*, July, 1978, pp. 97–105.
47. J. A. Stankovic, An Application of Bayesian Decision Theory to Decentralized Control of Job Scheduling, *IEEE Transactions on Computers*, C-34:117–130 (1985).
48. H. S. Stone, Multiprocessor Scheduling with the Aid of Network Flow Algorithms, *IEEE Transactions on Software Engineering*, SE-3:85–93 (1977).
49. H. S. Stone, Critical Load Factors in Two-Processor Distributed Systems, *IEEE Transactions on Software Engineering*, SE-4:254–258 (1978).
50. A. N. Tantawi and D. F. Towsley, Optimal Static Load Balancing in Distributed Computer Systems, *Journal of the ACM*, 32:445–465 (1985).
51. B. W. Wah and A. Hicks, Distributed Scheduling of Resources on Interconnection Networks, *National Computer Conference*, June, 1982, pp. 697–709.
52. B. W. Wah and J. Y. Juang, An Efficient Protocol For Load Balancing on CSMA/CD Networks, *Proceedings of the Eighth Conference on Local Computer Networks*, October, 1983, pp. 55–61.
53. B. W. Wah and J. Y. Juang, Resource Scheduling for Local Computer Systems with a Multiaccess Network," *IEEE Transactions on Computers*, December, 1985, pp. 1144–1157.
54. Y. T. Wang and R. J. T. Morris, Load Sharing in Distributed Systems," *IEEE Transactions on Computers*, C-34:204–217 (1985).
55. B. Walker, G. Popek, R. English, C. Kline, G. Thiel, The LOCUS Distributed Operating System, *Proceedings of the Ninth Annual Symposium on Operating System Principles*, 1983, pp. 49–70.
56. J. Welgarz, Multiprocessor Scheduling with Memory Allocation—A Deterministic Approach, *IEEE Transactions on Computers*, August, 1980, pp. 703–709.
57. E. Williams, Assigning Processes to Processors in Distributed Systems, *Proceedings of the International Conference on Parallel Processing*, 1983, pp. 404–406.
58. S. Zhou, A Trace Driven Study of Dynamic Load Balancing, *University of California—Berkeley Technical Report #UCB/CSD 87/305*, September, 1986.

Received 1 February 1990; revised 25 July 1990