A QUEUEING NETWORK MODEL FOR ANALYZING A CLASS OF BRANCH-AND-BOUND ALGORITHMS ON A MASTER–SLAVE ARCHITECTURE

O. J. BOXMA

CWI, Amsterdam, The Netherlands, and Tilburg University, Tilburg, The Netherlands

G. A. P. KINDERVATER

Erasmus University, Rotterdam, The Netherlands

(Received November 1987; revisions received May 1989, April 1990; accepted April 1990)

Partitioning methods lend themselves very well to implementation on parallel computers. In recent years, branch-and-bound algorithms have been tested on various types of architectures. In this paper, we develop a queueing network model for the analysis of a class of branch-and-bound algorithms on a master–slave architecture. The analysis is based on a fluid flow approximation. Numerical examples illustrate the concepts developed. Finally, related branch-and-bound algorithms are studied using a machine repair queueing model.

Parallel computers will make it possible to solve large problem instances in little time. In the field of combinatorial optimization, we may expect to benefit from parallelism especially for *hard* problems, for which no polynomial time algorithm is known. Using traditional sequential computers, we can only solve small problem instances to optimality. With the advent of parallel machines, the range of tractable problem instances will be extended enormously although, with bounded parallelism, we can never hope for a speedup from exponential to polynomial time algorithms.

Hard combinatorial problems are usually solved by some form of implicit enumeration of the set of feasible solutions. A widely used technique is *branch and bound*. Branch-and-bound algorithms generate search trees in which each node has to deal with a subset of the solution set. On a parallel computer, the processors can examine different parts of the search tree at the same time. This idea has been tested on various architectures; see, for example, Finkel and Manber (1987), Kindervater and Trienekens (1988), and Kindervater and Lenstra (1988). Often, parallel branch-and-bound algorithms exhibit an anomalous behavior. It may happen that P processors together are more than P times as fast as a single processor. This can be explained by the fact that a parallel search algorithm may find a good (or the optimal) solution much earlier than the corresponding sequential algorithm. Unfortunately, it can also be the other way around: Adding a processor may slow down the computation (Lai and Sahni 1984, Lai and Sprague 1985, 1986, Li and Wah 1986).

In this paper, we analyze the behavior of a class of branch-and-bound algorithms on a *master–slave* architecture. In such a parallel computer system, we have a central master processor which is connected to a number of slave processors. Since communication between two slave processors is only possible through the master processor, information circulating in the system is known by the master processor. This fact is both the strength and the weakness of the master–slave architecture: The master processor has full knowledge of the progress made by the solution process, but it may become a bottleneck if the information being sent becomes too much. We are interested in the performance of the master–slave architecture for branch-and-bound algorithms. We study the effect of changing the speed of the master and of the slaves, and of changing the number of processors. This is done via a queueing theoretic approach.

In the next section, we will describe the class of branch-and-bound algorithms under consideration. For this particular type of branch-and-bound algorithms, we consider two variants of implementing the parallel evaluation of nodes on a master-slave

Subject classifications: Computers/computer science: master–slave architecture. Programming: analysis of parallel branch-and-bound algorithms. Queues: fluid flow approximations.

0030-364X/91/3906-1005 $01.25

architecture. In Section 2, we give a queueing network model in which a slave processor evaluates a node, puts the results in a queue at the master processor, and immediately continues with a new node, sent by the master processor. This queueing model is analyzed by means of a fluid flow approximation in Section 3. The techniques developed are illustrated by some numerical examples in Section 4. Section 5 studies the second variant, where a slave processor receives a new node only after the master processor has consumed the slave's latest results. Here, the appropriate queueing system turns out to be a so-called machine repair model. The main conclusions are presented in Section 6.

In the past, many parallel branch-and-bound algorithms have been proposed and tested on various architectures. However, a theoretical analysis of the empirically observed performance has been lacking. The purpose of the present paper is to model an attractive class of branch-and-bound algorithms on a master-slave architecture, and to give a performance analysis of the model.

1. BRANCH AND BOUND

Many combinatorial problems can only be solved by some form of implicit enumeration of the set of feasible solutions. A well known technique is branch and bound. Branch-and-bound algorithms generate search trees in which each node has to deal with a subset of the solution set. A subproblem corresponding to a node is either solved directly, or its solution set is split and for each subset a new node is added to the tree. The process can be improved by computing a bound on the solution a node can produce. If this bound is worse than the best solution found so far, the node cannot produce a better solution and, hence, it can be excluded from further examination. The last issue to be clarified is the order in which the nodes are considered for evaluation. Hereto, the nodes are given a priority, determined by some heuristic function, and from among the available nodes the one with the highest priority is considered next for evaluation.

The observation that any two nodes, neither of which is an ancestor of the other, can be solved independently, provides a natural basis for the parallelization of branch-and-bound algorithms: An idle processor searches for an available, but not yet expanded node, evaluates this node, thereby possibly creating new nodes, and informs the other processors on newly found better solutions.

An appealing implementation on a master-slave architecture is the following. The master processor keeps track of the search tree generated so far, orders the nodes according to their priorities, and sends the node with the highest priority to a slave processor as one becomes idle. The slave processors evaluate the nodes they receive and send the results back to the master processor. In this implementation, the master processor has full knowledge of the search tree generated so far and can ensure that the "most promising" part of the search tree is examined by the slave processors. However, the master processor must have a high enough processing speed to handle the communication requests of the slave processors and to maintain the priority queue of available nodes. The effective speed of the master processor, therefore, not only depends on the average number of communication requests per time-unit, but also on the length of the priority queue. It is clear that a large priority queue may cause a dramatic slowdown of the master processor.

We consider two variants of this implementation. First, we model the variant where a slave processor puts its results in a queue in front of the master processor and immediately continues with a new node, already processed by the master processor. The benefits of this variant are clear: The slave processors are only idle if there are no nodes available for evaluation. However, if the number of nodes available for evaluation grows, the master processor becomes slower and, as a result, a long queue of nodes waiting to be processed by the master processor may form. This has the effect that valuable information may not reach the master processor in time and that the slave processors may be forced to do what turns out to be useless work.

The second variant avoids the possibility of a long queue in front of the master processor: A slave processor receives a new node only after the master processor has consumed the slave's latest results. The disadvantage here is that a slow master processor causes idleness of the slave processors.

The implementation of the first variant is not always possible due to hardware or software limitations. A number of experiments with the second variant have been reported. Trienekens (1989), for example, describes a successful implementation for the traveling salesman problem, but a similar experiment for the job shop scheduling problem was rather disappointing (Kindervater 1989). The outcome for both experiments can be explained from the fact that in the traveling salesman algorithm the time needed for the evaluation of a node by a slave processor is much longer than in the job shop program.

This paper aims to provide a theoretical background for experimental observations, such as the ones described above. We deal with the first variant in Sections 2, 3 and 4. The second variant is discussed in Section 5. Throughout the paper, we assume that there are enough nodes available for evaluation by the slaves. This is not a serious restriction because parallel computers are particularly useful for solving problem instances that require large search trees for finding the optimal solution.

2. QUEUEING MODEL DESCRIPTION

In the queueing network representation of the parallel processing of branch-and-bound nodes, the master processor is represented by a single server M, and the P slave processors are represented by P parallel servers $S_1, \ldots, S_P$, gathered in a service station S; cf. Figure 1. The nodes are represented by customers. In the figure, B&D symbolizes the birth and death of customers. This corresponds to the removal of processed nodes and the generation of new nodes. To further specify the queueing network, we have to describe the routing of customers and the service processes at M and S.

2.1. The Routing of Customers

When a customer arrives at M, he may have to wait in a queue until his service starts. After having obtained a service requirement, the customer leaves and immediately arrives at S, where he usually has to wait in a queue. In this queue, each customer has a priority that determines the order in which the customers are served by S. In the implementation of the branch-and-bound algorithms under consideration,

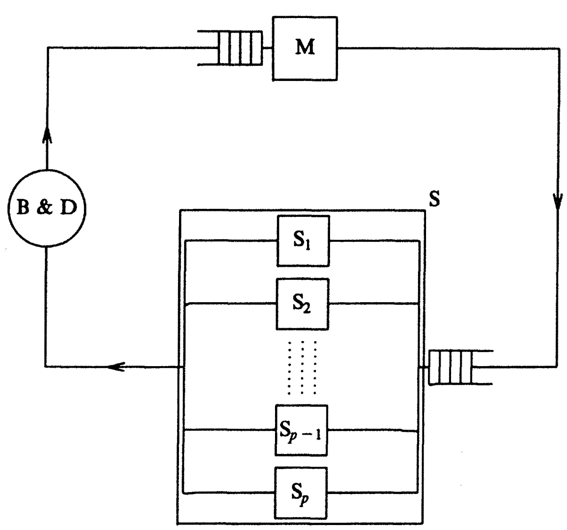

Figure 1. The queueing network model.

the priority queue is maintained by the master processor. In the queueing network model, however, it is more natural to identify the priority queue with the queue at service center S. Now there are two possibilities:

1. Before the customer is taken into service, the service center M receives information on which ground it decides to throw away a part of the queue at S, to which this customer belongs: The customer is instantaneously removed from the queueing network. This corresponds to the situation that the master obtains information from a node which makes the analysis of the nodes in a part of the priority queue obsolete. Customers who are thrown out of the queue at S are not replaced by other customers.
2. After a (possibly zero-length) waiting period, the customer is taken into service by one of the P servers; after having obtained the required service, the customer leaves S.

A customer who has successfully completed a service in S enters the part of the network designated in Figure 1 as B&D; here, he is immediately replaced by 0, 1 or 2 new customers, with probabilities p_0, p_1, p_2, respectively; $p_0 + p_1 + p_2 = 1$ (we assume that a branch-and-bound node has at most two descendants; the analysis to be presented in Section 3 remains valid when this assumption is relaxed). These new customers immediately arrive at M. The probabilities p_i may vary with time; we denote them by $p_i(t)$. The mean increase of the number of customers in the network after a service completion in S at time t will be denoted by

$$\phi(t) = p_1(t) + 2p_2(t) - 1. \qquad (1)$$

In approximation, $\phi(t)$ will be a decreasing function of t, with $\phi(0) = 1$ and $\phi(\infty) = -1$. In the branch-and-bound setting, this corresponds to the observation that the number of nodes generated by a node usually equals two in the beginning of a tree search, and that this number decreases to zero in the course of time. For most of the subsequent calculations, the exact form of $\phi(\cdot)$ is irrelevant.

2.2. The Service Process at M

The single server M serves customers in order of arrival (first-come, first-served). M's service of a customer consists of two parts:

1. a constant part of length a, which reflects the master's processing of the information contained in a node;

2. a part of length $b \ln(1 + y)$, which reflects the master's putting a node in a priority queue of size y. Note that insertion in a priority queue requires $O(\ln y)$ time units when its size is y.

Hence, the total service time of a customer in M, in the case that this customer has to be inserted in a priority queue of size y, equals

$$a + b \ln(1 + y).$$

Instead of constants, a and b may also be stochastic variables; in the analysis of this paper, that will turn out to be of minor importance.

In the following, the queue length of waiting customers in M at time t will be denoted by $\mathbf{y}_M(t)$.

2.3. The Service Process at S

When a server in S becomes idle, the customer at the front of the queue (if any) is immediately taken into service. When a newly arriving customer finds several servers idle, he chooses an arbitrary idle server. We assume that the P slave processors, and hence the P servers, are identical.

The service times of customers at S are independent, identically distributed stochastic variables with mean $1/\alpha$. Generally, it will not be necessary to specify the service time distribution at S further, but at a few places in the text we will consider the case of a negative exponential distribution.

Apparently the "capacity" of S is $P\alpha$: S is able to handle $P\alpha$ customers per unit of time, on the average. Throughout this paper we assume that $1/a \gg P\alpha$, i.e., M's maximum speed of handling customers is much higher than that of S. Of course, a large queue at S will slow down M considerably. The length of the queue at S at time t will be denoted by $\mathbf{y}_S(t)$.

Remark. In a parallel computer, communication takes a certain amount of time. We assume that the time to send messages between the master and the slaves has been taken into account in the service times.

3. MATHEMATICAL ANALYSIS OF THE NODE PROCESSING MECHANISM

In the previous section, a queueing network model was introduced to describe the node processing mechanism in parallel branch and bound. In this section, we present a mathematical analysis of the queue length processes in that queueing network. This analysis is basically of a nonstochastic nature. Of course, $\mathbf{y}_M(t)$ and $\mathbf{y}_S(t)$ are stochastic processes which may exhibit considerable fluctuations. Information concerning the (random) behavior of $\mathbf{y}_S(t)$ and $\mathbf{y}_M(t)$ requires a detailed queueing analysis. The problem of analyzing the transient behavior of queues is notoriously difficult, even when arrival and service rates are constant. In our case, a detailed mathematical analysis of the evolution of, say, $\mathbf{y}_M(t)$, requires analysis of the transient behavior of a single server queue with complex time dependent arrival and service rates. Hardly any results are available in the literature concerning such problems. Massey (1985) studies the asymptotic queue length behavior of an M/M/1 queue (i.e., a single server queue with Poisson arrival process and negative exponentially distributed service times) with time dependent arrival and service rates. Rider (1976) and Rothkopf and Oren (1979) derive approximations for the mean queue length at time t in this M/M/1 queue; their approximations are fairly complicated. These models are considerably less complex than the model under consideration, with its interaction between M and S. As there seems to be little hope of obtaining useful exact results, we have taken recourse to a standard type of approximation. The approximation, simple as it may be, will turn out to yield much insight into the behavior of both queue length processes. In queueing literature it is called a *fluid flow approximation* (cf. Newell 1971) or *deterministic fluid approximation* (cf. Newell 2nd ed., 1982).

Fluid flow approximations are based on the following observations: i) In a system with a large queue, many customers must arrive and depart before the queue changes much (in a relative sense). ii) In a period of time sufficiently long for many arrivals and departures to occur, the effect of random fluctuations—due to the stochastic nature of the arrival and service processes—will be relatively small. The latter observation can be supported theoretically by the Laws of Large Numbers and Central Limit Theorems. As an example for which detailed exact statements can be made, consider the departure process from the saturated service station S under the assumption that successive service times in S are independent, negative exponentially distributed stochastic variables with mean $1/\alpha$. Then successive departure intervals are independent, negative exponentially distributed stochastic variables with mean $1/P\alpha$. The number of departures, $D(t)$, in an interval of length t is Poisson distributed with mean $P\alpha t$ and variance $P\alpha t$. According to the Strong Law of Large Numbers

$$\frac{D(t) - \mathrm{E}[D(t)]}{\mathrm{E}[D(t)]} = \frac{D(t) - P\alpha t}{P\alpha t} \to 0, \quad t \to \infty \tag{2}$$

with probability one. Supplementary information is provided by the Central Limit Theorem, which shows that for large t

$$\Pr\left\{-y < \frac{D(t) - P\alpha t}{\sqrt{P\alpha t}} < y\right\}$$
$$\approx \frac{1}{\sqrt{2\pi}} \int_{-y}^{y} \exp(-x^2/2)\, dx. \tag{3}$$

Observations i and ii allow us to replace the discrete and random arrivals and departures at M and S by nonrandom continua (cf. Newell 1971, Chap. 2): We can view M and S as reservoirs, with fluids flowing in and out. In this setting, a reservoir can be *considered* to be empty for a lengthy period of time, without really being completely empty; it is empty only on a scale of measurement in which fluctuations in cumulative flows are negligible.

In our fluid flow analysis of the node processing mechanism, we distinguish two possible states in which the system can be, viz.

ME: M is empty;
MNE: M is not empty.

Once more, at a time t_0 at which the system is in state ME, $y_M(t_0)$ is not necessarily zero; but on the scale of measurement, it is negligible. Note that y is not printed in boldface, as the queue length process is no longer assumed to be stochastic.

Throughout the analysis, S will be considered to be nonempty, with all P servers being occupied. When $y_S(t)$ becomes zero, M serves so fast that S very soon saturates again. This is no longer true when there are hardly any customers in the system, but that situation is not of much interest.

For arbitrary customer increment rate $\phi(\cdot)$, the system state can switch back and forth between ME and MNE several times. In Subsection 3.1 we describe, in detail, the behavior of the queue length processes in each of these two states. In Subsection 3.2 we follow the evolution of $y_M(t)$ and $y_S(t)$ from beginning to end, in the case of a nonincreasing function $\phi(\cdot)$ with $\phi(0) = 1$ and $\phi(\infty) = -1$.

In Section 2 (see 2.1) we have mentioned an important feature of branch and bound: The master obtains information from a node which makes the analysis of the nodes in a part of the priority queue obsolete. In the queueing network setting, this corresponds to the situation that, upon departure of a customer from M, the tail of the queue at S is removed from the network: $y_S(t)$ instantaneously is reduced by a certain number. In describing the queue length processes in states ME and MNE, we first ignore such *sudden reductions of the queue at S.* In Subsection 3.3 we point out which simple changes are required to take reductions of the queue at S into account.

3.1. Queue Length Behavior in the States ME and MNE

We shall mainly concentrate on the queue length process $y_S(t)$; $y_M(t)$ follows from the relation

$$y_S(t) + y_M(t) = P\alpha \int_0^t \phi(u)\, du, \quad t \geqslant 0. \tag{4}$$

This relation holds for general $\phi(\cdot)$ under the assumption that the P slaves of S are always occupied, ignoring the possibility of a sudden reduction of the queue at S.

3.1.1. The State ME

In state ME, M is clearly nonsaturated: Its input rate is lower than its maximum possible processing rate. The output rate of S is $P\alpha$, all P servers being occupied; so the input rate to M, and accordingly the input rate to S, is $P\alpha(1 + \phi(t))$. Therefore, with t_0 the entrance time of the system in state ME is

$$y_S(t) = y_S(t_0) + P\alpha \int_{t_0}^{t} \phi(u)\, du$$
$$= P\alpha \int_0^t \phi(u)\, du. \tag{5}$$

The last equality follows from (4) because, by definition:

$$y_M(t) = 0$$

when the system is in state ME.

If $\phi(\cdot)$ is such that $y_S(\cdot)$ grows, this may slow down M so much that M becomes saturated; the system will switch to state MNE. The epoch at which the system changes from state ME to state MNE, t_1, is determined by the condition 'the flow into M equals the flow out of M' or

$$P\alpha(1 + \phi(t_1)) = [a + b \ln(1 + y_S(t_1))]^{-1}$$
$$= \left[a + b \ln\left(1 + P\alpha \int_0^{t_1} \phi(u)\, du\right)\right]^{-1} \tag{6}$$

with t_1 the smallest solution larger than t_0.

3.1.2. The State MNE

Suppose that, at a time t_1, the system enters state MNE. The server in M is now continuously busy; the input rate at M is still $P\alpha(1 + \phi(t))$, but its output rate—and the input rate to S—equals $[a + b\ln(1 + y_S(t))]^{-1}$. The queue length process $y_S(t)$ (or rather its fluid flow approximation) evolves according to the differential equation:

$$\frac{d}{dt}y_S(t) = -P\alpha + \frac{1}{a + b\ln(1 + y_S(t))}, \quad t \geq t_1. \tag{7}$$

The initial condition is determined by (6):

$$y_S(t_1) = P\alpha \int_0^{t_1} \phi(u)\,du$$
$$= \exp\left[\frac{1}{bP\alpha(1 + \phi(t_1))} - \frac{a}{b}\right] - 1. \tag{8}$$

The differential equation (7) plays a central role in our analysis of the queueing effects of the parallel processing mechanism. Rewrite (7) into

$$\int \frac{a + b\ln(1 + y_S)}{1 - P\alpha a - P\alpha b\ln(1 + y_S)}\,dy_S = \int dt$$

or, with C_1 some yet unknown constant:

$$-\frac{1}{P\alpha}y_S + \frac{1}{P\alpha}\int \frac{1}{1 - P\alpha a - P\alpha b\ln(1 + y_S)}\,dy_S$$
$$= t + C_1. \tag{9}$$

Introduce

$$C := \frac{1}{b}\left[\frac{1}{P\alpha} - a\right] \tag{10}$$

and the exponential integral (cf. Abramowitz and Stegun 1965)

$$\mathrm{E}_1(z) := \int_z^\infty \frac{\exp(-v)}{v}\,dv, \quad z > 0. \tag{11}$$

Substitution of $v = C - \ln(1 + y)$ in (11) shows that (9) can be rewritten as

$$-\frac{1}{P\alpha}y_S(t) + \frac{1}{(P\alpha)^2 b}e^C \mathrm{E}_1(C - \ln(1 + y_S(t)))$$
$$= t + C_1. \tag{12}$$

The initial condition determines the constant C_1:

$$-\frac{1}{P\alpha}y_S(t_1) + \frac{1}{(P\alpha)^2 b}e^C \mathrm{E}_1(C - \ln(1 + y_S(t_1)))$$
$$= t_1 + C_1. \tag{13}$$

Subtraction of the relations (12) and (13) finally gives us a relation between $y_S(t)$ and t:

$$-\frac{1}{P\alpha}[y_S(t) - y_S(t_1)]$$
$$+ \frac{1}{(P\alpha)^2 b}e^C[\mathrm{E}_1(C - \ln(1 + y_S(t)))$$
$$- \mathrm{E}_1(C - \ln(1 + y_S(t_1)))]$$
$$= t - t_1. \tag{14}$$

It seems impossible to find an explicit expression for $y_S(t)$ as a function of t, $t \geq t_1$, but (14) is already very useful. First, for each given value of $y_S(t)$ it is easy to explicitly calculate the corresponding t-value (the exponential integral $\mathrm{E}_1(\cdot)$ is extensively tabulated (Abramowitz and Stegun). Second, standard knowledge about $\mathrm{E}_1(\cdot)$ allows us to obtain useful insight into the behavior of $y_S(t)$.

It is clear from the differential equation (7) that, independently of the choice of $\phi(\cdot)$, $y_S(t)$, $t \geq t_1$, increases as long as this differential equation holds, tending to the limit $\exp(C) - 1$. Let us now study the following question: At what time t_ϵ will $y_S(t) + 1$ reach the level $\exp(C(1 - \epsilon))$? According to (14):

$$-\frac{1}{P\alpha}[\exp(C(1 - \epsilon)) - 1 - y_S(t_1)]$$
$$+ \frac{1}{(P\alpha)^2 b}e^C[\mathrm{E}_1(\epsilon C) - E_1(C - \ln(1 + y_S(t_1)))]$$
$$= t_\epsilon - t_1. \tag{15}$$

Now we use the fact that (Abramowitz & Stegun)

$$\mathrm{E}_1(z) = -\gamma - \ln z - \sum_{n=1}^{\infty} \frac{(-1)^n z^n}{nn!}, \quad z > 0 \tag{16}$$

with $\gamma = 0.57721\ldots$ denoting Euler's constant. Hence

$$\mathrm{E}_1(\epsilon C) = -\gamma + \ln\frac{1}{\epsilon C} + O(\epsilon), \quad \epsilon \to 0 \tag{17}$$

so

$$t_\epsilon \approx \frac{1}{(P\alpha)^2 b}e^C\left(\ln\frac{1}{\epsilon} + O(1)\right), \quad \epsilon \to 0. \tag{18}$$

These calculations enable us to estimate the behavior of $y_S(t)$ close to its limiting value. In particular, from comparing t_ϵ with $t_{2\epsilon}$, it follows that an $O(\epsilon)$ increase of $y_S(t)$ in this time region requires $O(1)$ time (one can, in fact, also derive this directly from the differential equation (7)). More precisely, from (14) and (17) one can show that for $t \to \infty$, $y_S(t) = -1 + e^C[1 - \exp[-(P\alpha)^2 be^{-C}t]]$. If $\phi(t) = 1$ in close approximation in a large time span in state MNE, $y_S(t) + y_M(t)$ grows linearly with $P\alpha$ customers per unit of time. Therefore, when $y_S(t)$ is close to its limiting value, the queue at M grows linearly with time in the time region under consideration.

The queue length process $y_M(t)$ follows from (4) once $y_S(t)$ has been determined. It depends on the choice of $\phi(\cdot)$ and of the various parameters whether a situation as sketched above (with the bulk of the growth of the customer population contributing to $y_M(t)$) actually occurs. See also the numerical examples in Section 4.

For the system to switch back to state ME, it is required that M's input rate $P\alpha(1 + \phi(t))$ is less than its output rate $[a + b \ln(1 + y_S(t))]^{-1}$ for some period of time. Let us suppose that $\phi(\cdot)$ and the various parameters are such that the system switches back to state ME. The epoch at which the system switches from state MNE to state ME, t_2, is determined by the condition $y_M(t_2) = 0$, or equivalently:

$$y_S(t_2) = P\alpha \int_0^{t_2} \phi(u)\, du.$$

Substitution in (14) yields:

$$\begin{aligned} &-\int_{t_1}^{t_2} \phi(u)\, du \\ &+ \frac{1}{(P\alpha)^2 b} e^C \left[E_1\left(C - \ln\left(1 + P\alpha \int_0^{t_2} \phi(u)\, du\right)\right)\right. \\ &\qquad \left. - E_1\left(C - \ln\left(1 + P\alpha \int_0^{t_1} \phi(u)\, du\right)\right)\right] \\ &= t_2 - t_1 \qquad (19) \end{aligned}$$

with t_2 the smallest solution, larger than t_1, of this equation. It has to be determined numerically.

3.2. Evolution of the Queue Length Processes

We now restrict ourselves to the case of a nonincreasing function $\phi(\cdot)$ with $\phi(0) = 1$ and $\phi(\infty) = -1$. We follow the evolution of $y_M(t)$ and $y_S(t)$ from beginning to end.

Initially, there is only one customer in the system (the root of the search tree). This customer is served in M, and subsequently in S; it is replaced by two new customers, who arrive at M; shortly thereafter there are three customers, etc. Very soon all processors of S are continuously busy. If, e.g., all service times at S are negative exponentially distributed with mean $1/\alpha$ and M is much faster than the P processors, the length of the initial period is approximately

$$\frac{1}{\alpha} + \frac{1}{2\alpha} + \ldots + \frac{1}{(P-1)\alpha};$$

(indeed, when j servers are active in S, the time until the first departure from S is negative exponentially distributed with mean $1/j\alpha$; the departing customer is almost certainly replaced by two other customers, who—after a very short visit to M—increase the number of active servers in S to $j + 1$). After the initial period, $P\alpha$ customers leave S per unit of time (on the average), and $P\alpha(1 + \phi(t))$ customers arrive at M per unit of time. M is extremely fast as long as the queue length at S, $y_S(t)$, is not too large: M has at first no difficulty handling its input stream, so its output stream also has the intensity $P\alpha(1 + \phi(t))$. In the fluid flow approach, M is still considered to be empty: The system is still in state ME. $y_S(t)$ grows at a rate $P\alpha\phi(t)$, cf. (5). There are now two possibilities:

i. M slows down so much that its maximal output rate equals its input rate: M starts to saturate, and the system enters state MNE;
ii. M's speed is not reduced enough to reach the saturation point, and all customers are being processed without the system ever entering state MNE.

Case i obviously is the more interesting one. The system enters state MNE. The queue length process $y_S(t)$ now evolves according to the differential equation (7). M's queue length initially grows but, as a counteracting force, $\phi(t)$ decreases; finally, the input rate $P\alpha(1 + \phi(t))$ in M becomes lower than the output rate and M's queue length starts to decrease. This process continues until M becomes empty again: The system switches back to state ME.

At this epoch, the input rate at S switches to $P\alpha(1 + \phi(t))$. If $\phi(\cdot)$ has already become negative, the queue length at S immediately starts to decrease, and continues to do so ($\phi(\cdot)$ being a nonincreasing function). Consequently, M speeds up, and the system stays in state ME until there are no customers left. However, if $\phi(\cdot)$ still is positive, then in principle both possibilities, i and ii, discussed above again exist, and the system may switch back to MNE, etc. Such an alternating series of states ME and MNE may, for example, occur if shortly after entering state MNE the function $\phi(\cdot)$ drops from almost one to a small positive value and keeps this value for a substantial period. The system will react by a change from state MNE to state ME, and because the number of customers is still growing M will get saturated once more. Figure 2 depicts the typical behavior of M's input rate $P\alpha(1 + \phi(t))$ and its service speed $[a + b \ln(1 + y_S(t))]^{-1}$.

3.3. Reductions of the Queue at S

Neither Figure 2, nor the global description of the queue length processes, considers the phenomenon that part of the queue at S is instantaneously thrown out of the system. This phenomenon, which also implies a sudden increase of M's speed, can easily be captured in the mathematical analysis. Suppose that a reduction of the queue at S occurs at an epoch t_d, and that x customers are removed from the network. If this happens while the system is in state ME, the output rate, $P\alpha(1 + \phi(t_d))$, of M is not affected. Much more interesting is the situation in which the sudden drop in the queue length of S occurs while the system is in state MNE. Instantaneously the output rate of M increases to

$$[a + b \ln(1 + y_S(t_d +))]^{-1}$$
$$= [a + b \ln(1 + y_S(t_d -) - x)]^{-1}.$$

The queue length at S still behaves according to the differential equation (7), but with a new initial value $y_S(t_d +)$. The speedup of M may soon lead to an empty queue at M, so that the system enters state ME. Of course, it is possible that *several* considerable reductions of the queue at S occurs. Not much is known about the frequency with which this phenomenon occurs, nor about the sizes (x) of the corresponding jumps. Therefore we do not discuss the issue in much detail here. It suffices to observe that our model is able to determine the influence of sudden reductions of the queue at S on the speed of the master, and on the subsequent behavior of the queue sizes.

In Section 4 we present some numerical examples which, for various choices of the function $\phi(\cdot)$ and the parameters P, α, a and b, exhibit the global behavior of $y_S(t)$ and $y_M(t)$. In one example, the phenomenon of a reduction of the queue at S is also taken into account.

Remark. At this stage, it seems appropriate to cite Newell's cautionary note concerning the use of fluid flow approximations (Newell 1982, p. 36): "Any conclusions obtained here are tentative and subject to unknown errors arising from the use of deterministic approximations. One must be particularly cautious of the possibility that the queue lengths calculated here may be overshadowed by queues generated by stochastic effects." Although we do not expect such an overshadowing effect here, there may be periods during which the queues are small and the stochastic behavior is dominating.

Remark. $\phi(\cdot)$ has so far been considered as a process-independent function. In reality, $\phi(\cdot)$ may depend on the queue length process; it might, in particular, be realistic to decrease $\phi(\cdot)$ after the occurrence of a sudden reduction of the queue at S as described before (and this decrease should be related to the size of the reduction). Such process-dependent behavior of $\phi(\cdot)$ can be incorporated in the model. The behavior of $y_S(t)$ initially would still be determined by the differential equation (7), but the input rate in M would suddenly decrease.

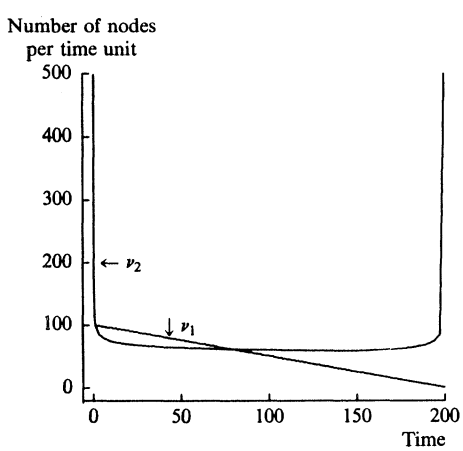

Figure 2. M's input rate v_1 and service speed v_2 = ($v_1 = P\alpha(1 + \phi(t))$ and $v_2 = [a + b \ln(1 + y_S(t))]^{-}$); $P\alpha = 50$, $a = b = 0.0020$, and $\phi(\cdot)$ is linearly decreasing.

4. NUMERICAL EXAMPLES

To give a global idea of the behavior of $y_S(t)$ and $y_M(t)$, we will now present the results of some numerical computations. In all cases, we consider a linearly decreasing function $\phi(\cdot)$ with $\phi(0) = 1$. The process stops at a time T with $\phi(T) = -1$. The total number of customers served by the slaves at time T is $P\alpha T$. In all examples, we chose this number to be 10,000.

In Figure 3, the case $P\alpha = 50$ and $a = b = 0.0020$ is shown (see also Figure 2, and note that P and α are occurring as a product in all formulas). In the beginning, y_S is increasing very fast and M is getting saturated almost immediately. At that moment, the queue length y_M starts to grow. Since $\phi(\cdot)$ is a decreasing function, the number of customers arriving at M is

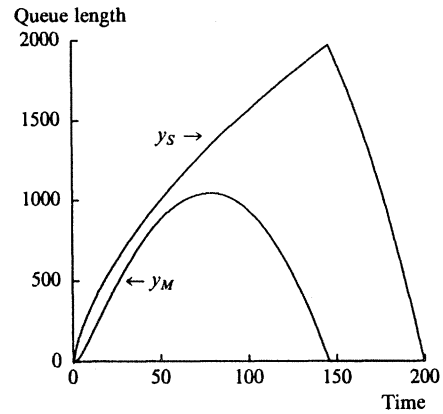

Figure 3. y_S and y_m for $P\alpha = 50$ and $a = b = 0.0020$.

decreasing. Therefore, M will eventually become empty and the system changes from state MNE to state ME. At this point in time, y_S starts to decrease since $\phi(\cdot)$ is already negative.

Figure 4 shows the effect of changing $P\alpha$, which corresponds to altering the number of slaves or the processing speed of the slaves. For $P\alpha = 20$, the master is fast enough to serve the incoming customers and $y_M \approx 0$. If $P\alpha = 80$, the master gets into serious trouble. The speed of the master is much too slow compared with the number of incoming customers. Here, we can observe the fact that y_S is approaching an asymptotic value if the system is in state MNE for a long enough period.

There appears to be a delicate interaction between the processing capacities of the master and the slaves. Increasing the processing capacity of the slaves may change an almost continuously idle master into a saturated master with a very long queue. The beneficial effect of increasing the processing capacity of the slaves may now be reduced; for example, a node with information that would make a large part of the priority queue obsolete (i.e., a part of the queue at S would be thrown away) is delayed for a long time, thus possibly causing a deterioration of the running time of the algorithm.

In Figure 5, we consider different speeds of the master. The effects are about the same as when changing $P\alpha$.

Sudden reductions of the queue at S may cause an alternating sequence of the states ME and MNE. An example is given in Figure 6. In state MNE, a part of the queue at S is thrown away. As a consequence, M's speed increases so much that y_M becomes zero. Since the total number of customers in the system is still increasing rapidly, M gets saturated again, and the system enters state MNE again.

5. THE MACHINE REPAIR MODEL

For the class of branch-and-bound algorithms considered in this paper, it can be advantageous that the master has full knowledge of the search tree developed so far. An enormous queue length at the master can cause a slowdown of the computation. Therefore, in this section we consider branch-and-bound algorithms where a slave does not start with the evaluation of a new node until the master has processed the latest information the slave has sent.

This gives rise to the queueing model of Figure 7, with exactly P customers, each customer corresponding to one particular slave. This is a well known queueing model, often referred to as the machine repair model (the P customers being P machines which after breakdown have to be repaired in repair facility M). In a computer context, the model also represents a multiaccess system (Kobayashi 1978). In such a case, the P slaves correspond to P terminal users. Each of these terminal users alternates between an active

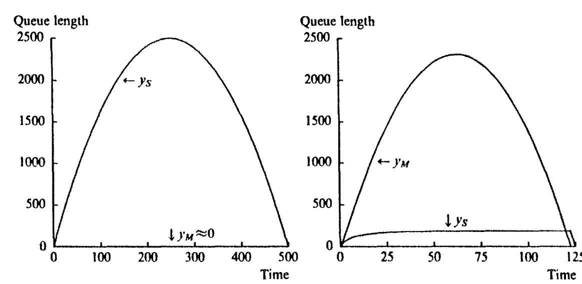

Figure 4. The effect of changing $P\alpha$; $a = b = 0.0020$, $P\alpha = 20$ (left), $P\alpha = 80$ (right).

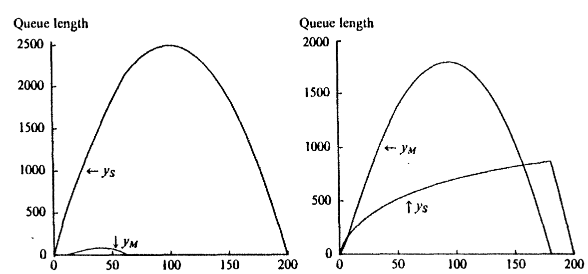

Figure 5. The effect of changing a and b; $P\alpha = 50$, $a = b = 0.0015$ (left), and $a = b = 0.0025$ (right).

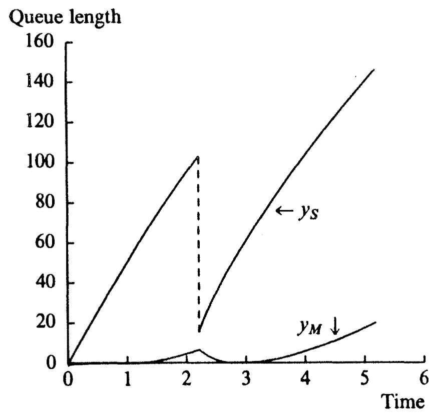

Figure 6. An example with a reduction of the queue at S; $P\alpha = 50$ and $a = b = 0.0020$.

(think) phase and a passive phase; after a think phase, a job is sent to the central processor M.

The machine repair model has been studied extensively in the queueing literature (see, for example, Gross and Harris 1985, Kobayashi 1978, and Tijms 1986). Hardly any time-dependent results are known; however, under some distributional assumptions, quite simple explicit formulas for the *steady-state* queue length distribution at the repair facility, the mean number of busy machines, etc., have been derived. In this section, we shall use those steady-state results to obtain insight into the performance of the master-slave architecture for branch-and-bound algorithms where a slave processor only receives a new node when the master has consumed the slave's latest results.

As in the previous model, we assume that the service times at the P servers $S_1, \ldots, S_P$ of service station S are independent, identically distributed with mean $1/\alpha$. The assumptions concerning the service process in M differ from those in the previous model. For reasons of mathematical tractability, it is assumed that the service times at M are independent, negative exponentially distributed stochastic variables, with mean $1/\beta$. Note that the fluid flow approximation of Section 3 allowed us to leave the service time distribution at M unspecified. We return to this issue in the remark at the end of this section. The rate of the service times at the master is further assumed to be constant in time. However, the steady-state analysis for the case of constant master speed that we are about to present will yield insight into the effect that a change in speed of the master has (see Figure 8 below).

It is easy to see that under the assumption of negative exponentially distributed service times at M, S is equivalent—with respect to the number of busy servers—to the so-called M/G/P loss model. This is an open queueing model with a Poisson arrival process, P servers with generally distributed service times, and no waiting room; an arriving customer who finds all servers occupied is lost. Indeed, as long as the loss model and S contain less than P customers, their number of customers evolve in exactly the same way. When the loss model contains P customers, no more arrival is accepted until a customer has left; after an exponential period of time, a new arrival takes place. But exactly the same situation occurs in S, when it contains all P customers.

We restrict ourselves to the consideration of the limiting probability distribution of the number of busy servers, $\mathbf{B}$, at S (which number equals P minus the number of customers in M). This amounts to studying the limiting distribution of the number of busy servers in the M/G/P loss model. This limiting distribution, and hence the distribution of $\mathbf{B}$, is given by (see, e.g., Kelly 1979, pp. 13, 79, or Tijms 1986, pp. 290, 291):

$$p_n := \Pr\{\mathbf{B} = n\} = \frac{r^n/n!}{\sum_{j=0}^{P} r^j/j!}, \qquad n = 0, 1, \ldots, P \quad (20)$$

with

$$r := \beta/\alpha.$$

The probability that an arriving customer in the M/G/P loss model is lost, $E_P(r)$, is given by Erlang's loss formula:

$$E_P(r) = P_P = \frac{r^P/P!}{\sum_{j=0}^{P} r^j/j!}. \quad (21)$$

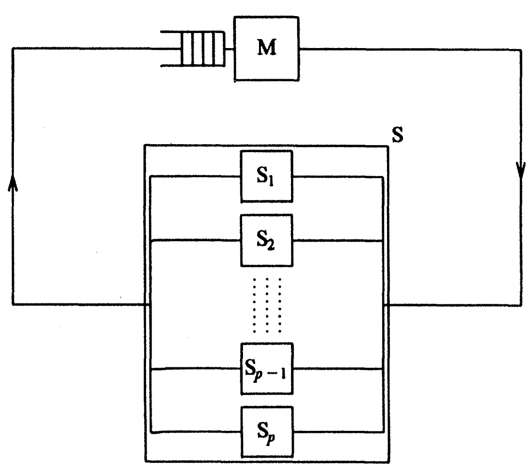

Figure 7. The machine repair model.

The mean number of busy servers at S, N, follows from (20):

$$N := \mathrm{E}[\mathbf{B}] = r[1 - \mathrm{E}_P(r)]. \tag{22}$$

The relation between N and $\mathrm{E}_P(r)$ is easily interpreted. Indeed, with r the amount of traffic offered to the M/G/P loss system per unit of time, N equals the mean amount of traffic handled per unit of time—and this should equal the mean number of busy servers. In this connection, note that αN represents the throughput of S, and hence also of M; so the mean cycle time of a job in the closed system is given by $P/\alpha N$.

In principle, (22) can be evaluated numerically. However, this evaluation may be cumbersome when P and/or r are large while, moreover, (21) does not yield much insight. Therefore, the behavior of N and $\mathrm{E}_P(r)$ for large values of P and/or r has been investigated extensively. See Whitt (1984) for an interesting exposition and several early references, and Newell (1984) for various asymptotic expansions. In particular, Newell presents a simple first-order approximation for $\mathrm{E}_P(r)$ for $r \to \infty$, leading to

$$\begin{aligned} N &\approx r, \quad r \leq P, \\ N &\approx P, \quad r > P. \end{aligned} \tag{23}$$

Newell's second-order approximation (see also Whitt) leads to the following approximation for N. Introduce

$$\kappa := \frac{r}{\sqrt{P}}\left(\frac{P}{r} - 1\right)$$

and the standard normal distribution function

$$\Phi(x) := \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)\, dz, \quad -\infty < x < \infty.$$

The mean number of busy servers in S is for large values of r approximated by:

$$\begin{aligned} N &\approx r\left[1 - \frac{(r/P)^{P-1} e^{P-r}}{\sqrt{2\pi P}\ \Phi(\kappa P/r)}\right], \quad r \leq P \\ N &\approx r\left[1 - \left(\frac{P}{2\pi}\right)^{1/2} \frac{\exp(-\kappa^2/2)}{r\Phi(\kappa)}\right], \quad r > P. \end{aligned} \tag{24}$$

This approximation is based on Stirling's approximation for factorials, and the normal approximation to the Poisson distribution.

Figure 8 displays the exact fraction of busy servers in S, N/P, as a function of $r = \beta/\alpha$ for $P = 1, 2, 4, 8, 16, 32, 64$ and 128. The figure clearly shows the usefulness of the simple first-order approximation (23). N grows linearly with β/α until the speed of the master M, β almost equals $P\alpha$, the maximal speed of

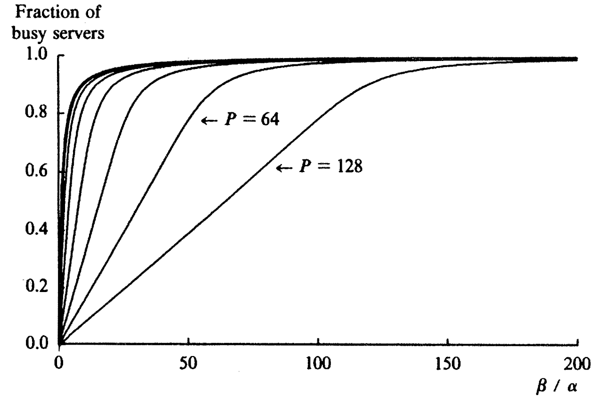

Figure 8. Fraction of busy servers as a function of β/α for $P = 1, 2, 4, 8, 16, 32, 64, 128$.

S; further increasing β has hardly any effect. In branch-and-bound algorithms, the speed of the master varies with the number of generated but not yet examined nodes. The effect of such fluctuations in the speed of the master processor on the fraction of busy servers can also be derived from Figure 8.

Figure 9 displays the fraction N/P as a function of $r/P = \beta/P\alpha$ for the same parameter choices as in Figure 8. The figure shows that for $P > r$ ($\beta/P\alpha < 1$) the fraction of busy servers decreases rapidly when $\beta/P\alpha$ decreases. For fixed speeds of the master and the slaves, it is, therefore, only worthwhile to add slave processors as long as $P < r$ ($\beta/P\alpha > 1$).

So far, we have been concerned mainly with the *mean* of the number of busy servers in S. Newell (1984) also presents approximations for the *distribution* of the number of busy servers in S. He states that, for P large and fixed, and $r > P$, and in particular $1 - P/r \gg r^{-1/2}$, the distribution of idle servers in S is approximately geometric:

$$\Pr\{n \text{ idle servers}\} = p_{P-n} = (1 - P/r)(P/r)^n \qquad n = 0, 1, \ldots, P \tag{25}$$

with the mean number of idle servers in S approximately equal to $P/(r - P)$.

Remarks. In this section, the service times at S are generally distributed, whereas the service times at M are exponentially distributed. It is an interesting and well known fact for the machine repair model (and the M/G/P loss model) that (20) for the number of customers at S holds regardless of the form of the service time distribution at S. When M uses a processor-sharing discipline, (20) even holds when the service times at M have a general distribution with

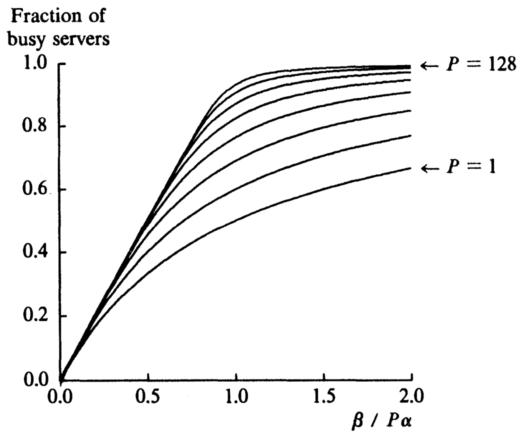

Figure 9. Fraction of busy servers as a function of $\beta/P\alpha$ for $P = 1, 2, 4, 8, 16, 32, 64, 128$.

mean $1/\beta$ (cf. Tijms, p. 291). For the first-come first-serve discipline at M under consideration, this *insensitivity* for the service time distribution at M is not true.

Formula (20) can easily be generalized to the case that the mean service time in M depends on the number of customers waiting in M, or equivalently, that the arrival rate at the M/G/P loss system depends on the number of busy servers. Let β_n denote the service speed in M when n customers are present in M. Then (20) should be replaced by

$$p_n := \Pr\{\mathbf{B} = n\} = \frac{\prod_{k=1}^{n} (\beta_{P-k+1}/k\alpha)}{\sum_{j=0}^{P} \prod_{k=1}^{j} (\beta_{P-k+1}/k\alpha)}, \quad n = 0, 1, \ldots, P. \tag{26}$$

6. CONCLUSIONS

The queueing network model developed in this paper allows us to analyze the behavior of a class of branch-and-bound algorithms on master-slave architectures. The main performance measures under consideration are the number of customers at the master and at the slaves. These are natural performance measures because they determine the effectiveness of an implementation. For both variants, we have studied the influence of changing the speed of the master and of the slaves, and of changing the number of slave processors.

For the case where a slave starts evaluating a new node as soon as it becomes idle (Section 3), the state of the system can be determined completely at any point in time. We have shown that there is a delicate interaction between the processing capacities of master and slaves. For example, increasing the speed of the slave processors or adding extra slave processors may turn an almost continuously idle master into a saturated master with a large queue. The resulting long delay, at the master, of nodes with valuable information may counteract the beneficial effect of increasing the processing capacity of the service station S. The "best" situation is probably the one in which the master never saturates. Given the speed of the master and of the slaves and a function $\phi(\cdot)$, one may use (6) to obtain the highest number of slave processors, P_0, such that a saturation of the master never occurs, i.e.,

$$P_0\alpha(1 + \phi(t)) < \left[a + b \ln\left(1 + P_0\alpha \int_0^t \phi(u)\, du\right)\right]^{-1} \quad \text{for all } t.$$

For the variant of Section 5 where a slave starts evaluating a new node only after the master has processed the slave's latest results, we can only give a steady-state analysis. Still, this analysis yields useful insight. Increasing the speed of the master, β, to enhance that the slaves are almost always busy, is only useful as long as $\beta \leq P\alpha$, the total processing capacity of the slaves; similarly, increasing the number of slaves, P, is only useful as long as $P\alpha \leq \beta$.

Finally, given system parameters such as the speed of the master and of the slaves, and given branch-and-bound parameters, such as the function $\phi(\cdot)$ and the average number of steps needed for the processing of a branch-and-bound node, the queueing network model allows us to decide whether the two discussed implementations can be used effectively.

REFERENCES

ABRAMOWITZ, M., AND I. A. STEGUN. 1965. *Handbook of Mathematical Functions.* Dover, New York.

FINKEL, R., AND U. MANBER. 1987. DIB-A Distributed Implementation of Backtracking. *ACM Trans. Prog. Languages and Systems* **9**, 235–256.

GROSS, D., AND C. M. HARRIS. 1985. *Fumdamentals of Queueing Theory*, 2nd ed. John Wiley, New York.

KELLY, F. P. 1979. *Reversibility and Stochastic Networks.* John Wiley, New York.

KINDERVATER, G. A. P. 1989. Exercises in Parallel Combinatorial Computing. Ph.D. Thesis, Centre for Mathematics and Computer Science, Amsterdam.

KINDERVATER, G. A. P., AND J. K. LENSTRA. 1988. Parallel Computing in Combinatorial Optimization. *Ann. Opns. Res.* **14**, 245–289.

KINDERVATER, G. A. P., AND H. W. J. M. TRIENEKENS. 1988. Experiments With Parallel Algorithms for Combinatorial Problems. *Eur. J. Opnl. Res.* **33**, 65–81.

KOBAYASHI, H. 1978. *Modeling and Analysis*. Addison-Wesley, Reading, Mass.

LAI, T.-H., AND S. SAHNI. 1984. Anomalies in Parallel Branch-and-Bound Algorithms. *Comm. ACM* **27,** 594–602.

LAI, T.-H., AND A. SPRAGUE. 1985. Performance of Parallel Branch-and-Bound Algorithms. *IEEE Trans. Comput.* **C-34,** 962–964.

LAI, T.-H., AND A. SPRAGUE. 1986. A Note on Anomalies in Parallel Branch-and-Bound Algorithms With One-to-One Bounding Functions. *Inform. Process. Lett.* **23,** 119–122.

LI, G.-J., AND B. W. WAH. 1986. Coping With Anomalies in Parallel Branch-and-Bound Algorithms. *IEEE Trans. Comput.* **C-35,** 568–573.

MASSEY, W. A. 1985. Asymptotic Analysis of the Time Dependent M/M/1 Queue. *Math. Opns. Res.* **10,** 305–327.

NEWELL, G. F. 1971. *Applications of Queueing Theory*. Chapman and Hall, London.

NEWELL, G. F. 1982. *Applications of Queueing Theory*, 2nd ed. Chapman and Hall, London.

NEWELL, G. F. 1984. *The M/M/∞ Service System With Ranked Servers in Heavy Traffic*. Springer Verlag, Berlin.

RIDER, K. L. 1976. A Simple Approximation to the Average Queue Size in the Time-Dependent M/M/1 Queue. *J. Assoc. Comput. Mach.* **23,** 361–367.

ROTHKOPF, M. H., AND S. S. OREN. 1979. A Closure Approximation for the Nonstationary M/M/s Queue. *Mgmt. Sci.* **25,** 522–534.

TIJMS, H. C. 1986. *Stochastic Modeling and Analysis*. John Wiley, New York.

TRIENEKENS, H. W. J. M. 1989. Computational Experiments With an Asynchronous Parallel Branch and Bound Algorithm. Report EUR-CS-89-02, Department of Computer Science, Erasmus University, Rotterdam.

WHITT, W. 1984. Heavy-Traffic Approximations for Service Systems With Blocking. *AT&T Bell Labs. Tech. J.* **63,** 689–708.