Real-Time Interactive MPEG-4 System Encoder Using a Cluster of Workstations

Yong He, *Student Member, IEEE,* Ishfaq Ahmad, *Member, IEEE,* and Ming L. Liou, *Fellow, IEEE*

***Abstract*— MPEG-4 currently being finalized by the Moving Pictures Experts Group of the ISO is a multimedia standard. MPEG-4 aims to support content-based coding of audio, text, image, and video (synthetic and natural) data, multiplexing of coded data, as well as composition and representation of audio-visual scenes. One of the most critical components of an MPEG-4 environment is the system encoder. An MPEG-4 scene may contain several audio and video objects, images, and text, each of which must be encoded individually and then multiplexed to form the system bitstream. Due to its flexible features, object-based nature, and provision for user interaction, MPEG-4 encoder is highly suitable for a software-based implementation. A full-scale software-based MPEG-4 system encoder with real-time encoding speed is a nontrivial task and requires massive computation. We have built such an encoder using a cluster of workstations collectively working as a virtual parallel machine. Parallel processing of MPEG-4 encoder needs to be carried out carefully as objects may appear or disappear dynamically in a scene. In addition, objects may be synchronized with each other. User interactions may also prohibit a straightforward parallelization. We propose a modeling methodology that captures the spatio-temporal relationship between various objects and user interaction. We then propose a number of scheduling algorithms that periodically allocate MPEG-4 objects to multiple workstations ensuring load balancing and synchronization requirements among multiple objects. Each scheduling algorithm has its own performance and complexity characteristics. The experimental results, while showing real-time encoding rates, exhibit tradeoffs between load balancing, scheduling overhead cost, and global performance.**

***Index Terms*— MPEG-4, parallel processing, petri nets, scheduling algorithm, video compression.**

I. Introduction

AUDIO-VISUAL standards are required to enable the interfacing of various technologies. MPEG-4, currently being developed by the Moving Pictures Experts Group (MPEG) of ISO and to be finalized by early 1999, will become the standard of multimedia [1]. The aim of MPEG-4 is to define a standard syntax and interfacing for technologies in computer, telecommunication, and television/film industries. MPEG-4 will support content-based communication, access, and manipulation of digital audio-visual objects. In addition, MPEG-4 will provide improved compression efficiency and support new functionalities not available in existing standards. It is specially designed to work in a variety of network platforms such as the Internet and wireless networks. Compared to the conventional frame-based compression techniques, the object-based coding and representation of the multimedia information enable MPEG-4 to cover a broad range of emerging applications [2].

One of the most critical components of an MPEG-4 environment is the system encoder, comprising audio, image, text, and video encoders as well as encoders for scene description. Objects must be individually encoded and then multiplexed to form the system bitstream. Due to its flexible features, object-based nature, and provision for user interaction, MPEG-4 system encoder is highly suitable for a software-based implementation. However, a full-scale software-based MPEG-4 system encoder with real-time encoding speed is a nontrivial task and requires massive computation. One problem is that in an MPEG-4 scene, objects may appear or disappear randomly and may be synchronized with each other. Furthermore, user may interact with the scene, making it difficult to have a static parallelization.

In this paper, we propose a scheme for a full-scale MPEG-4 system encoder using a cluster of workstations collectively working as a virtual parallel machine. First, we propose a modeling methodology that captures the spatio-temporal relationship between various objects and user interaction. We then propose a number of scheduling algorithms that allocate MPEG-4 objects to multiple workstations ensuring load balancing and synchronization requirements among multiple objects. Through proper scheduling of the workstation cluster, the encoder is able to encode multiple objects in parallel to achieve real-time speed. The performance of the encoder scales according to the number of workstations used. The experiment results indicate that a real-time encoding rate can be achieved for the sequences with multiple media objects.

The rest of this paper is arranged in the following manner. Section II gives a brief overview of the MPEG-4 standard. Section III describes the proposed parallelism of the MPEG-4 encoder in detail, including the modeling methodology and dynamic scheduling algorithms for the video encoder exploited. Section IV provides the experimental results, and the last section presents some concluding remarks and future research directions.

II. MPEG-4 Standard Overview

In the past decade, a number of ISO and ITU standards, such as MPEG-1, MPEG-2, and H.261, H.263, have been

Manuscript received September 3, 1998; revised February 24, 1999. This work was supported by the Hong Kong Telecom Institute of Information Technology and the Research Grants Council of the Hong Kong Special Administrative Region. The associate editor coordinating the review of this paper and approving it for publication was Prof. Jan-Ming Ho.

Y. He and M. L. Liou are with the Department of Electrical and Electronic Engineering, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: eehey@ee.ust.hk; eeliou@ee.ust.hk).

I. Ahmad is with the Department of Computer Science, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: iahmad@cs.ust.hk).

Publisher Item Identifier S 1520-9210(99)04099-7.

TABLE I
VIDEO CODING STANDARDS

Standard	Format	Compressed Bit Rates	Applications
H.261	QCIF/CIF	p × 64 kbps, p = 1, 2, …, 30	Videophone and video conferencing via ISDN, etc.
H.263	SQCIF~16CIF	Flexible	From low bitrate videophone to high quality video conferencing, etc.
MPEG-1	SIF	1.5 Mbps	Storage/retrieval of VCR quality video on CD-ROM, VCR quality VOD via ADSL, etc.
MPEG-2	Flexible	>2Mbps	Broadcasting satellite service, cable TV distribution, digital terrestrial television broadcast, electronic cinema, electronic news gathering, fixed satellite service, home television theatre, multimedia mailing, video surveillance, etc.
MPEG-4	Flexible	Flexible	Interactive multimedia, surveillance, mobile multimedia, content-based storage and retrieval, streaming video on the internet/intranet, video games, collaborative scene visualization, broadcast, studio and television post-production, DVD, etc.

proposed and adopted by industry for representing and coding of digital audio-visual data. These standards reduce the size of audio-visual data through compression and facilitate creation, transmission, and presentation of multimedia information by providing compatibility among various technologies [3].

Table I presents the major features and applications specified in the various standards. H.261, developed by ITU, is targeted for audiovisual service at $p \times 64$ kbps ($p = 1, \cdots, 30$) bit rate, which covers the entire ISDN channel capacity [4]. The successor to H.261 is H.263, which is supposed to compress the moving picture component of audio-visual services and applications at a very low bitrate [5] but provides a better picture quality compared with H.261. MPEG-1, the first standard developed by the MPEG of ISO/IEC, specifies the coding of audio-visual data at a bitrate of about 1.5 Mbps [6]. The second standard (MPEG-2) is a generic coding standard for low to high-resolution moving pictures and associated audio data with the bitrate ranging from 2 to 30 Mbps [7].

The new generation of multimedia applications requires new functionalities that are not be supported by the previous standards. A new standard (MPEG-4) is going to support universal coding of all types of audiovisual objects and enable advanced functionalities, such as content-based interactivity, high compression, and random access. Hence, a broader range of present and future multimedia applications can be covered by MPEG-4.

The range of applications using MPEG-4 is simply enormous [8]. The applications can be real-time or nonreal-time, interactive or noninteractive. Typical examples include videotelephony, video conference, cooperative work, remote classroom, remote monitoring, news gathering, tele-shopping, CAD tools, tourism, encyclopedias, multimedia messaging, entertainment, animation, scientific visualization, games, etc. Multimedia authoring and video editing capabilities are particularly attractive features of MPEG-4 as they will facilitate the development of multimedia processing and editing tools—as opposed to existing word-processors and typesetters—in the next century. In addition, previous standards such as H.263 and MPEG-2 are in a sense embedded in MPEG-4, ensuring that existing applications such as digital TV and videophone are also supported by MPEG-4.

MPEG-4 treats a scene to be coded as multiple individual media objects, such as audio, video, graphics, and animation, with a different spatio-temporal characteristic. Each media object may be manipulated dynamically due to user interaction, which results in time varying computation cost. In addition, each tool and algorithm adopted to encode a certain type of media object may have different computational complexity and data dependence. Orchestrating numerous tasks of the encoder and scheduling multiple processors pose a number of research challenges to achieve globally optimal performance using parallel processing.

A part of MPEG-4 standard is the object-based hybrid natural and synthetic video, which uses technologies enabling the functionalities such as content-based interactivity, efficient compression, error resilience, and object scalability [9]. A video object is defined as the entities in the bitstream that a user can access and manipulate. Usually, the shape of the video object is arbitrary and may vary with time. Fig. 1 illustrates an MPEG-4 video coding and composition scenario. The video objects, such as *Man*, *Car*, and the background *Map*, are separated from the input video signal by the segmentation information. Each video object is encoded separately by a video object (VO) encoder. After transmitting through the network, the bitstreams are decoded separately and the video objects are composed for presentation by the compositor. According to user interactions, the compositor can reconstruct the original scene [scene (a)] or manipulate the objects and create a different scene [scene (b)]. Furthermore, by downloading the new video object from a local or remote database library, the compositor can replace and/or insert new object during the

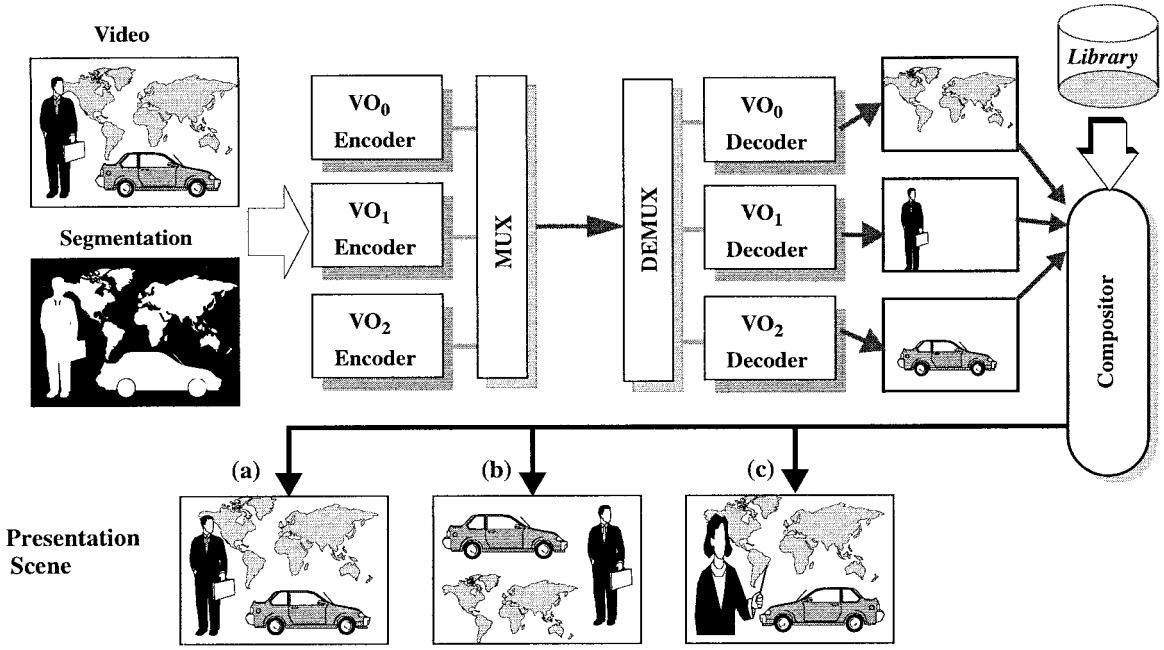


Fig. 1. MPEG-4 video scene coding and composition.

presentation. For example, one can replace the object *Man* by the new object *Woman* from the library [scene (c)].

The major components of the MPEG-4 video object encoder are shape coder, motion estimation and composition, and texture coder. Most of these procedures are similar to H.263 and MPEG-1/2 but are modified to handle arbitrary object shapes. The shape coder is used to compress the segmentation information that indicates the object region and contour within the scene. Motion estimation and compensation (ME/MC) are used to reduce temporal redundancies, and the techniques such as unrestricted ME/MC, advanced prediction mode, and bidirectional ME/MC are supported to obtain a significant quality improvement. The texture coder that deals with the intra and residual data after motion compensation includes algorithms that are also similar or identical to the ones used in H.263, including DCT or shape adaptive DCT (SA-DCT), MPEG or H.263 quantization, intra DC and AC prediction, and VLC. Most algorithms and tools are only performed on the pixels inside the object, and such data-dependent operations may result in the variable computation cost of the video object. Further details on the MPEG-4 video standard can be found in [10].

III. Implementing the MPEG-4 System Encoder

A media object defined in MPEG-4 can be classified as either live data or preorchestrated data, depending on its generation time. Live data is generated in real-time and its characteristics can only be obtained along the sequence, while preorchestrated data refers to stored data whose playout script has already been specified beforehand. Generally, MPEG-4 based session is a composite of preorchestrated and live media data.

Fig. 2 is a playout example of an MPEG-4 multimedia session in time chart. The presentation scenario begins with concurrent playout of VO_0 and VO_1, followed by a new video object VO_3 at the time t_1. Here, VO_2 and AO_0 can be treated as the background audio-visual object throughout the whole session. The two-dimensional (2-D) graphic object is involved from time t_1 to t_2. Face and body animation (FBA) is a synthetic object controlled by a set of model parameters. Binary format for scene (BIFS) provides a complete framework for describing scene composition. The distinct characteristic of each object, especially for video objects, such as the shape size, playout duration, and spatial allocation, can be different from each other. Furthermore, the temporal relationship among the objects may be designed synthetically. For example, we can specify that VO_0 and VO_1 are synchronized during the presentation from t_0 to t_1, and VO_2 and AO_0 are synchronized from t_0 to t_2; the playout frame rate of VO_0 and VO_1 is 30 frames/s, and the frame rate of VO_2 and VO_3 is 10 frames/s. According to the states of the media objects, the entire session can be partitioned into a number of presentation intervals, as illustrated in Fig. 2.

The maintenance of temporal relationships among multiple data is the most crucial requirement in a distributed multimedia system [11]. In general, a distributed MPEG-4 system consists of an encoder server including various media encoders, a network, and multiple presentation clients. To enable a proper playout at the client site, the entire system must be able to operate in a synchronized fashion. In real-time applications, the video encoder is the most time consuming component. It is impossible to achieve such real-time performance with a single PC or workstation, and thus, parallel processing is required to speedup the computation of video encoder. However, to ensure efficiency, we have to determine the processing schedule of the video object tasks and manage the system resources according to the playout requirements of the tasks.

Fig. 3 illustrates the implementation scenario of the proposed MPEG-4 system encoder. The model generator cre-

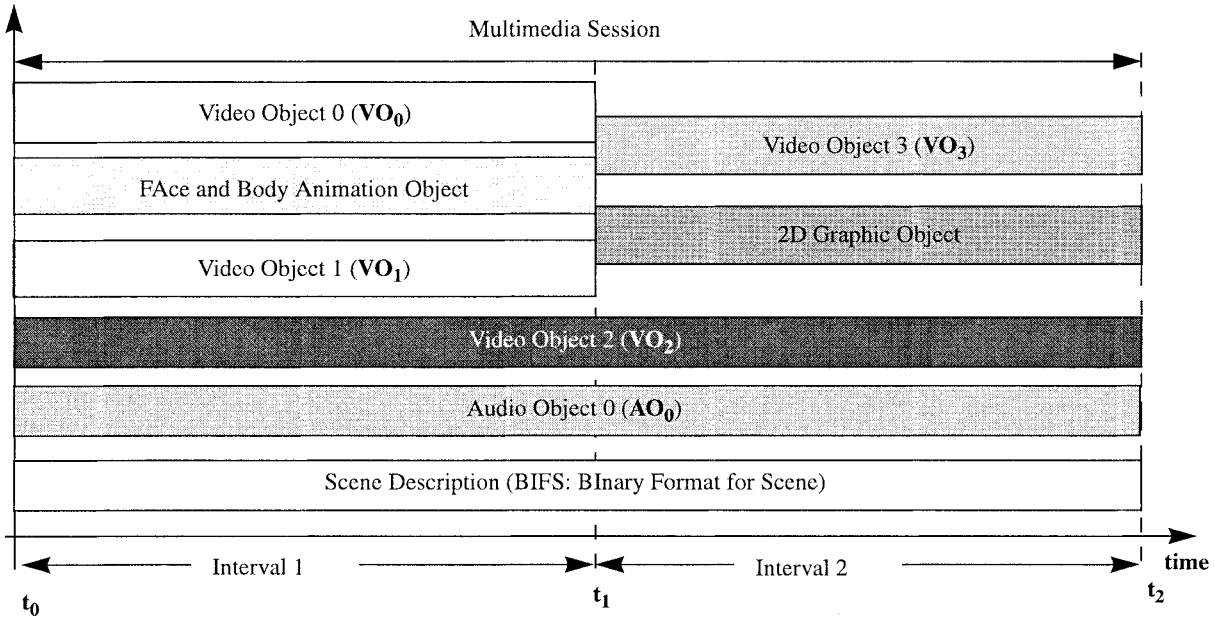


Fig. 2. Playout scenario of an MPEG-4 multimedia session.

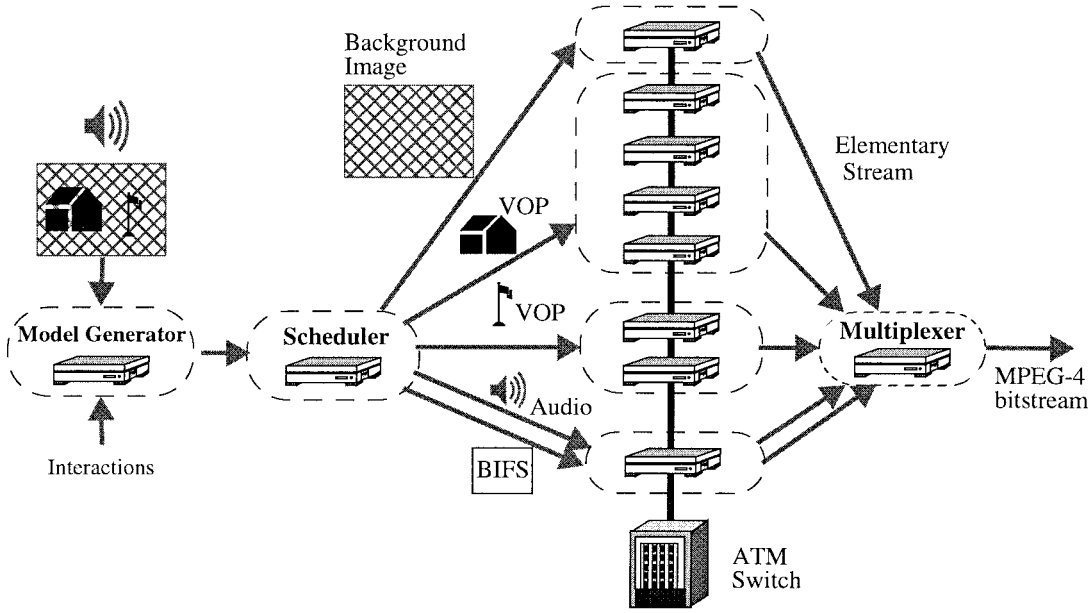


Fig. 3. Proposed MPEG-4 system encoder implementation.

ates the model by analyzing the user interactions and the properties of input media objects such as audio, video, background image, and scene description. Based on the information generated by the model, the scheduler allocates the objects to workstations. In general, video is most computationally intensive and requires more machines. Moreover, different video objects may require different computational power. In our encoder, we run the audio, text, and BIFS encoders in one machine. The elementary bitstreams of individual coded objects are multiplexed and transmitted. The user interactions with the objects may require rescheduling of the encoding tasks.

A. Modeling MPEG-4 Based Interactive Video

Various modeling techniques for multimedia presentation have been proposed such as graphic models [12], petri net-based models [13], object-oriented models [14], language-based models [15], and temporal abstraction models [16]. Here, we employ such a model known as the object composition petri net (OCPN) to represent the occurrence of multiple video objects due to its ability to explicitly capture all necessary temporal relations [17]. In general, an OCPN refers to a petri net graph, and each place represents the playout of an object, while each transition represents a synchronization point. However, in order to model the flow of MPEG-4-

based media objects, more definitions are essential to allow the dynamic behaviors such as user interactions and object attributes variation. Therefore, we present an augmented model called MPEG-4 OCPN (MOCPN), which can describe the temporal properties of the video objects,and user interactive operations.

The MOCPN is 7-tuple $MOCPN = \{P, T, A, M, D, TS, PT\}$ where

1) $P = \{p_0, p_1, \cdots, p_m\}$ is a finite set of places. $m \geq 0$.
2) $T = \{t_0, t_1, \cdots, t_n\}$ is a finite set of transitions. $n \geq 0$ and $P \cap T = \Phi$.
3) $A = \{P \times T\} \cup \{T \times P\}$ is a mapping representing arcs between places and transitions.
4) M: $P \rightarrow I$ represents the tokens distributed in the place where I is the integer.
5) D: $P \rightarrow R^+$ represents the augmented duration D of the place where R is a non-negative real number.
6) TS: $T \rightarrow \{IntraS, InterS\}$ different transition types. $IntraS$ transition represents the intraobject synchronization and $InterS$ represents the interobject synchronization.
7) PT: $P \rightarrow \{PR, PC, PS\}$ represents different place types, PR is the request of the clients, PC is the model construction operation, and PS defines the scheduling operation in the encoder.

The above P, T, and A definitions are the same as that of the OCPN. The number of tokens to be deposited at a given place P is indicated by M. The firing condition are determined by the tokens at the input place.

Generally, an MPEG-4 based video session is very complex and enforces different characteristic at different syntax levels, such as video session (VS), VO, and VO plane (VOP), as specified by the standard. It is important that such hierarchical levels of synchronization be captured by the model to enable the handling of large and complex MPEG-4 scenarios. For MOCPN, such hierarchical modeling capabilities can be achieved through subnet replacement. At the highest level of abstraction, MOCPN can be represented by a set of abstract place, and each abstract place can be decomposed into an MOCPN subnet. The abstract places in the sub-MOCPN indicate finer-grained data units and timing constraints. Similarly, the sub-MOCPN places can, in turn, be an abstraction of the lower MOCPN subnet. Such a replacement process can be recursively applied up to the lowest level where each place cannot be decomposed any more. In our approach, we define three levels of MOCPN.

1) VS_MOCPN;
2) VO_MOCPN;
3) VOP_MOCPN.

The abstract place of MOCPN at different level is defined as

1) PVS: $P \rightarrow VS$ represents the entire video session;
2) PVO: $P \rightarrow \{VO_0, VO_1, \cdots, VO_{N-1}\}$ represents a set of video objects;
3) $PVOP$: $P \rightarrow \{VOP_0, VOP_1, \cdots, VOP_{M-1}\}$ represents a set of video object planes.

VS_MOCPN is the highest level with an abstract place PVS that indicates the entire video session. VO_MOCPN is the middle level with a set of places PVO that represent the video objects, and the transition known as *InterS* describes the temporal precedence and synchronization between the video objects. The lowest level VOP_MOCPN is composed of atomic places $PVOP$, which represent each frame of the video object. The intraobject synchronization $IntraS$ and interobject synchronization $InterS$ can be captured by the VOP_MOCPN. Fig. 4(a) shows a decomposition process of MOCPN, and Fig. 4(b) shows the global VOP_MOCPN representation of the video session example in Fig. 2. The time instance t_0, t_1, and t_2 are set to be 0, 3, and 6 s, respectively.

For most preorchestrated video sequences, as the timing constraints are known *a priori*, it is possible to determine the structure of a MOCPN beforehand. For live video data, however, since related knowledge for constructing the MOCPN cannot be verified *a priori*, the model has to be generated during the video playout session. In addition, the preorchestrated or live video data can be ceased and new video data can be added at any time by the user interactions. Furthermore, the temporal relationships between the video objects may also be manipulated by the user. Due to unpredictable user interactions, the MOCPN model should be able to support both deterministic as well as imprecise events on the fly.

Fig. 5 illustrates the model generation strategy of the MOCPN [using Fig. 4(b) as an example] that allows user interactions. Corresponding to the user request, the MOCPN model can be generated at run time. At the beginning of time t_0, we can obtain the attributes such as playout deadlines and data dependency of VO_0, VO_1, and VO_2 initially and construct the MOCPN as place PC_0 indicated. Such a model may remain the same if all VO's status are stable. Caused by a user interaction PR_0 at time t_1, VO_0 and VO_1 are halted, and a new VO_3 is succeeded and synchronized with VO_2. Thus, the model should be changed by PC_1, which is in charge of updating the attributes of the video objects and remodeling the MOCPN accordingly. The same approach PC_2 is taken at time t_2, and the video session is stopped by the request PR_1. With such dynamic construction process, external interrupts can be quickly responded to.

The attributes of each video object and video object plane are illustrated as follows:

VO_i	ith VO, where $i = 0, 1, \cdots, N-1$;
$VOP_{i,j}$	jth frame of video object VO_i, where $j = 0, 1, \cdots, m_i - 1$;
δ_i	start time of the video object VO_i;
τ_i	synchronization interval of the video object VO_i;
$S_{i,j}$	size of $VOP_{i,j}$;
$d_{i,j}$	playout deadline of $VOP_{i,j}$;

Since MOCPN can represent synchronization of multiple objects, the playout deadline for each VOP can be deduced through analyzing its precedence relations and playout duration

$$d_{i,j} = \delta_i + j \cdot \tau_i. \tag{1}$$

B. Dynamic Scheduling of Multiple Objects

Using the information generated by the model, the video objects need to be scheduled to multiple workstations for

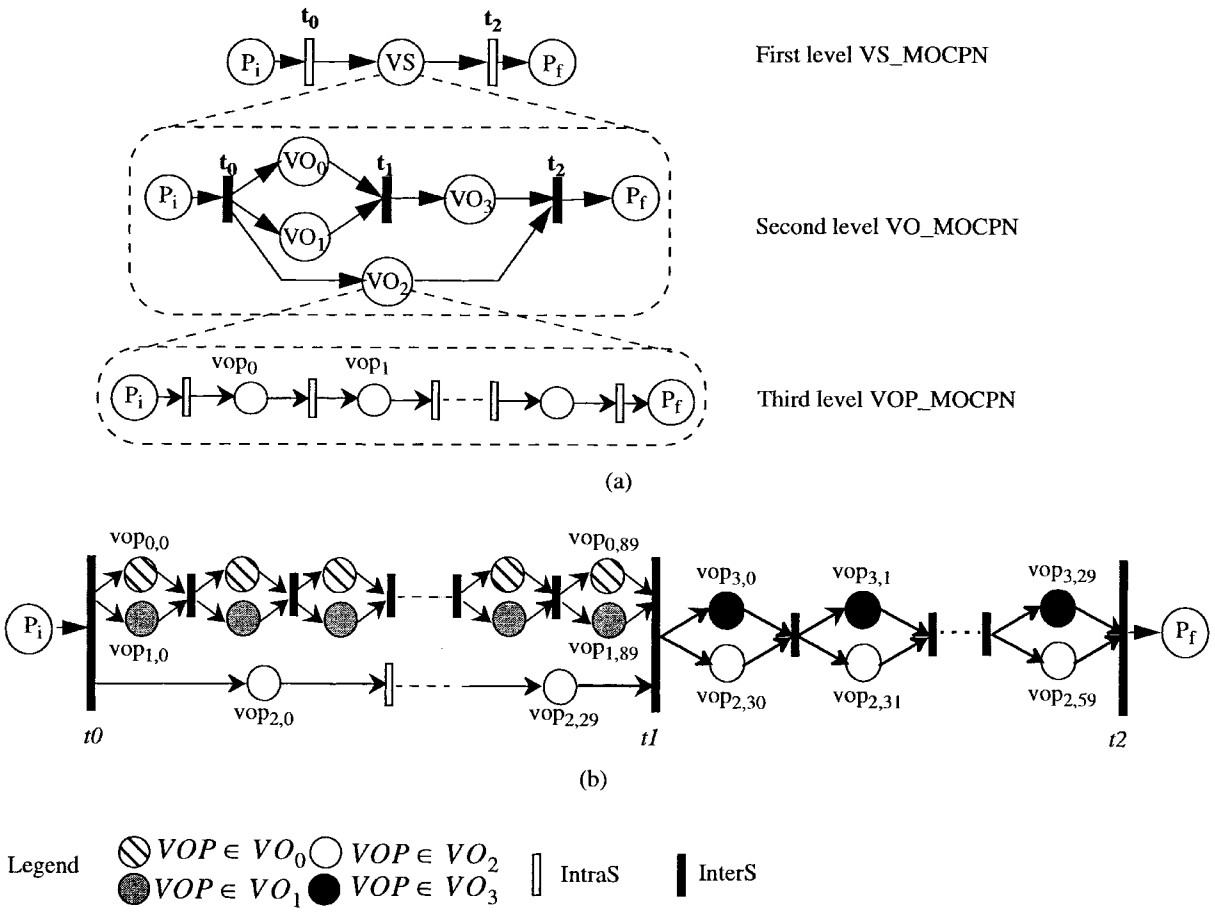


Fig. 4. Example of hierarchical MOCPN.

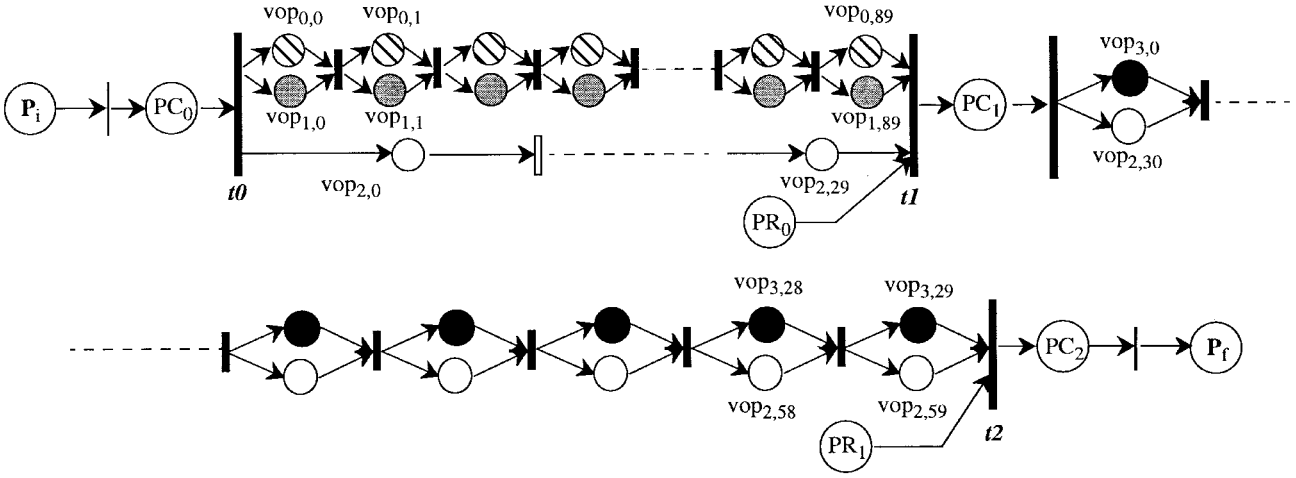


Fig. 5. Construction of MOCPN model.

concurrent encoding. The objective of scheduling in a parallel environment is to minimize the overall execution time of a concurrent program by properly allocating its tasks (in this case, video objects) to the processors [18]. A scheduling algorithm can be classified as static or dynamic. Due to the unpredictable behavior of each video object, the scheduling scheme in an MPEG-4 system should adapt to the playout requirements captured by the MOCPN model and simultaneously allocate the resources in an optimal way. For the encoder MOCPN, which indicates the processing operations, the augmented non-negative time D is designed to stand for the processing time cost, and the distribution of tokens marked by M for a place is specified to represent the workstations assignment by the scheduling algorithm. Suppose

the following.

- T_{vs} $D \rightarrow PVS$ is the total execution time cost for entire video session.
- $T_{vo}(i)$ $D \rightarrow PVO$ is the execution time cost for VO_i.
- $T_{vop}(i, j)$ $D \rightarrow PVOP$ is the processing time for each $VOP_{i,j}$.
- T_s $D \rightarrow PS$ is the scheduling time for task assignment.

Generally, $T_{vop}(i, j)$ is composed of the encoding time T_{enc} and the interprocessors communication time T_{com}. For parallel processing, a data partitioning method is applied such that each workstation may handle certain data blocks, which introduces an additional partitioning time T_p. In a parallel environment, the elapsed time of a parallel program is measured by the longest finish time of a processor. If K is the number of processors allocated to the $VOP_{i,j}$, $T^k_{enc}(i, j)$ is the encoding time of the kth processor for encoding $VOP_{i,j}$, and we suppose both T_{com} and T_p are the same for all the processors. Then, the execution time is given by

$$T_{vop}(i, j) = \max_k \{T^k_{enc}(i, j) + T_{com} + T_p\} \tag{2}$$

where

$$k = 1, 2, \cdots, K.$$

Obviously, $T^k_{enc}(i, j)$ is directly proportional to the encoding data size. If the data partitioning method can guarantee load balancing between the processors, each processor may be assigned nearly the same amount of data, which means the entire frame size $S_{i,j}$, is partitioned into K blocks equally.

Therefore

$$T^k_{enc}(i, j) \approx \alpha \cdot \left\lceil \frac{S_{i,j}}{K} \right\rceil \tag{3}$$

where α is the ratio coefficient.

For each video object, the execution time cost can be calculated as

$$\begin{aligned} T_{vo}(i) &= T_s + \sum_{j=0}^{m_i-1} T_{vop}(i, j) \\ &= T_s + \sum_{j=0}^{m_i-1} (\max_k \{T^k_{enc}(i, j) + T_{com} + T_p\}) \end{aligned} \tag{4}$$

where m_i is the total number of frames for VO_i.

The completion time of the entire session T_{vs} depends on the scheduling decision for encoding multiple video objects.

When the video sequence contains a single video object or multiple independent video objects, only the intraobject synchronization should be enforced at the encoder. To preserve the intraobject synchronization, the following equation should be satisfied at the encoder:

$$\sum_j (\max_k \{T^k_{enc}(i, j) + T_{com} + T_p\}) \le d_{i,j} - \delta_i \tag{5}$$

which means the successive encoded bitstream should be supplied to the presentation clients before the playout deadline.

If the video sequence contains various interdependent objects, both intra and interobject synchronization should be guaranteed at the encoder. To meet the interobject synchronization between the video objects such as VO_0 and VO_1, let us assume both objects start at the same δ and have the same playout duration τ, namely, $\delta_0 = \delta_1 = \delta$ and $\tau_0 = \tau_1 = \tau$; the following equation should be satisfied:

$$|T_{vop}(0, j) - T_{vop}(1, j)| \le \varepsilon \tag{6}$$

where ε is the maximum allowable differential delay in presentation of two video objects with the same temporal reference specified by the standard. Equation (6) dictates that both VOP's should complete the processing at nearly the same time before their playout deadline.

Various scheduling algorithms are reported in literature [19]. In the context of MPEG-4, the states of the video object may change at any time. Therefore, a dynamic scheduling scheme with low scheduling cost T_s is required. The tradeoff between the overall performance and scheduling cost should be considered. Multiple objects scheduling problems are known to be NP-hard problems, and therefore, heuristic methods are widely selected as the feasible solutions. The earliest deadline first (EDF) algorithm is widely used in the parallel and distributed environment computing [20]. The principle of EDF is to assign the tasks with earlier deadlines higher priorities. The execution order is determined by the priority of each task. It is a greedy approach since the tasks associated with earlier deadlines must be selected first.

We propose three scheduling algorithms that have different scheduling costs and performance levels. The first is a low-cost approach called the *round-robin scheduling* (RRS) algorithm, which schedules the video objects to workstations in a sequential EDF order and in a round-robin fashion. It adapts to the size variation of the video objects, which results in a minimum scheduling overhead cost. Hence, RRS algorithm is suitable for the applications with heavy client interactions due to its quick responding and scheduling time. The second algorithm, called *group scheduling* (GS) algorithm, divides the workstations into a number of groups such that each group performs the encoding on a single video object concurrently using an EDF order. With enough workstations, the GS algorithm can achieve a higher speedup and ensure the load balance within each group. In order to deal with VO's whose sizes change greatly with varying computation power requirements, the third scheduling algorithm, which is called *GOV-adjusting scheduling* (GAS), is proposed that, under a certain condition, merges tasks to achieve load balancing among the groups. Moreover, rescheduling is performed periodically on the basis of *group of VOP* (GOV) for further adjusting the processing configuration and related parameters.

An additional notation used in the algorithms is presented as follows:

- V scheduling sequence of the video session;
- P_k kth workstation of the system, where $k = 1, 2, \cdots, K$, and K is the total number of workstations;
- R_k scheduler on P_k;
- L_x scheduling intervals triggered by the user interactions;
- G_i $M \rightarrow g_i$, $g \in I$, which is a mapping from the set of tokens to a set of groups G_i containing g_i work-

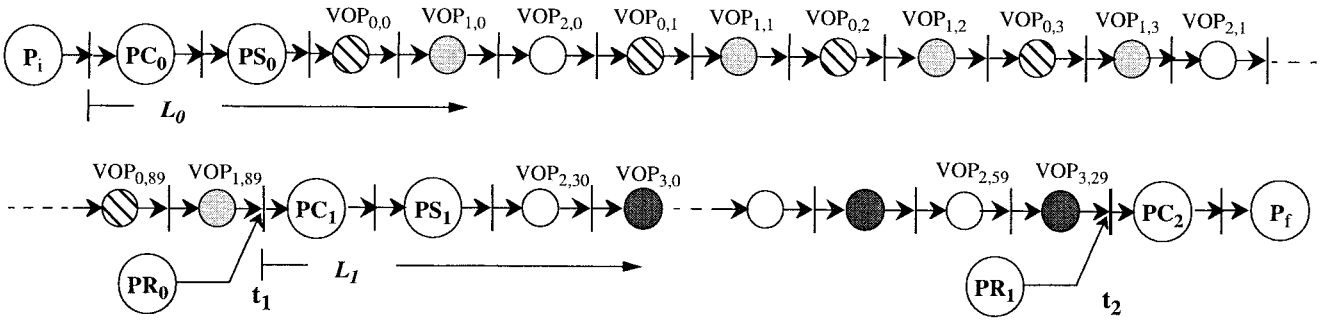


Fig. 6. MOCPN model for RRS scheduling.

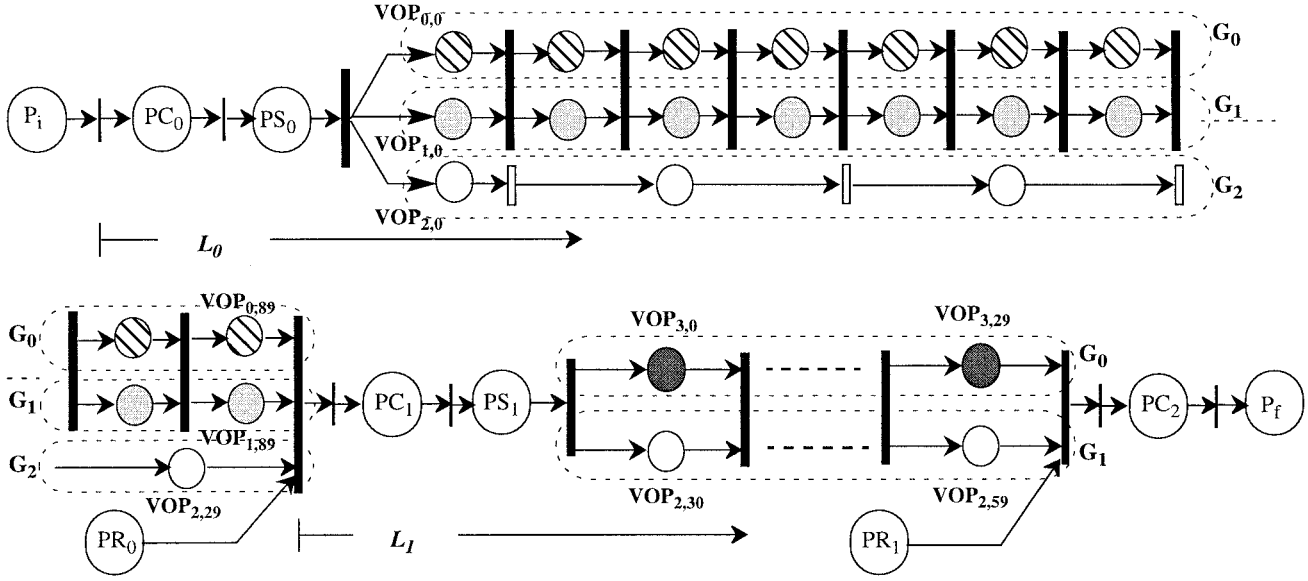


Fig. 7. MOCPN model for GS scheduling.

stations each. Basically, such mapping describes the workstations distribution by the scheduling algorithm;

ν_k partitioned data area of the VOP to P_k.

1) The RRS Algorithm: The RRS algorithm using the EDF rule sorts all VOP's in a nondecreasing order of their playout deadlines $d_{i,j}$. If two VOP's have the same playout deadline, the smallest processing time (SPT) is applied, that is, a VOP with a smaller size precedes the one with a larger size. A data partitioning method [21] partitions each VOP into a number of pieces equal to the number of workstations. The RRS algorithm allocates the pieces to the workstations in a round-robin fashion.

The RRS Algorithm:

1) Initialize the scheduling interval L_x, where $x = 0$.
2) Sort all $VOP_{i,j}$ in V using EDF rule with SPT rule for the tie breaking.
3) Initialize $R_k = \{\}$ and $g_0 = K$, where $P_k \in G_0$, $k \in [0, K-1]$.
4) Point to the first VOP of V.
5) Schedule R_k to the pointing VOP: $R_k = R_k \cup \{VOP_{i,j}\}$.
6) Partition the VOP and map the data area ν_k to P_k, where $\nu_k \approx \lceil S_{i,j}/K \rceil$.
7) Advance the pointer to the next VOP along the V if no interaction received. Otherwise, trigger the scheduler R_k to update the state of the video objects and begin the next scheduling interval L_x, where $x = x + 1$, go to step 2.
8) Repeat the last three steps until the end of the sequence V.

Fig. 6 is the MOCPN of the scheduling V ordered by the EDF rule with two scheduling place PS. The user replaces the objects at t_1 and requests to stop the entire session at t_2, and the entire sequence can be treated as two scheduling intervals L_0 and L_1.

The RRS algorithm enforces the workstations to encode each VOP concurrently. By storing all reference data in the local memory, each workstation is able to process the partitioned ν_k locally, and no data exchange is required ($T_{com} = 0$). The scheduling time for sorting the VOP's can also be neglected ($T_s \approx 0$) because the calculation and comparison of (1) is very fast. Moreover, such RRS algorithm can adapt to the variations in $S_{i,j}$ automatically. Since each workstation processes the different partition region ν_k of video objects concurrently, if the $S_{i,j}$ becomes larger, the data area ν_k will also be enlarged, and vice versa. Each workstation may

TABLE II
WORKSTATIONS DISTRIBUTION BY GS ALGORITHM

Video object	Size ratio	4 workstations	8 workstations	12 workstations	16 workstations	20 workstations
VO_0	0.7	1(25%)	5(62%)	8(67%)	11(69%)	14(70%)
VO_1	0.2	1(25%)	1(12.5%)	2(17%)	3(19%)	4(20%)
VO_2	0.05	1(25%)	1(12.5%)	1(12.5%)	1(6%)	1(5%)
VO_3	0.05	1(25%)	1(12.5%)	1(12.5)	1(6%)	1(5%)

TABLE III
OBJECT MERGING PROCEDURE (WORKSTATIONS = 4)

Video object	Size ratio	Workstations	Condition in Equation (9)	1st merging		Condition in Equation (9)	2nd merging		Condition in Equation (9)
				Size ratio	Workstations		Size ratio	Workstations	
VO_0	0.7	1(25%)	0.05<0.125 **continue**	0.7	2(50%)	0.1<0.125 **continue**	0.7	3(75%)	0.3 > 0.125 **stop**
VO_1	0.2	1(25%)		0.2	1(25%)		0.3	1(25%)	
VO_2	0.05	1(25%)		0.1	1(25%)				
VO_3	0.05	1(25%)							

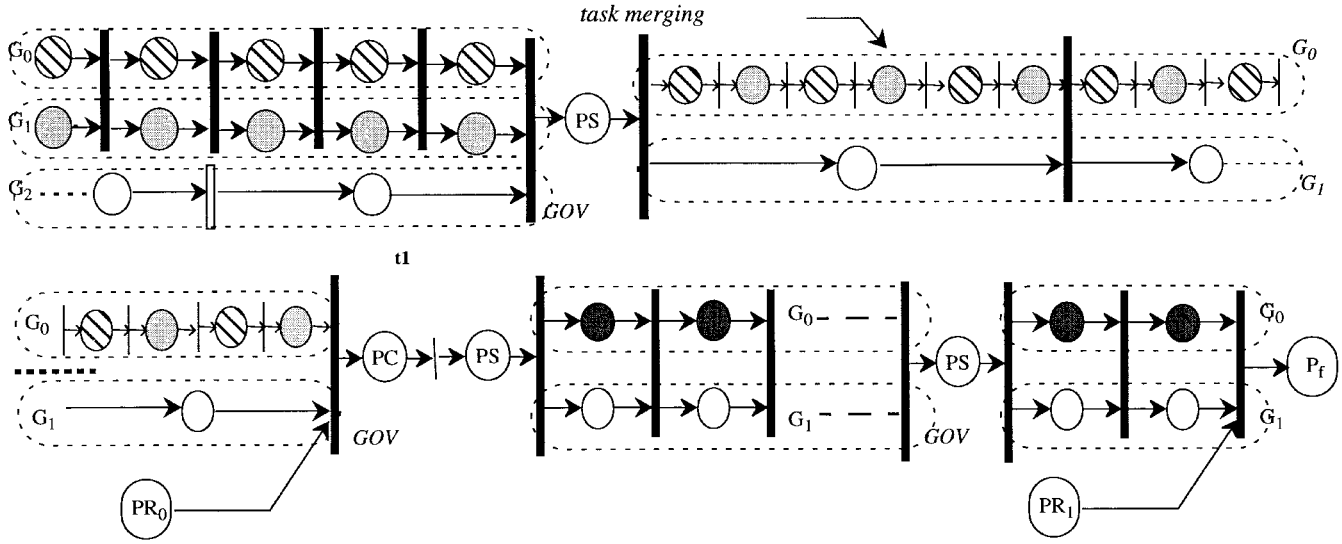


Fig. 8. MOCPN model of GAS algorithm.

spend more time on larger VOP's and less time on smaller VOP's simultaneously.

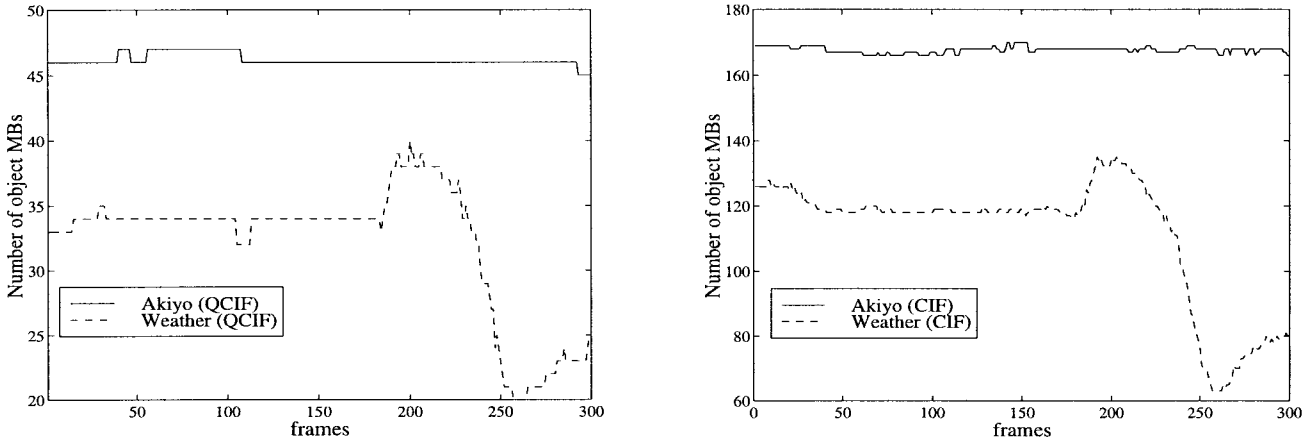
In order to minimize the interprocessors communication, the reference data of the video object is stored in the local memory for motion estimation. Because the RRS algorithm can respond to the user interactions and perform the rescheduling very quickly, it is suitable for the applications supporting interactivity from multiple clients in real time.

2) The GS Algorithm: According to the video encoder structure specified in MPEG-4, each video object is encoded independently. Therefore, we can divide the available number of workstations K into N groups ($G_0, G_1, \cdots, G_{N-1}$), each containing g_i workstations, that is $\sum_{i=0}^{N-1} g_i = K$.

The number of workstation groups is equal to the number of existing video objects, and each group handles one video object. The number of workstations in a group g_i is proportional to the shape size and playout duration. That is, a larger object is assigned more workstations. In addition, an object with a shorter playout durations is assigned more workstations. After the scheduling, each video object can be encoded concurrently by a group of workstations with further data partition approach. In order to minimize the scheduling overhead cost, the GS algorithm determines the workstation assignment only at the beginning of the each scheduling interval L_x.

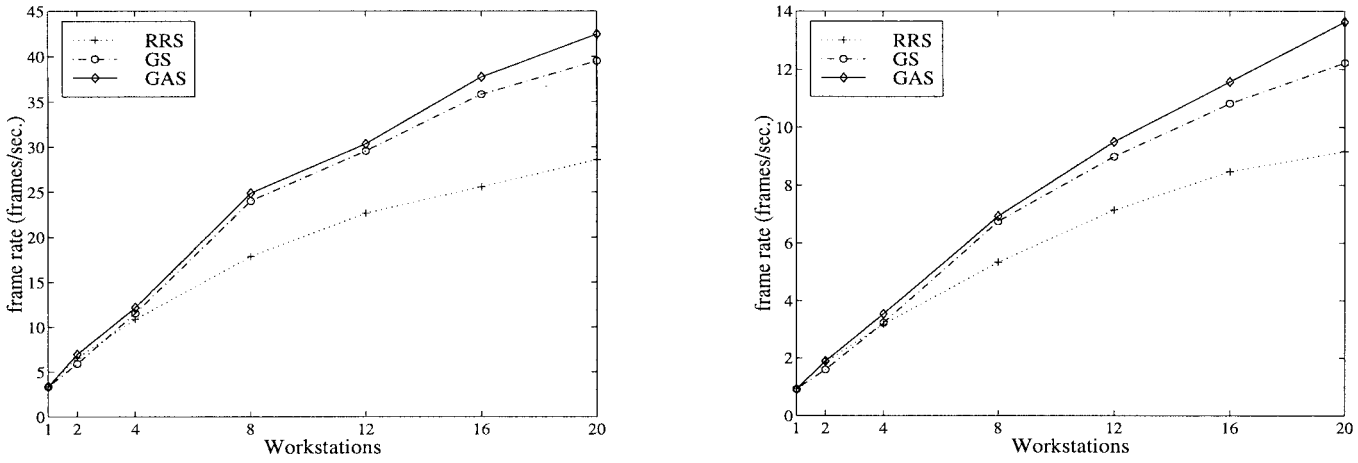
GS Algorithm:

1) Initialize the scheduling interval L_x, where $x = 0$.



(a) (b)

Fig. 9. Size variation of VO_0 "Akiyo" and VO_1 "Weather". (a) QCIF format and (b) CIF format.



(a) (b)

Fig. 10. Encoding rate of video session with two video objects (VO_0 "Weather" and VO_1 "Akiyo"). (a) QCIF format and (b) CIF format.

2) Sort VOP's of existing video object VO_i in V using EDF rule.
3) Measure the shape size of the first $VOP_{i,j}$ along L_x as $\tilde{S}_{x,i}$ and its playout duration as $\tau_{x,i}$.
4) Initialize $R_k = \{\}$ and calculate the g_i for workstations distribution. Let

$$\eta_i = K \cdot \frac{\tilde{S}_{x,i'}/\tau_{x,i'}}{\sum_{i=0}^{N-1}(\tilde{S}_{x,i}/\tau_{x,i})}$$

$$g_{i'} = \begin{cases} \lfloor \eta_{i'} \rfloor, & \text{if } \lfloor \eta_{i'} \rfloor \geq 1 \\ 1, & \text{otherwise} \end{cases} \quad (7)$$

where

$$i = 1, 2, \cdots, N-2$$

and

$$g_{N-1} = K - \sum_{i=1}^{N-2} g_i.$$

5) Point to the first $VOP_{i,j}$ of each VO along L_x.
6) Schedule R_k to the pointing VOP along the V : $R_k = R_k \cup \{VOP_{i,j}\}$, $P_k \in G_i$, and $VOP_{i,j} \in VO_i$.
7) Partition the VOP and map the data area ν_k to P_k, where $\nu_k \approx \lceil S_{i,j}/g_i \rceil$.
8) Point to the next $VOP_{i,j+1}$ along the V if no user interaction received. Otherwise, update the video object VO_i and begin the next scheduling interval L_x, where $x = x + 1$, and go to step 2.
9) Repeat the last three steps until the end of the sequence V_i.

Fig. 7 shows the MOCPN of GS scheduled V ordered by the EDF rule with two scheduling intervals L_0 and L_1 bounded by the user interaction. The workstation group assigned to a video object VO_i is represented by G_i as attached on the net.

The communication cost due to data exchange within each group can be avoided ($T_{com} = 0$) by storing the reference data in local memory for motion estimation. Generally, the performance of GS algorithm is better than RRS algorithm. However, since each video object has to be assigned to at least one workstation, load imbalance may occur when the sizes of video objects are different or the number of

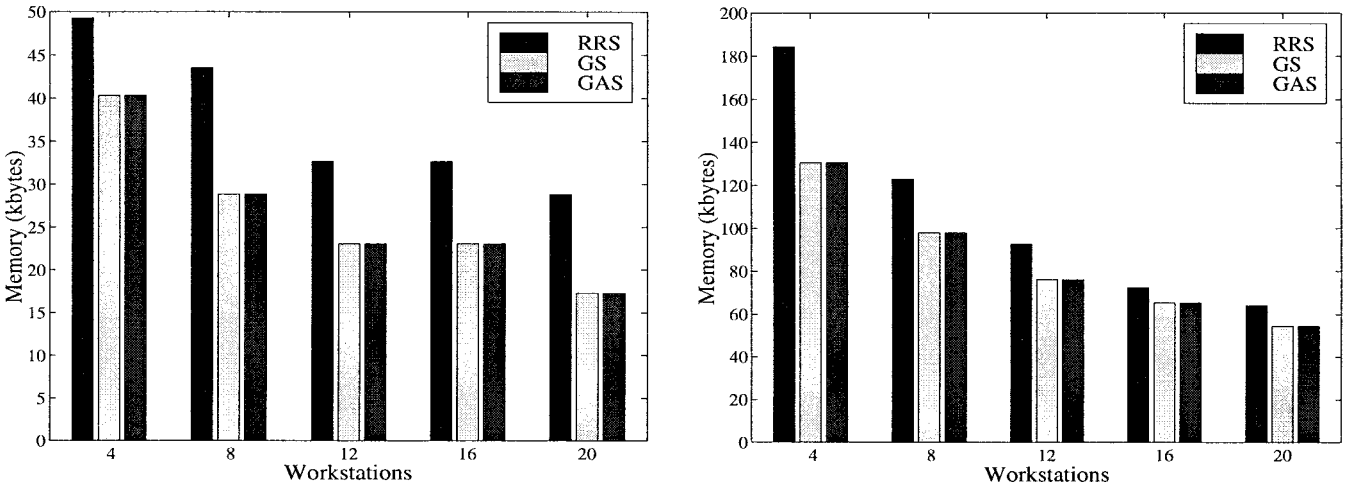


Fig. 11. Local memory requirement. (a) QCIF format. (b) CIF format.

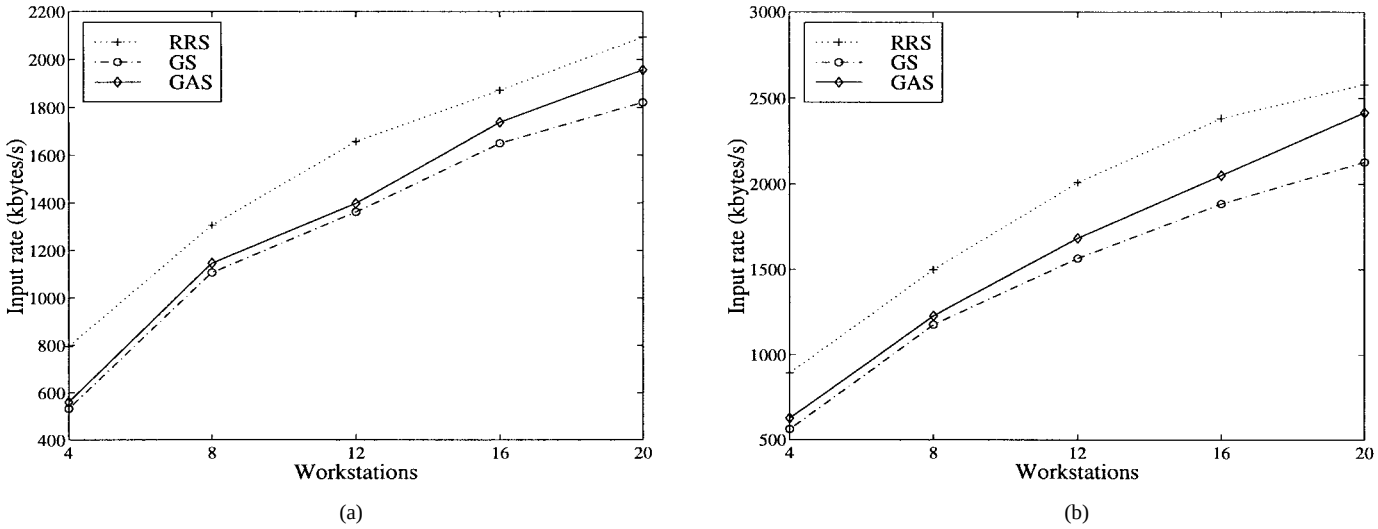


Fig. 12. Input data rate. (a) QCIF format. (b) CIF format.

workstations is limited. For example, if the size ratio between two video objects such as VO_0 and VO_1 is 1 : 10 and just two workstations are available, the workstation assignment ratio will be 1 : 1, and consequently, the workload of workstation P_1, which is scheduled to encode VO_1, will be larger than that of P_0 scheduled to encode VO_0. Such a load imbalance may lead to performance degradation.

3) The GAS Algorithm: In order to overcome the load unbalancing problem in the GS algorithm, we can observe the object variation and adjust the workstation configuration of each group dynamically. However, this may introduce a high interprocessor communication cost due to the collection of the load information. Furthermore, because each workstation stores the previously reconstructed data for motion estimation in its local memory, when the workstation is scheduled to encode a new video object, it has to restore the related reference data from other workstations. Therefore, the data structure of each processor (a workstation in our case) and group configuration should be modified according to the changes in the data partitioning. Such an overhead and communication latency may outweigh the benefit gained from the load balancing. For a tradeoff, the GAS scheduling algorithm periodically detects the workload information of the workstations and performs rescheduling. In order to minimize the interprocessor communication cost, the period is based on GOV. GOV is an optional syntax level specified by the standard for random access and error recovery purpose. The GOV header is followed by the I-VOP performing intracoding which is independent of the previous VOP's. Therefore, the change in workstation assignment will not introduce an additional interprocessor communication.

Another problem with the GS algorithm is that the distribution of the workstations may not be proportional to the size of the video objects. This situation occurs when the percentage of the workstations assigned to the object is larger than the

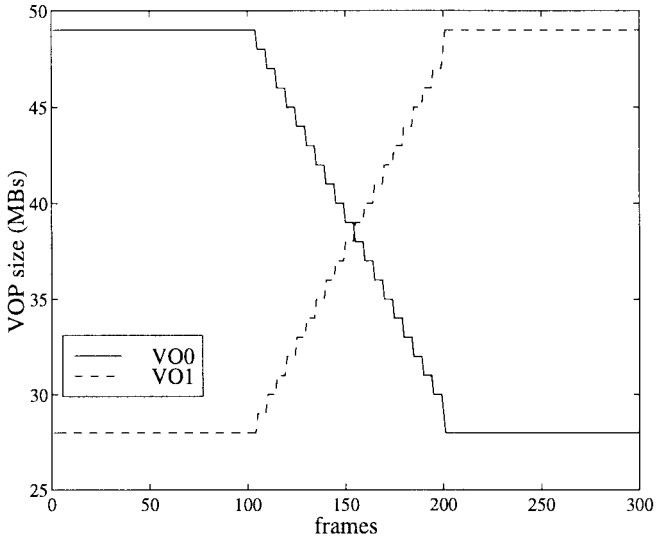


Fig. 13. Size variation of an example sequence with two video objects (VO_0 and VO_1).

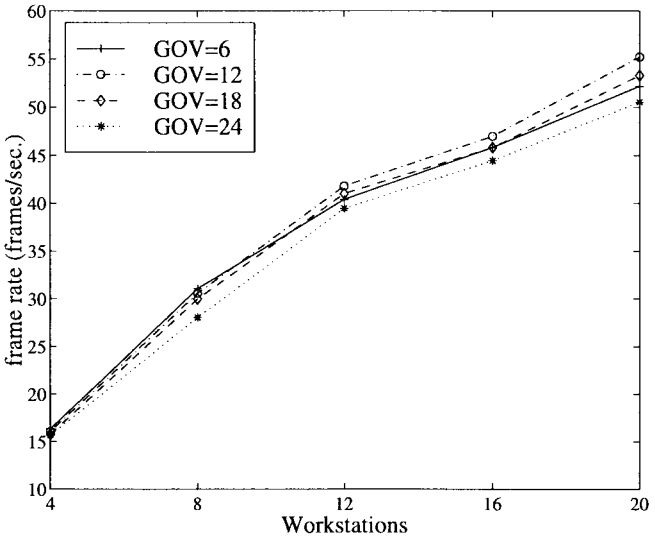


Fig. 15. Comparison of the encoding performance with different GOV length.

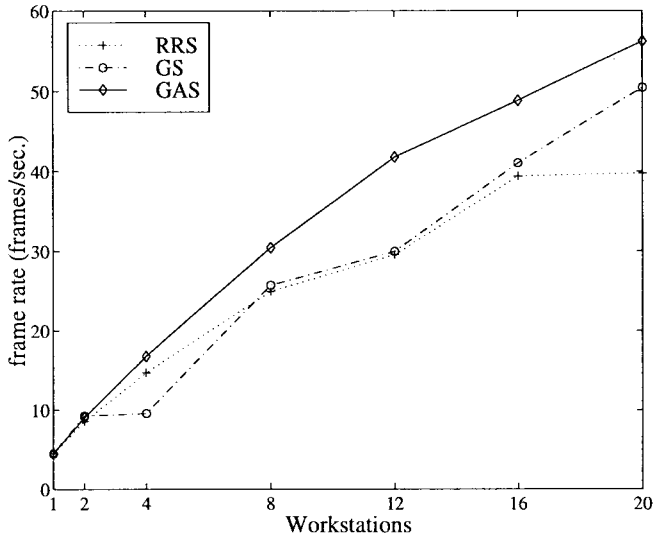


Fig. 14. Encoding frame rate.

object size ratio as given by

$$\frac{g_{i'}}{K} \gg \frac{\alpha_{i'} \cdot \tilde{S}_{x,i'}/\tau_{x,i'}}{\sum_{i=0}^{N-1} (\alpha_i \cdot \tilde{S}_{x,i}/\tau_{x,i})} \tag{8}$$

and $\tilde{S}_{x,i'}$ is the size of I-VOP of the video object $VO_{i'}$ during the scheduling interval L_x.

For example, suppose we have four video objects whose sizes are 14, 4, 1, and 1 unit, respectively ($\alpha_i = 1$ and $\tau_i = 1$). For this case, Table II illustrates the workstations assignment generated by the GS algorithm. As can be observed, the load imbalance is significant when the number of workstations is below 12.

By comparing the size ratio of the smallest object (e.g., VO_0) and the minimum percentage of the workstation that can be assigned to that object, we can estimate the load imbalance caused by using GS algorithm

$$\frac{\alpha_0 \cdot \tilde{S}_{x,0}/\tau_{x,0}}{\sum_{i=0}^{N-1} (\alpha_i \cdot \tilde{S}_{x,i}/\tau_{x,i})} < \frac{1}{\beta \cdot K} \tag{9}$$

where β is a weak coefficient; a larger β value means more tolerance to load imbalance. Generally, we set β equal to 2 in our experiments. To overcome load imbalance, one feasible solution is to merge the smallest objects together recursively until (9) no longer holds. There can be various merging approaches [22]. In our approach, task merging is done by finding a pair of video objects that have the minimum computation cost and then merging these objects. The merging method is recursively applied until the imbalance condition (9) no longer holds.

Table III illustrates such operations using the previous example when the number of workstations is 4. According to the GS decision, each video object is assigned one workstation, and the imbalance is observed by examining (9). Then, VO_3 and VO_2 are merged together, and the workstations are reassigned. Equation (9) still holds. Therefore, we merge VO_1 with the VO_2 and VO_3 and redistribute the workstations again. Now, the load imbalance can be ignored as (9) no longer holds. When next GOV begins, such rescheduling is performed again since the states of the multiple objects may be changed by the user interactions.

GAS Algorithm:

1) Initialize the scheduling interval length L_x as the length of GOV, defined as γ.
2) Sort VOP's of each existing video object VO_i in V using the EDF rule.
3) Measure the shape size of the first I-VOP along the L_x as $\tilde{S}_{x,i}$.
4) Adjust the ratio coefficient α_i as follows:

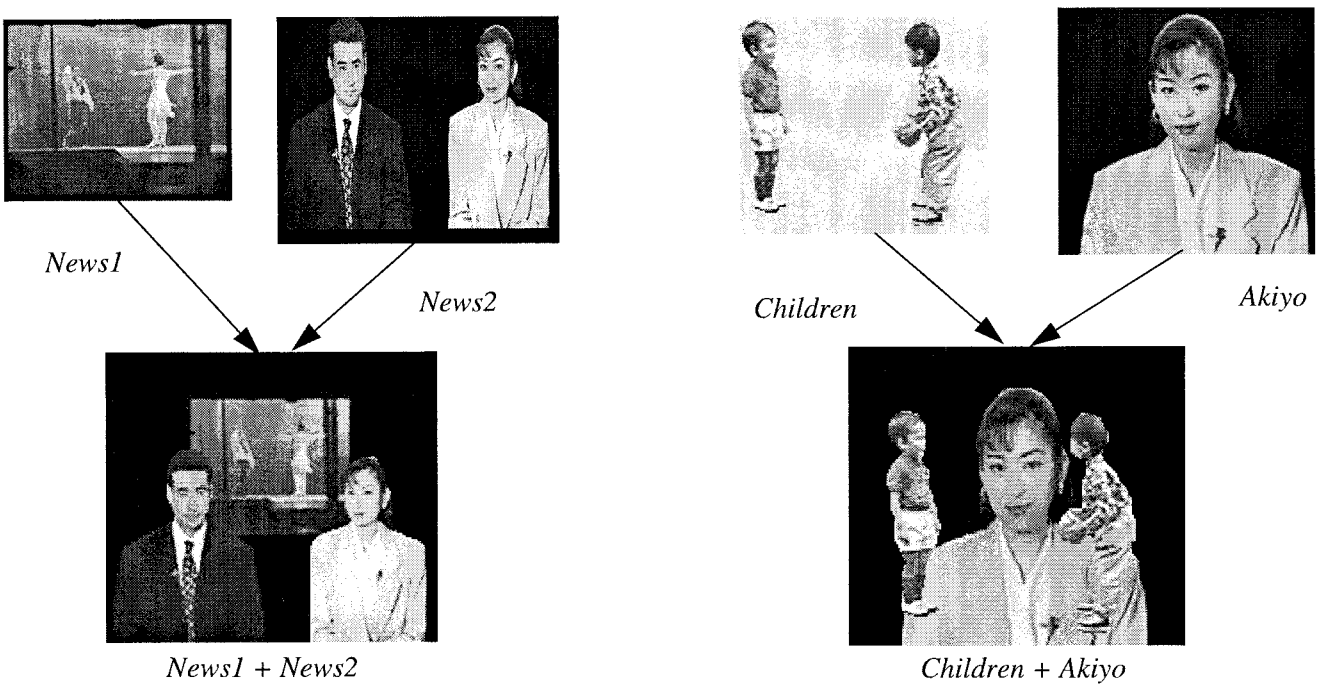


Fig. 16. Samples of sequences with multiple video objects.

Let ΔT_i be the average processing time of the previous GOV of VO_i, and the length of GOV is γ

$$\Delta T_i = \frac{\sum_{j=0}^{\gamma-1} \left(\max_k \{T_{enc}^k(i,j)\} \right)}{\gamma}$$

and average object size

$$\Delta S_i = \left(\sum_{j=0}^{\gamma-1} S_{i,j} \right) \Big/ \gamma$$

and then, we updated α_i as

$$\alpha_i = \frac{\Delta T_i \cdot g_i}{\Delta S_i}. \tag{10}$$

5) Initialize $R_k = \{\}$ and calculate the value of g_i for workstations distribution as in

$$\eta_{i'} = K \cdot \frac{\alpha_{i'} \cdot \tilde{S}_{x,i'}/\tau_{x,i'}}{\sum_{i=0}^{N-1} (\alpha_i \cdot \tilde{S}_{x,i}/\tau_{x,i})}$$
$$g_{i'} = \begin{cases} \lfloor \eta_{i'} \rfloor, & \text{if } \lfloor \eta_{i'} \rfloor > 1 \\ 1, & \text{otherwise} \end{cases} \tag{11}$$

where

$$i = 0, 1, 2, \cdots, N-2$$

and

$$g_{N-1} = K - \sum_{i=1}^{N-2} g_i. \tag{12}$$

6) Calculate (9) to detect the load imbalance and perform merging operations accordingly.
7) Point to the first $VOP_{i,j}$ of V.
8) Schedule R_k to the pointing $VOP_{i,j}$ with $P_k \in G_i$ and $R_k = R_k \cup \{VOP_{i,j}\}$.
9) Partition the VOP and map the data area ν_k to P_k, where $\nu_k \approx S_{i,j}/g_i$.
10) Advance the pointer to the next $VOP_{i,j+1}$ along the V.
11) Repeat the last three steps until the end of the GOV.
12) Detect the interaction queue and update the object status.
13) Go to step 2, and start the next GOV scheduling until the end of the video session.

Step 4 is used to predict the processing time needed for the object according to the processing time statistics of the previous GOV. Such a prediction adjustment makes the distribution of the workstations more precise since video objects may have different processing time requirements along the time due to its size or movement change and the algorithm used.

Because MPEG-4 only specifies a standard coded video syntax and decoding procedure and most choices in the encoding methods are left open, the GOV length can be changed by the encoder as will for the purpose of random access. It is desirable to select the identical length for all video objects when using GAS scheduling scheme. If a different object has a different access requirement and, hence, a different GOV length, the schedule adjustment can only be performed at the least common multiple length of all the objects.

Fig. 8 is an example of GAS scheduling MOCPN model where GAS algorithm performs at the beginning of each GOV.

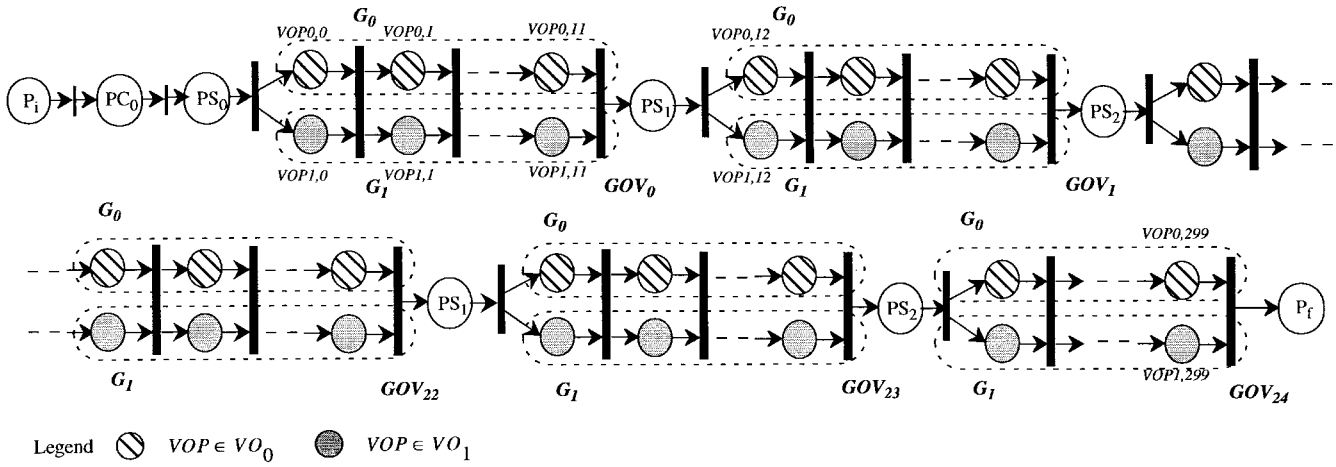


Fig. 17. MOCPN model of GAS scheduling on the sequence "Akiyo + Weather" in QCIF format.

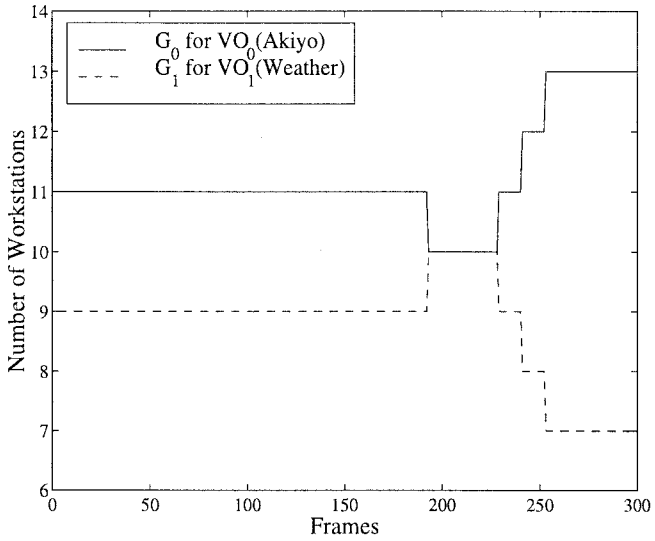


Fig. 18. Workstations assignment with GAS scheduling scheme.

It also shows a scenario when VO_0 and VO_1 are merged together some time during the session.

IV. Experimental Results

We have implemented the encoder using the proposed scheduling schemes on a number of test sequences. Our video encoder uses the MPEG-4 video verification model (VM8.0). The segmentation information of the object is assumed to be available already. Our computing platform is a cluster of 20 Sparc Ultra 1 workstations connected by a ForeSystems ATM switch (ASX-1000) providing fast communication among the workstations. The cluster is configured as a virtual 2-D processor grid, with each workstation having its own x–y coordinates. For interprocessor communication and synchronization, we use *message passing interface* (MPI) [23], ensuring the portability of the encoder across various platforms. Furthermore, we have used various additional software optimization, such as a fast motion estimation algorithm [24], *visual instruction set* (VIS) and compiler optimization, for performance improvement in the encoding speed.

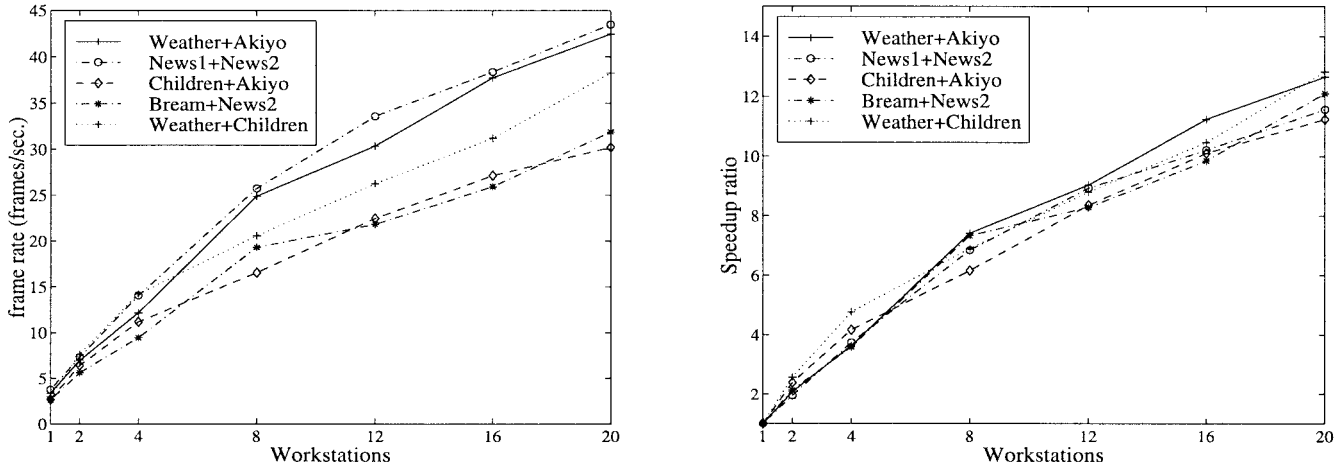
First, we designed a test sequence with two foreground video objects retrieved from the standard test sequences "Akiyo" and "Weather." "Akiyo" is a head-shoulder object with small movements, while the object size of "Weather" changes rapidly during the last 100 frames. Fig. 9(a) and (b) show the size variations of the two objects with QCIF or CIF format, respectively, during the entire session.

Fig. 10 shows the encoding frame rates (one VOP equal to one frame) achieved by the encoder using the three scheduling algorithms with various numbers of workstations. The encoder can achieve frame rate higher than the real-time performance (30 frames/s) using the QCIF sequence. For the two CIF objects sequence, a frame rate close to 14 frames/s is obtained by using 20 workstations. Generally, the performance of GS is better than RRS when sufficient workstations are available. The performance of GAS is the best among the three scheduling algorithms due to its periodical adjustments, even though more scheduling overhead is incurred.

Fig. 11 presents the local memory requirement of the workstation for storing the reference data to perform the motion estimation. RRS requires more memory than GS and GAS since it has to store the reference data for all video objects, while GS and GAS only store the reference of a single video object. The memory requirement is reduced as the number of workstations increases since the larger the number of the workstations, the smaller the partitioned data area assigned to each workstation.

Fig. 12 illustrates the average input rate per workstation during the encoding. Because the input video data must be available before the related encoding begins, higher encoding frame rate requires higher input data rate. With RRS algorithm, each workstation has to read all the available VOP's. On the other hand, with GS or GAS algorithm, each workstation just reads the VOP's of a single object. Therefore, the input rate of RRS is higher than other two algorithms.

We also designed a sequence with two video objects whose shape sizes change greatly and symmetrically, as shown in Fig. 13. Such a sequence can provide a better comparison among the algorithms.



(a) (b)

Fig. 19. Encoder performance. (a) Encoding rate. (b) Overall speedup ratio.

For this sequence, the performance of each scheduling scheme is illustrated in Fig. 14. The results indicate that under different conditions, GS or RRS may outperform each other. On the other hand, GAS can achieve the best performance since it combines the advantages of both GS and RRS algorithms.

Fig. 15 provides the relations between the length of the GOV and the encoding performance. As the length of GOV becomes shorter, the periodical adjustment may adapt to the dynamic behavior of the video object more precisely, and thereby, an efficient load scheduling and improved coding speedup can be achieved, but the encoding performance may also be reduced due to the high frequent scheduling overhead when the period is too short.

We also tested the GAS algorithm on several composed sequences. All of the video objects, such as "Akiyo," "News1," and "Weather," as labeled in Fig. 19, are obtained from the MPEG-4 standard test library with QCIF format and represent various characteristics in terms of spatial detail and movement. Fig. 16 shows two samples of the sequence with two video objects each, "News1 + News2" and "Akiyo + Children."

Fig. 17 is the MOCPN model for the sample sequence "Akiyo + Weather" using GAS scheduling scheme, and Fig. 18 presents the variation of the workstations number assigned to each video objects within each GOV whose length is 12 frames.

Fig. 19(a) shows the encoding frame rate achieved by the GAS algorithm for these sequences using various number of workstations. The encoder achieves a frame rate higher than the real-time performance (30 frames/s) on most standard test sequences. Fig. 19(b) shows the overall speedup, indicating that the performance of the encoder can scale well with the number of workstations used.

To evaluate the performance of the scheduling algorithm with respect to the user interactions, we simulated the user interactions on the sequence and varied the request interarrival rate. The experiments were performed on the sequence "Weather + Akiyo." Fig. 20 depicts the response and schedule time using different algorithms on eight workstations. As

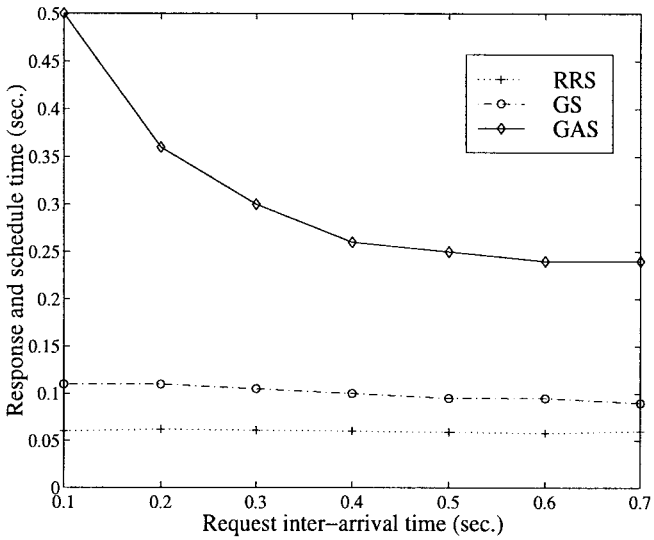


Fig. 20. Response and schedule time with user interactions.

indicated by these results, the RRS algorithm outperforms both GS and GAS in terms of response and schedule time, implying that it is suitable for the environment with high frequent client interactions. The response time of all the algorithms decreases (more so for the GAS algorithm) as the request interarrival time increases. Because the GAS algorithm performs the rescheduling periodically, with more frequent interactions, new requests suffer from the queuing delay and the response time becomes larger.

Next, we present the performance of the full system encoder for various media objects each using different object encoders on the workstations cluster. Table IV shows various media objects and the encoder used in the experiment, respectively.

Because the processing time of each media object is significantly different as Fig. 21 shows, we used control parallelism to merge audio, image, FAP, and BIFS objects together running on a single workstation, and perform GAS scheduling scheme on the video processing. As Fig. 22 shows, with 12

TABLE IV
MEDIA OBJECTS AND ENCODERS

Media type	Object file name	Encoder
Video	*Akiyo (qcif) (200 frames)*	Video encoder VM 8.0
Video	*Weather (qcif) (200 frames)*	Video encoder VM 8.0
Audio	*Pies (200 frames)*	Reference software refsoft980508
Image	*Lena512 (1 frames)*	CJPEG encoder, ver. 6a
Facial Animation (FAP)	*Opossum2 (ASCII file)*	Rockwell FAP encoder rockwell_v11
BIFS	*Scene (ASCII file)*	BIFS encoder t2b

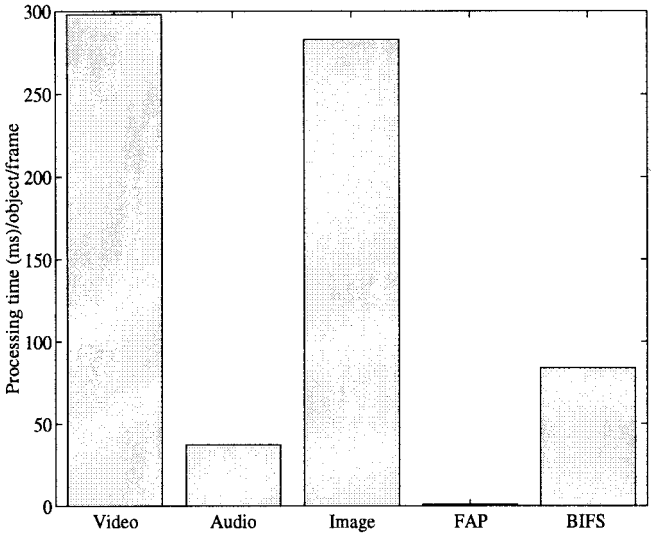


Fig. 21. Average processing time of each media object on a single workstation.

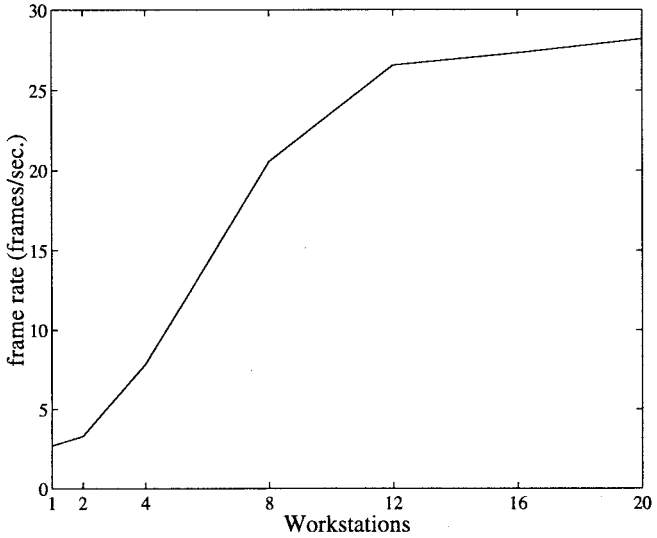


Fig. 22. Encoding frame rate of various media objects composed sequence.

workstations, the entire encoder can achieve an encoding rate close to 25 frames/s on the composed sequence with various types of media objects.

V. CONCLUSIONS

In this paper, we have presented an object-based MPEG-4 system encoder using a cluster of workstations. The encoder allows user interactions. Allocation of video objections to workstations for encoding is done by using a modeling scheme that describes the dynamic behavior of the multiple video objects and user interactions during the entire MPEG-4 video session. We have proposed three scheduling schemes for assigning the encoding tasks to the workstations with proper load balancing. Each algorithm has its own tradeoff between performance and complexity and, therefore, is suitable to different application environments. The encoder can achieve a real-time encoding rate on some composed sequences which demonstrate its potential to be used in a real system. In our future work, we will explore MPEG-4 decoders and interactive rendering methodology for supporting multimedia communication and buffer management between the server and clients to complete our interactive multimedia system.

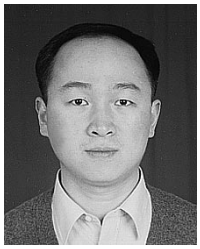
ACKNOWLEDGMENT

The authors would like to thank Dr. Y.-Q. Zhang of the Sarnoff Corporation for technical support.

REFERENCES

[1] L. Chiariglione, "MPEG and multimedia communications," *IEEE Trans. Cicuits Syst. Video Techol.*, vol. 7, pp. 5–18, Feb. 1997.
[2] ISO/IEC, "MPEG-4 overview," JTC1/SC29/WG11 N2564, Dec. 1998.
[3] B. G. Haskell *et al.*, "Image and video coding—Emerging standards and beyond," *IEEE Trans. Cicuits Syst. Video Techol.*, vol. 8, pp. 814–837, Nov. 1998.
[4] M. L. Liou, "Overview of the $p \times 64$ kbps video coding standard," *Commun. ACM*, vol. 34, no. 4, pp. 59–63, Apr. 1991.
[5] Draft ITU-T Recommendation H.263, "Video coding for narrow telecommunication channel at <64 kbit/s," Apr. 1995.
[6] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, no. 4, pp. 46–58, Apr. 1991.
[7] ISO Committee Draft 13818-2, "Generic coding of moving pictures and associated audio: Recommendation H.262," ISO/IEC JTC1/SC29/WG11, Nov. 1993.
[8] ISO/IEC, "MPEG-4 applications document," JTC1/SC29/WG11 N2563, Dec. 1998.
[9] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Trans. Cicuits Syst. Video Techol.*, vol. 7, pp. 19–31, Feb. 1997.
[10] ISO/IEC, "MPEG-4 video verification model 8.0," JTC1/SC29/WG11 N1796, July 1997.
[11] B. Furht, "Multimedia systems: An overview," *IEEE Multimedia*, vol. 1, pp. 47–59, Spring 1994.

[12] F. Tompa, "A data model for flexible hypertext database system," *Inform. Services Use*, vol. 7, pp. 85–100, Jan. 1989.
[13] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, pp. 5–35, Apr. 1989.
[14] Y. Masunaga, "An object-oriented multimedia database management system," *J. Inform. Process.*, vol. 14, pp. 60–74, 1991.
[15] R. Steinmeta, "Synchronization properties in multimedia systems," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 401–412, Apr. 1990.
[16] D. Anderson *et al.*, "Support for continuous media in the dash system," in *Proc. 10th Int. Conf. Distrib. Comput. Syst.*, May 1990, pp. 54–61.
[17] M. Woo, N. U. Qazi, and A. Ghafoor, "A synchronization framework for communication of pre-orchestrated multimedia information," *IEEE Network*, vol. 8, pp. 52–61, Jan. 1994.
[18] Y. K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, pp. 506–521, May 1996.
[19] G. C. Buttazzo, *Hard Real-Time Computing Systems.* Boston, MA: Kluwer, 1997.
[20] P. Brucker, *Scheduling Algorithms.* New York: Springer, 1998.
[21] Y. He, I. Ahmad, and M. L. Liou, "A shape-adaptive partitioning method for MPEG-4 video encoding," in *Proc. Int. Conf. Electron., Circuits, Syst.*, 1998, pp. 239–242.
[22] J. C. Liou and M. A. Palis, "A comparison of general approaches to multiprocessor scheduling," in *Proc. 11th Int. Parallel Process. Symp.*, 1997, pp. 152–156.
[23] D. W. Walker and J. J. Dongarra, "MPI: A standard message passing interface," *Supercomput.*, vol. 12, no. 1, pp. 56–68, Jan. 1996.
[24] Z. L. He and M. L. Liou, "A high-performance fast search algorithm for block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 826–828, Oct. 1997.



Yong He (S'97) was born in Shanghai, China. He received the B.Eng. and M.Eng. degrees from Southeast University, NanJing, China, in 1992 and 1995, respectively. He is currently a Ph.D. candidate in the Department of Electrical and Electronic Engineering at the Hong Kong University of Science and Technology, Hong Kong.

His research interests include image processing, video coding techniques, parallel and distributed computing, and scheduling algorithms.

Ishfaq Ahmad (M'92) received the B.Sc. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 1985 and the M.S. degree in computer engineering and the Ph.D. degree in computer science from Syracuse University, Syracuse, NY, in 1987 and 1992, respectively.

Currently, he is an Associate Professor with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong. His research interests include various aspects of parallel and distributed computing, high-performance computer architecture and their assessment, multimedia systems, and video coding. He has published extensively in the above areas.

Dr. Ahmad received the Best Student Paper Awards at Supercomputing'90 and Supercomputing'91. He has been a Guest Editor for two special issues of *Concurrency: Practice and Experience* and is guest editing a forthcoming special of the *Journal of Parallel and Distributed Computing*. He has served on the program committees of various international conferences and is a Member of the IEEE Computer Society.

Ming L. Liou (M'63–SM'78–F'79) received the B.S. degree from National Taiwan University, Hsinchu, Taiwan, R.O.C., the M.S. degree from Drexel University, Philadelphia, PA, and the Ph.D. degree from Stanford University, Stanford, CA, in 1956, 1961, and 1964, respectively, all in electrical engineering.

He joined the faculty of the Department of Electrical and Electronic Engineering, the Hong Kong University of Science and Technology, Hong Kong, as a Professor in October 1992 and was appointed as the Director of Hong Kong Telecom Institute of Information Technology in January 1993. His current research interests include very low bit-rate video, motion estimation techniques, packet video, HDTV, VLSI architecture, and implementation of parallel and distributed computing systems for visual applications. From 1984 to 1992, he was a Director at Bellcore, Red Bank, NJ, conducting research in data transmission, digital subscriber line transceivers, and video technology. He joined AT&T Bell Labs in 1963 as a Member of Technical Staff and had held various supervisory positions until 1984, when he transferred to Bellcore. During his career at AT&T Bell Labs, he did research on numerical analysis, system theory, FM distortion analysis, and computer-aided design of communication circuits and systems, including circuits containing periodically operated switches. He has published numerous papers in various fields.

Dr. Liou received the IEEE Circuits and Systems Society (CAS) Society Special Prize Paper Award in 1973 and the Darlington Prize Award in 1977. He has been very active in professional activities and served in various capacities, including Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1979 to 1981, Executive Vice President of the CAS Society in 1986, where he was responsible for regional activities, President of the CAS Society in 1988, the founding editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY from 1991 to 1995, and General Cochair of the 1997 IEEE International Symposium on Circuits and Systems held in Hong Kong. He is a member of Sigma Xi, Eta Kappa Nu, Phi Tau Phi, the Hong Kong Institution of Science, and a Fellow of the Hong Kong Institution of Engineers.