A Study of Storage Partitioning Using a Mathematical Model of Locality

E.G. Coffman Jr. and Thomas A. Ryan Jr.
The Pennsylvania State University*

Both fixed and dynamic storage partitioning procedures are examined for use in multiprogramming systems. The storage requirement of programs is modeled as a stationary Gaussian process. Experiments justifying this model are described. By means of this model dynamic storage partitioning is shown to provide substantial increases in storage utilization and operating efficiency over fixed partitioning.

Key Words and Phrases: storage partitioning, memory management, dynamic storage allocation, space sharing, multiprogrammed storage, working-sets, program behavior models, mathematical modeling

CR Categories: 4.32, 4.39

*Department of Computer Science, 426 McAllister Hall, University Park, PA 16802.

Presented at the Third ACM Symposium on Operating Systems Principles, Palo Alto, California, October 18–20, 1971.

1. Introduction

We are concerned in this paper with multiprogramming systems in which, because of the locality exhibited by address referencing patterns and the desire to maximize the degree of multiprogramming, programs are executed without being fully resident in main memory. The design problem of principal interest is the strategy according to which main storage is partitioned among active processes. We feel (along with others) that at the present time this problem is of equal if not greater importance than the problem of selecting suitable page or segment replacement rules for these systems.

Our approach to the above problem is a mathematical one. Consequently, our point of departure includes, among other things, a model of program behavior in terms of the well-known property of *locality* [1] in address referencing. After various means for describing locality are discussed, such a model will be proposed and characterized at some length. Because of its importance and its applicability to other studies, we shall take some pains in justifying the model we have selected.

The locality model proposed will be exploited in a comparison of two basic strategies for partitioning storage: the *fixed partition* strategy whereby the (maximum) degree of multiprogramming is fixed and storage is partitioned equally among the active programs, each of which is strictly confined to its block of the partition; and a *dynamic partitioning* strategy whereby the space occupancy of programs is determined by an appropriately defined "working-set" process. The general result of this study is that, even with moderate variation in the locality exhibited by executing programs, dynamic partitioning leads to a significant improvement in storage utilization (and paging rates). This statement is quantified in Section 3.

Past work on this design problem is limited. For a

general reference dealing extensively with the context of this problem we recommend Denning [1]. Oden and Shedler [2] present a mathematical model of multiprogrammed memory contention under the specific assumption of LRU (least recently used) replacement rules and subject to a given strategy for selecting programs from which to obtain replacement pages. Basically, their mathematical approach is at a different (more detailed) level than the approach used in this paper. Our approach, which is not constrained by the above types of assumptions, resembles that of Gaver and Lewis [3] in a similarly motivated study of buffer design problems. With the methods we use, general results are more easily obtained. On the other hand, these results depend on a model using broader assumptions concerning program behavior; consequently, we must (and shall) verify that these assumptions are in fact reasonable.

2. Description of Locality

To be used most effectively, a description of locality should be based, if possible, on a performance measure related to storage utilization, e.g. paging rates. As a result, in a demand paging system[1] a desirable measure of the (addressing) locality of a program just after the ith page reference would take the form of a minimal set $W_p(i)$ of pages such that the probability of referencing outside this set on the $(i + 1)$-st reference is less than or equal to p, and such that the sequence of values $W_p(1), W_p(2), \ldots$ is consistent with demand paging. In practice such a measure would not generally be computable (sufficient information is not available), nor would it necessarily be possible to formulate such a measure for all p in a way that is consistent with demand paging.

Therefore it is natural to consider definitions whereby a deterministic criterion (e.g. recency or frequency of use) is used to establish the "locality" set membership so as to ensure the desired paging rate. Of course, whether or not the desired paging rate obtains depends on the existence of the page reference patterns anticipated by the criterion chosen. An example of this type of definition for paging systems is the so-called *working-set* model of locality whose properties have been studied in some detail by Denning [4]. The working set at time t is simply the set of distinct pages referenced in the l immediately preceding page references, where l is a parameter of the model.

[1] Although the models and definitions in this paper need not be restricted to paging systems, these systems provide a convenient and appropriate source of illustrations. Thus in most instances the presentation in the paper will be specialized to this application.

It will be clear subsequently that our model of the time dependent behavior of working-set sizes is not tied to a specific definition of working sets. However, in the next section we show that the above definition is consistent with the proposed mathematical model. As a result, since the (Denning) working-set model of program behavior has also been the locality model most studied to date, we shall assume this definition when necessary to fix ideas. In any case, in order that our subsequent results have operational meaning it is necessary for the general requirements mentioned earlier to be satisfied. In particular, a working-set definition must be well conceived in the sense that having a working set in main storage implies an acceptable paging rate for the corresponding program.

3. The Probability Model of Locality

Consider the execution of a single program and let $X(t)$ denote the corresponding working-set size as a function of time. Our basic model represents $X(t)$ as a *stationary*, *normal* (or *Gaussian*) stochastic process [5]. The stationarity assumption implies that the distribution function for working-set size

$$F(x) = \Pr\{X(t) \leq x\} \tag{1}$$

is the same for all t. Technically of course, from a practical point of view, our assumption of stationarity implies that we are examining $X(t)$ only after sufficient time has elapsed for initial value effects to become negligible.

The assumption of a Gaussian process means that $F(x)$ is given by the normal distribution whose density function is given by

$$f(x) = [1/(2\pi\sigma^2)^{\frac{1}{2}}] \exp[-(x - m)^2/2\sigma^2] \qquad -\infty < x < \infty \tag{2}$$

where

$$m = \int_{-\infty}^{\infty} xf(x)\,dx \tag{3}$$

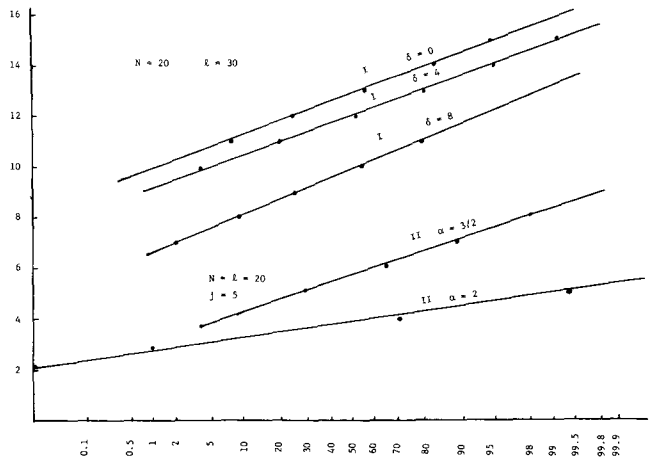
is the expected value (mean working-set size) and where

$$\sigma^2 = \int_{-\infty}^{\infty} (x - m)^2 f(x)\,dx \tag{4}$$

is the variance. Moreover, in general, for an arbitrary set of n time points $t_1, \ldots, t_n$ the joint distribution of $X(t_1), \ldots, X(t_n)$ is normal. (We return to a specification of this joint distribution later.) In view of the enormous literature dealing with the properties of Gaussian processes, this is indeed a key assumption. Fortunately, strong arguments can be advanced in support of this aspect of the model.

First, however, we note two immediate deviations inherent in the "normal approximation." The distribution is defined for negative as well as positive values of working-set size, and the distribution is continuous

Fig. 1. The normal approximation.



whereas working-set sizes take on only discrete values. The magnitude of the first approximation will be negligible because of the parameter values (m and σ) of interest. The extent of the second approximation is kept small by concentrating on those applications in which the working-set size varies over a sufficiently large number of values.

The normal approximation can be justified for the present purposes both on the grounds that it is consistent with alternate mathematical models of working sets and on the more intuitive grounds supplied by estimates of the page referencing behavior of programs. With respect to alternate mathematical models, for example, suppose the variation in working-set size can be modeled as a random walk (over the integers) with a centralized "elastic" force. That is, there exists a given position (corresponding to the mean working-set size) such that the probability of a "jump" toward that position is proportional to the distance the current position is away from it. In other words, suppose $x \geq 0$ denotes the magnitude of the difference between the current working-set size and the average value. Then the probability that x increases at the next step is inversely proportional to x; the process has a tendency to approach the mean that increases with the distance away from the mean. With this model we have the discrete analog of a certain Gaussian process known as the Ornstein-Uhlenbeck [5] process. In connection with random walks, it is also worth pointing out that any model according to which $X(t)$ can be regarded as the sum of independent random variables leads to the normal approximation by virtue of the central limit theorem.

In order to test the normal approximation more precisely, Monte Carlo experiments were conducted with the following probability model of page referencing [2]. Consider an arbitrary reference string $r_1, \ldots, r_i, \ldots$ where each reference denotes a page taken from a set of N pages. The LRU *stack* [6], $\underline{s}_i = [s_i(1), \ldots, s_i(N)]$, just after the ith reference r_i is defined as a list of the set of N pages ordered according to recency of usage; i.e. $s_i(k)$ is the kth most recently used page relative to r_i. (Pages not yet referenced in $r_1, \ldots, r_i$ are placed in an arbitrary order at the bottom of $\underline{s}_i$.) Random reference strings were generated by selecting successive page references from the LRU stacks according to a stationary distribution $\{b_k\}_{k=1}^{N}$ defined on stack positions. Thus, after r_i was selected from $\underline{s}_{i-1}$ according to $\{b_k\}$, $\underline{s}_i$ was then computed from $\underline{s}_{i-1}$ and r_i (by a very simple algorithm [6]) and the process repeated for r_{i+1}. Thus for all sequences and for all i $\Pr[r_i = s_{i-1}(k)] = b_k$, $1 \leq k \leq N$. Clearly, the property of locality implies that $b_1, b_2, \ldots, b_N$ is a nonincreasing sequence. Note that this model avoids the problem of modeling the precise pages which constitute the addressing locality as a function of time; it is the *extent* of locality and of its variation in time that is being modeled.

Using random page reference strings defined as above, the sequence of (Denning) working-set sizes was accumulated for various values of l. The distributions used in the experiments conducted were the uniform distribution, a family of arithmetic distributions, and a family of geometric distributions. As a general statement, the normal approximation was found to be excellent. The results from a number of typical runs are displayed in Figure 1 on (normal) probability paper (the normal distribution is associated with a set of sample points falling on a straight line).

In Figure 1, the lines marked I correspond to the unnormalized arithmetic distribution, $b_k' = 1 + (N - k)\delta$, with the parameter values shown. The lines marked II correspond to the unnormalized "geometric" distribution $b_k' = \alpha^{N-j-k+1}$ $(k \leq N - j)$, $b_k' = 1$ $(k > N - j)$. The parameters α and j are shown in the figure.

Quite recently a study [4] of working-set properties has also developed the normal approximation through an application of the central limit theorem to the page reference patterns to be expected from the majority of programs. Recent and preliminary empirical results [7] have also suggested the normal approximation, but primarily for larger "window" sizes, l. For smaller values of l, the probability mass function exhibits a shape more closely resembling the Poisson distribution.

The tractability of the model we have defined is due primarily to the fact that we have avoided the difficult combinatorial problems arising in the analysis of detailed models of paging algorithms [8, 9]. The potential

drawback, of course, is that the commonly used performance measure—expected paging rate—is not directly obtained. However in those cases where it is desired, approximate paging-rate behavior is obtainable by applying the empirical results of Belady and Keuhner [10] or Coffman and Varian [11]. In particular it has been shown that under general circumstances increases in paging rates vary with (at least) the square of decreases in the amount of storage allocated (below that which is required for the total program). Thus a transformation from space loss to paging rate is possible to this limited extent, and it has the advantage of being based on empirical data. For simplicity, however, we shall use only the space loss itself as a performance measure.

4. Storage Partitioning

The simplest way to control main storage when it is shared by several programs is to create a fixed partition and require each program to operate wholly within its block of the partition. In this way the (maximum) degree of multiprogramming is fixed. Intuitively, if the block size is sufficiently large to contain the working sets of programs a high percentage of the time, then the fixed partition scheme should work acceptably. To refine and quantify this idea we shall make use of the model of the previous section.

Suppose an M-page memory is partitioned into n blocks of b pages each. Let $X_i(t)$, $i = 1, \ldots, n$, denote the independent Gaussian processes for the working-set sizes of n independent programs executing in the n blocks. Let the (normal) distribution functions for the $X_i(t)$ have the common parameters m and σ. (Clearly, this last assumption is made for convenience rather than through necessity.) The joint behavior of the n independent processes specifies at each point in time the system storage configuration $\underline{Z}(t) = (Z_1(t), \ldots, Z_n(t))$, where $Z_i(t) = \min [b, X_i(t)]$, $1 \leq i \leq n$. The configuration $\underline{Z}(t)$ directly models a system in which programs execute concurrently, and with the appropriate change in time scale, it also represents processor sharing. Over longer time periods it models multiprogramming systems in which programs are given time-slices one at a time, in a round robin, for example. Finally, it is implicit in an analysis of $\underline{Z}(t)$ that we are studying a system in which a storage allocation procedure corresponding to the working-set definition is in effect. In other words, according to the definition of $\underline{Z}(t)$, we have working-set storage management in the sense that a maximum amount of working set for each executing program is maintained in its block. Once again for a specific case we can fall back on the Denning working set.

For given parameters, the effectiveness of the fixed partition system is very simply expressed in terms of the probability g that no blocks are in saturation, i.e. containing less than the total working set and thus causing higher paging rates. Specifically, since the n processes are independent, we have

$$g = \left[\int_{-\infty}^{b} f(x)\, dx\right]^n \tag{5}$$

where $f(x)$ is given in (2). Letting $\Phi(x)$ denote the cumulative distribution function for a normally distributed random variable with zero mean and unit variance, we have

$$g = \Phi^n[(b - m)/\sigma]. \tag{6}$$

A plot of g vs. m for various σ is shown in Figure 2, where we have assumed $M = 40$, $n = 4$, and therefore $b = 10$. The information contained in Figure 2 can be summarized by the statement that *the fixed partition scheme works well under the assumptions of small variations in working-set sizes (relative to the mean values) and a block size b that is roughly 2σ in excess of the mean working-set size, m.* These conditions obtain in the figure for the sharp breaking curve corresponding to $\sigma = 1$ if we assume $m = 8$. Under these circumstances storage utilization is high but there is no appreciable sacrifice in the paging rates brought about by saturated blocks. Note that the curves can be regarded as system performance curves indicating the proper choice of b. Assuming a small σ, then a b that is too small ($b < m + 2\sigma$) causes a precipitate collapse in performance due to the saturation of blocks.

A performance measure from which paging rates can be estimated more directly is the mean amount of space required by the n working sets in excess of that provided, i.e. the average amount of a working set not contained in a block, summed over the n blocks. In particular, we define

$$\bar{s} = E\{\textstyle\sum_{i=1}^{n} X_i'(t)\} \tag{7}$$

where

$$\begin{aligned} X_i'(t) &= 0, && \text{if} \quad X_i(t) \leq b, \\ &= X_i(t) - b, && \text{otherwise.} \end{aligned} \tag{8}$$

Thus

$$\bar{s} = n \int_b^{\infty} (x - b) f(x)\, dx.$$

Substituting (2) and carrying out the integration gives

$$\begin{aligned} \bar{s} = [n\sigma/(2\pi)^{\frac{1}{2}}] \exp [-(m - b)^2/2\sigma^2] \\ + n(m - b)\Phi[(m - b)/\sigma] \end{aligned} \tag{9}$$

The behavior of $\bar{s}$ will be displayed along with the corresponding quantity of a dynamic partitioning scheme to be described next.

Proceeding under the assumption that information is available on the contents of working sets, a more promising approach to the allocation of storage to a fixed number n of programs consists of allocating an amount of space to each program which is equal to the current working-set size. If the sum of the current working-set

Fig. 2. Fixed partition performance.

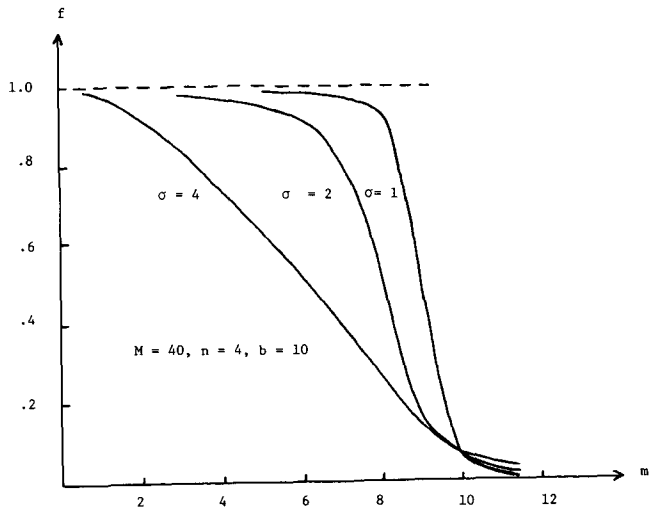


Fig. 3. Dynamic vs. fixed partitioning.

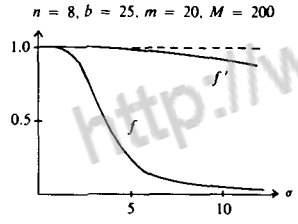
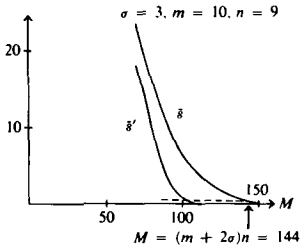


Fig. 4. Expected exceedances.



sizes exceeds M, some mechanism can be assumed for equitably contracting the n working sets so that a fit is obtained. (The details of such a scheme are not of concern here.) Intuitively, the above method of storage management should be an improvement over fixed partition schemes because temporarily large working sets can use space made available by temporarily small working sets. In order to measure this effect, we again use the model of locality presented in Section 3.

We define performance measures corresponding to g and $\bar{s}$ of (5) and (7). Let g' be the probability that the sum of n normally distributed random variables, each with parameters m and σ, exceeds M, and let $\bar{s}'$ be the mean amount by which this sum exceeds M. Now the sum of normally distributed random variables is again normally distributed;[2] summing n normal random variables with parameters m and σ gives a sum whose normal distribution has parameters nm and $n^{\frac{1}{2}}\sigma$. Thus

$$g' = \int_{-\infty}^{M} f_n(x)\,dx \tag{10}$$

and

$$\bar{s}' = \int_{M}^{\infty} (x - M) f_n(x)\,dx \tag{11}$$

where $f_n(x)$ is given by (2) with m and σ^2 replaced by nm and $n\sigma^2$, respectively. Working these results into a form more suitable for evaluation, we obtain

$$g' = \Phi[(M - nm)/n^{\frac{1}{2}}\sigma], \tag{12}$$

and

$$\bar{s}' = \sigma[(n/2\pi)^{\frac{1}{2}}] \exp\,[-(nm - M)^2/2n\sigma^2] + (nm - M)\Phi[(nm - M)/n^{\frac{1}{2}}\sigma]. \tag{13}$$

Using (12) and (13), a comparison of the fixed and dynamic partitioning schemes is illustrated in Figures 3 and 4. In Figure 3 one can observe the dramatic improvement represented by dynamic partitioning when the variation in working-set sizes is large relative to the mean. The value of σ such that $b = m + 2\sigma$ (i.e. $\sigma = 2.5$) is sufficiently small to give good performance for both systems. (It is interesting however that while the fixed system behaves well with $\sigma = 2.5$ it is still about 20,000 times more likely to have one or more jobs overflow than the dynamic system.) The performance of the fixed partitioning system drops off very rapidly as $m + 2\sigma$ becomes larger than b.

In terms of the expected "exceedances" $\bar{s}$ and $\bar{s}'$, Figure 4 shows the different memory sizes required by

[2] Note that, in view of the Central Limit Theorem, the sum will be approximately normal even if the individual working set sizes are not.

the fixed and dynamic methods for a given level of performance. For small M, $\bar{s}$ and $\bar{s}'$ become asymptotically the same because the entire memory in the dynamic case as well as all blocks in the fixed partition case becomes saturated. As stated earlier, the region of most interest for fixed partitioning occurs around $b = m + 2\sigma$. From the figure, the variation in working-set sizes is such that a total memory size is required for the dynamic method which is around 30 percent less than that required by the fixed method for a given performance level in this region. This indicates directly the substantial economies possible with dynamic partitioning when the variation in working-set sizes is relatively large.

Another important advantage of the dynamic scheme is its greater adaptability to variations in the working-set size distribution $F(x)$—when our "stationarity" assumption for $F(x)$ is not realistic. Performance can degrade substantially with increases in the parameters m and σ when fixed partitioning is used, but the dynamic scheme described above does not suffer to nearly the same extent.

5. Final Remarks

We have proposed a stationary Gaussian process as a model of the behavior of program execution due to locality. The model was originally motivated by the working-set definition, and experimental results were provided which showed that the Gaussian assumption was reasonable.

The model has been applied in a straightforward fashion to the comparative analysis of fixed and dynamic storage partitioning. The general conclusion was that the dynamic method is superior except under the restrictive assumption that the relative variation in working-set sizes is small and the block size of the fixed partition is set at the 2σ point of the distribution. Also, it should be emphasized again that the dynamic scheme has the advantage that it adapts to changes in the parameters m and σ while fixed partitioning generally does not.

An extension of dynamic storage partitioning is obtained by allowing the number of jobs in the system to change dynamically. In a practical realization of this method one must be concerned about instability effects at the boundary M. For example, if we introduce new programs or reactivate old ones at exactly the point when sufficient space is available, the likelihood of overflowing in a short time interval can be rather high. To avoid this "bunching" of overflow and the consequent overhead, programs can be introduced or reintroduced into execution only when the space available exceeds h, where h is a parameter of the method which is chosen greater than the expected size of the job to be introduced.

In addition to the performance measures for the other methods, we are interested here in the expected number of jobs in the system (which one would expect to be greater than the number that can be handled by the methods discussed in the earlier sections) and in the overhead caused by introducing and eliminating jobs. We conjecture that this method would be very robust against changes in the parameters m and σ.

A few results can be obtained with the assumptions we have already made. For example, an upper bound for the expected number of jobs in the system is $E(n_{\max})$ where $n_{\max}$ is the largest n such that $X_1(t) + \cdots + X_n(t) \leq M$. As we would guess, $E(n_{\max})$ is approximately $M/m - 1/2$. A lower bound for the expected number of jobs can be found in a similar manner.

Another result which follows easily from our assumptions is that, if $h = 2m$ and if the job introduced has expected size m at introduction, clearly, at most one-half of the paging can be caused by the introduction of new jobs. To obtain more precise information concerning the expected number of jobs in the system, the number of jobs introduced per unit time, and other measures of performance, we must make assumptions concerning the joint behavior of $X(t)$ at different values of t. This is done by specifying the covariance, cov $(X(t), X(s))$, which by stationarity is a function of $t - s$ only. An assumption has been made that cov $(X(t), X(s)) = \sigma^2 \exp(-a \mid t - s \mid)$, which corresponds to the Ornstein-Uhlenbeck process mentioned earlier, and some progress is being made with this model.

References

1. Denning, P.J. Virtual memory. *Computing Surveys, 2,* 3 (Sept. 1970), 154–189.
2. Oden, P.H., and Shedler, G.S. A model of memory contention in a paging machine. IBM Res. Tech. Rep. RC 3053, IBM, Yorktown Heights, N.Y., Sept. 1970.
3. Gaver, D.P. Jr., and Lewis, P.A.W. Probability models for buffer storage allocation problems, *J ACM 18,* 2 (Apr. 1971), 186–198.
4. Denning, P.J., and Schwartz, S.C. Properties of the working set model. *Comm. ACM 15,* 3 (Mar. 1972), 191–198.
5. Feller, W. *Introduction to Probability Theory and Its Applications, Vol. II,* Wiley. New York, 1966.
6. Mattson, R.L., Gecsei, J., Slutz, D.R., and Traiger, I.W. Evaluation techniques for storage hierarchies. *IBM Syst. J. 9,* 2 (1970) 78–117.
7. Yue, P.C. IBM Res. Div., Yorktown Heights, N.Y. Private communication.
8. King, W.F. III. Analysis of paging algorithms. Proc. IFIPS Conf., Ljubljana, Yugoslavia, 1971, pp. 155–159.
9. Aho, A.V., Denning P.J., and Ullman, J.D. Principles of optimal page replacement. *J. ACM 18,* 1 (Jan. 1971), 80–93.
10. Belady, L.A., and Kuehner, C.J. Dynamic space-sharing in computer systems. *Comm. ACM 12,* 5 (May 1969), 282–288.
11. Coffman, E.G., and Varian, L.C. Further experimental data on the behavior of programs in a paging environment. *Comm. ACM 11,* 7 (July 1968), 471–474.
12. Cramer, H., and Leadbetter, M.R. *Stationary and Related Stochastic Processes.* Wiley, New York, 1967.

阅读此文的还阅读了：

26. Study on the Mathematical Model of Hydraulic Jump Atomization

27. Study of Bioreactor Landfill Cell Design Using Base Model

28. The mathematical model of intelligent transportation system

29. Study on Mathematical Stress-strain Model of Frozenthawed Soil Based on Structure Mechanism

30. Improving the Performance of Content-Locality based Storage Management

31. English Study, Memory and Mathematical Thinking

32. Metal Spray Forming Process Examined Using Mathematical Model

33. Potential Assessment of Bandon Bay,Gulf of Thailand by Using Mathematical Model for Sustainable Coastal Resources Management

34. The Study on the Simulation Mathematical Model of Toroidal Worm-Gearing

35. SAR Target Configuration Recognition Using Locality Preserving Projections

36. VQ-Based Image Watermarking Using Codebook Partitioning

37. STUDY ON MATHEMATICAL MODEL OF JIG PROCESS

38. Study on the Mathematical Model and the Controlling Rule for Variable Rate Fertilization

39. CHAOTIC BEHAVIOUR IN A MATHEMATICAL CANCER MODEL

40. LONG SENTENCE PARTITIONING USING TOP-DOWN ANALYSIS FOR MACHINE TRANSLATION

41. Using Term-based Partitioning Frameworkon MongoDB and Elastic Search

42. Machine Performance Degradation Recognition Using Locality Preserving Projections and Clustering Approach

43. An Optimized Lifetime Model using Energy Holes Reduction near Sink's Locality of WSN's

44. Study on Model Construction of Gray Correlation Method Based on Probability Theory and Mathematical Statistics

45. Using Parallel Partitioning Strategy to Create Diversity for Ensemble Learning

46. Hierarchical Storage design using queueing model in VOD system

47. Frequency and Similarity-Aware Partitioning for Cloud Storage Based on Space-Time Utility Maximization Model

48. Mathematical Model of Cycle Economy

49. PRELIMINARY STUDY OF PLATELET AGGREGATION MECHANISM USING MATHEMATICAL MODEL

50. STUDY ON THE INFLUENCE OF DIFFERENT OPERATIONAL MODES OF THE SANMENXIA RESERVOIR ON THE TONGGUA