Sensor measurement scheduling: an enhanced dynamic, preemptive algorithm

Gregory A. McIntyre, STUDENT MEMBER SPIE
Kenneth J. Hintz, MEMBER SPIE
Department of Electrical and Computer Engineering
Mail Stop 1G5
George Mason University
Fairfax, VA 22030
E-mail: gmcintyr@bzy.gmu.edu
khintz@gmu.edu

Abstract. This paper presents an enhanced architecture for a sensor measurement scheduler as well as a dynamic sensor scheduling algorithm called the On-line, Greedy, Urgency-driven, Preemptive Scheduling Algorithm (OGUPSA). The premise is that the function of sensor management can be partitioned into the two tasks of information management, essentially an information to measurement mapping, and a sensor scheduler which takes the measurement requests along with their priorities and optimally maps them to a set of sensors. OGUPSA was developed using the three main scheduling policies of Most-Urgent-First to pick a task, Earliest-Completed-First to select a sensor, and Least-Versatile-First to resolve ties. By successive application of these policies. OGUPSA dynamically allocates, schedules, and distributes a set of measurement tasks from an information manager among a set of sensors. OGUPSA can detect the failure of a measurement task to meet a deadline and improves the dynamic load balance among all sensors while being a polynomial time algorithm. One of the key components of OGUPSA is the information in the applicable sensor table. This table is the mechanism that is used to assign requested tasks to specific sensors.

Subject terms: sensor scheduling, sensor management, resource scheduling, sensors

1 Introduction

Multisensor fusion has become a heavily researched area for modern sensor systems with sensor management being one of its key element. It is imperative that sensors operate synergistically in order to take full advantage of each sensor's strengths while not underutilizing any other sensors. This is accomplished through the use of a sensor manager which can be defined as a "…process which seeks to manage or coordinate the use of sensing resources in a manner that improves the process of data fusion and ultimately that of perception, synergistically[1]." A sensor management system can be expected to

- Reduce the workload for the operator by automating sensor allocation, moding, pointing, and emission control[2]
- Prioritize and schedule service requests to meet both integrated flight management and weapon control requirement[2]
- Aid in sensor data fusion by coordinating fusion requests with the data collected from different sensor and sensor modes[2]
- Support reconfiguration and degradation due to the partial or total loss of sensors or sensor modes[2]
- Develop a sensor schedule that optimizes (or at least suboptimizes) the use of the available sensor assets
- Communicate desired actions to the individual sensors

Given these functions, a sensor management system can be partitioned into two separate processes. The first process is the information manager that determines what measurement tasks are required in order to maintain a specified level of knowledge about the environment or information rate as well as the time by which these measurement tasks are required to be completed. Its ultimate goal is to optimally task a sensor scheduler to make measurements. The second process is the sensor scheduler itself. Stated another way, sensor management (consisting of the information management and sensor scheduling) can be considered as a resource-constrained scheduling problem. Scheduling involves the allocation of resources (including time) to measurement tasks such that the desired objective function is minimized (or maximized). The resource allocation problem is, therefore, little more than a synonym for the scheduling problem.

Popoli[3] describes the sensor scheduling problem as "…given the ability to decide which tasks are important…, how do we set up a time line of tasks for the sensor to perform." He then proceeds to describe two general approaches for scheduling sensor tasks, best first and brick packing. Best first is a myopic method that schedules tasks according to measurement task priorities. Brick packing is a locally optimal approach that subdivides a given time interval into smaller intervals and schedules tasks according to completion times. Other tasks are then "packed" into any unscheduled time intervals.

This paper presents an enhanced architecture for a sensor measurement scheduler as well as an improved dynamic sensor scheduling algorithm called the On-line, Greedy, Urgency-driven, Pre-emptive Scheduling Algorithm (OGUPSA)[4].

2 Review of Scheduling Algorithm

An information management system should have some mechanism which handles resource allocation problems as well as has scheduling policies to arrange the order among the tasks (or jobs) to be executed. Most of the papers which take a theoretical approach to resource allocation and scheduling come from the Operations Research (OR), Computer Science (CS), and Electrical Engineering (EE) communities. The generalized resource allocation problem is NP-hard and it is customary to use heuristic/approximate algorithms such as simulated annealing[5], tabu search[6], stochastic probe approaches[7], genetic algorithms[8,9], and neural networks[10,11,12,13] to solve NP-hard problems. For some special cases, however, there do exist polynomial time algorithms[14]. Generally, the scheduling problem with constraints is modeled using graph theoretical approaches[7,15,16,17,18,19,20], but in some cases it also can be modeled as a Petri net[21,22,23]. In this section, selected techniques are classified according to their focus, OR, CS, or EE, although it is sometimes difficult to distinguish among them. Useful ideas can be acquired from each community.

2.1 The OR Community View

From the operations research point of view, the resource allocation problem is normally viewed as an optimization problem with constraints. It is a special case of the nonlinear (mainly integer) programming problem. OR researchers concentrate primarily on static, *i.e.*, off-line, techniques to solve job-shop and flow-shop problems including queuing theory. The typical resources considered in OR problems are manpower, machines, factory cells, *etc*. The most common objective functions utilized by the OR community are:

- minimizing makespan (schedule length)
- minimizing maximum cost
- minimizing tardiness
- minimizing the number of tardy jobs

Effective approaches such as branch-and-bound algorithm, decomposition technique, and heuristic algorithms are often used in order to solve NP-hard problems. Many well-known algorithms used for resource allocation problems can be found in a paper by Cherkassky and Zhou [12].

2.2 The CS Community View

Computer Science researchers mainly focus on dynamic (on-line) techniques. Their typical resources include:

- CPU cycles
- memory
- processors

with an objective function being based on:

- minimizing response times
- maximizing the number of arriving tasks that meet their deadline
- achieve load balance among processors

For real-time scheduling, Ramamritham and Stankovic [24] summarize the state of the real-time field in the areas of scheduling and operating system kernels. They primarily discuss four types of scheduling approaches, which are:

- static table-driven scheduling
- static priority preemptive scheduling
- dynamic planning-based scheduling
- dynamic best effort scheduling.

Generally, real-time tasks must be scheduled to meet their deadlines and to maintain high utilization of system resources. In practice, real-time tasks are categorized as periodic or aperiodic according to their arrival times. If the related system is a distributed real-time system, the task assignment will become a key problem. Usually, the objectives of task assignment are:

- load balance
- low communication costs
- high utilization of system resources
- fault tolerance

2.3 The EE Community View

The electrical engineering point of view is best expressed in terms of optimal control theory in which a process is modeled by a functional with the particular parameters chosen to minimize a performance measure. Various approaches are available for continuous and discrete event systems which are based on the degree to which the future is predicable as determined by the stochastic nature of the process and is stationarity. Optimal control of dynamic systems is concerned with the problem of determining the time history of the system's control variables which minimizes a specified performance index or cost function, subject to whatever constraints are imposed on the solution. The performance index is generally a function of the system's state variables and control variables. In order to have a well-posed problem to solve, the performance index is required to be:

- a scalar function of the state and control variables,
- a symmetric (even symmetry) function of the cost,
- a monotonically non-decreasing function of the cost, and
- a function which has a value of zero when the cost is zero.

In optimal control, one set of constraints which always has to be satisfied is the set of state equations which model the system being controlled. For a discrete-time system, they have the form:

$$x_{k+1} = f\left(x_k, u_k, k\right) \tag{1}$$

This equation gives the time response of the state based on the previous value of the state and control variables and time. If the system being controlled is linear and time-invariant, the state equations become

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k \tag{2}$$

where **A** and **B** are matrices of appropriate dimension. Other constraints which might be imposed on the solution are boundary conditions of the initial and final states and maximum values on the state and/or control variables.

The optimal control problem can then be stated as the following: determine the value of the control variable u_k for $0 < k < (N\text{-}1)$ while satisfying the system's state equations, boundary conditions, and any other constraints imposed on the system. Methods for determining the optimal control include the calculus of variations, Pontryagin's Minimum Principle, and the dynamic programming method developed by Bellman.

There are several specific optimal control approaches which can be considered including receding horizon control[25], decentralized control of discrete systems[26], and application of the principle of decomposition coordination.

By combining techniques and principles from the OR, CS, and EE fields, an on-line, dynamic scheduling algorithms can be developed.

3 The Sensor Scheduling Algorithm

First, it is important to distinguish between the functions performed by the information manager and the sensor scheduler. The information manager is concerned with intersensor issues such as:

- How accurately to measure?
- Which service to perform (*e.g.*, search, track, fire control, *etc.*)?
- From what physical location of the environment to obtain a measurement?
- When is the earliest time allowed to begin the measurement?
- What is the latest completion time allowed for the measurement?

The problem for the information manager is to determine how to maximize the effectiveness of individual sensors or a collection of sensors while simultaneously optimizing such conflicting objectives or goals as

- Detection
- Tracking
- Identification/Classification
- Emission control (EMCON)

In contrast, sensor scheduling deals with intrasensor issues (*e.g.* measurement to sensor allocation) in order to determine how to best accomplish a list of measurement tasks based on sensor availability and capabilities.

An effective implementation of this requires a hierarchical methodology based on layers, each with an appropriate, but necessarily imperfect, model of the next lower process for local optimization within that layer. The sensor scheduler then maps these measurement requests to sensors or pseudo-sensors (*e.g.* a specific mode of a multimode sensor or the cooperative use of bearings-only sensors). At each level in the hierarchy, this imperfect but workable model can be used to approximate lower level functions. An example of a general sensor manager architecture of this type is described in Denton *et al*[27].

The sensor scheduling problem is one of how to effectively assign a set of measurement tasks to a set of sensors with some constraints. These constraints may include time constraints as well as the constraint that a task may be able to be executed by only a subset of the set of all sensors. A scheduling algorithm should dynamically assign arriving tasks to these sensors in order to maximize the number of the tasks that meet their deadline. This is in addition to the need for a preemptive mechanism to insure that some important tasks will be executed promptly. Since this paper focuses only on the sensor scheduling problem for real-time systems, scheduling arriving measurement tasks from the information manager to meet their timing constraints is a major consideration. Therefore, one of the sensor scheduler's primary objectives is to maximize the number of arriving tasks that meet their deadline with no concern for how the priorities were established.

3.1 Terminology and Definitions

The sensor scheduling problem can be viewed as a method of ordering and allocating a measurement task list from the information manager to observation lists for each sensor where it is assumed that each task, T_i, has the following properties assigned to it by the information manager:

- Measurement task type *i*
- Interruptable/noninterruptable
- Priority
- Initiation time (time no earlier than)
- Deadline (complete no later than).

The following terminology will be used in our discussion.

t^i_{TNLT}	time no later than deadline for task i
t^i_{TNET}	time no earlier than constraint for initiation of task *i* (no constraint when $t^i_{TNET} = 0$)
P_i	priority of measurement task *i*
S	$\{ S_1, S_2, \ldots, S_m \}$, the set of all available sensors
SS	Subset of sensors S_i that can meet the deadline of task T_i
UI_i	urgency index (UI) for task *i*, which is a value derived from the priority and deadline of task *i*
UI function	a function f: $R^2 \rightarrow R$, which is used to map (P_i, t^i_{TNLT}) to UI_i, $i = 1, 2, \ldots, l$
INAQ	inactive queue, a queue of arriving tasks which have been prioritized but not yet scheduled
T_{INA}	$\{ T_1, T_2, T_3, \ldots \}$, all tasks in inactive queue. All tasks are sorted by their UI.
INAQHP	a working queue used by scheduler, normally receiving tasks, which have the highest urgency, from INAQ
T_{INAHP}	$\{ T_1, T_2, \ldots, T_n \}$, all tasks in INAQHP. It will be modified by scheduler dynamically.
AQ	active queue consisting of m individual queues Q_j, one associated with each sensor. They are FIFO queues.
HV	$\{ tc_1, tc_2, \ldots, tc_m \}$, horizon vector, where tc_j, $j = 1, 2, \ldots, m$, is the time when all tasks in Q_j will be completed

$SATS_i$	a set of applicable tasks in T_{INAHP} for sensor i, $i = 1, 2, \ldots, m$. It is a subset of T_{INAHP}, such that $$\bigcup_{i=1}^{m} SATS_i = T_{INAHP} \qquad \textbf{(3)}$$
$SAST_j$	a set of applicable sensors for tasks type j, $j = 1, 2, \ldots, l$. which models the constrained conditions of sensors.
Q^j_{LP}	a subset of tasks which are interruptable and have lower priorities than currently scheduling task T_k in Q_j.
AST	Applicable Sensor Table, a table which records the important relation between task type and sensor. A sample table is show in Table 1where Δt_{ij}, $i = 1, 2, \ldots, l$, $j = 1, 2, \ldots, m$, means the dwell time when a type i task uses sensor j to do a measurement. $\Delta t_{ij} = \infty$ means type i task is not applicable to sensor j,

3.2 The Dynamic Sensor Scheduler Architecture

The Sensor Scheduler (SS) is an important part of any sensor management system. A detailed architecture used to develop the OGUPSA algorithm is shown in Figure 1. The input consist of prioritized measurement requests or tasks and the output consists of rejected measurement request and sensor command.

The arriving measurement tasks first enter the inactive queue in which they are sorted by their urgency indices (UI). This is done by the INAQ Manager. The tasks with the highest UI will be moved into INAQHP by the Scheduler, if the INAQHP is not full. The task with the highest UI in the INAQHP will be dispatched into active queues (AQ) by the Scheduler according to the OGUPSA algorithm. If a measurement task can not be scheduled to meet its deadline, it is rejected and notification is passed back to the information manager. When a measurement task in the AQ is completed by a sensor, the AQ Manager will remove it from the AQ and modify the HV.

There are many ways to choose a UI function. Normally, one assumes that the priorities of tasks are positive integers. The UI function in our case is

$$f(P_i, t^i_{TNLT}) = P_i + \frac{1}{t^i_{TNLT} + C} \qquad \textbf{(4)}$$

where C is a positive constant, which is used to make the second term in the UI function above less than 1. That means that tasks with the same priorities will be sorted by their due times, that is Earliest-Deadline-First (EDF).

3.3 The OGUPSA Algorithm

The original development of OGUPSA focused on the unit execution time (UET) task scheduling problem without any task preemption. The recent enhancements to OGUPSA include the expansion and development of the AST to more realistic non-UET tasks. Logic has also been added to insure that a task requiring more than a unit execution time is not interrupted during the performance of a task. As part of the information in the sensor queue is the time to complete the task. It is assumed that the sensor is tasked until that time. Task interruption is based on the Urgency Index and is restricted to tasks currently in the active sensor queues that have scheduled but not currently being executed. If a task is preempted, it is placed in the INAQHP for rescheduling. If the interrupted task cannot be scheduled, this fact is reported back to the information manager which may pass it back to the mission manager. The tasks can be discarded (if it has become obsolete), a new deadline (modified time no later than) set, or the priority increased by either the information manager or mission manager.

Another improvement restricts the scheduling and initiation of a task by adding a "commence no sooner than a desired time" value to each requested measurement. This can be used to schedule future tasks at specific times. The final enhancement involves the use of pseudo-sensors. Two types of pseudo-sensors have been incorporated into a simulation of OGUPSA[28]. The first is a sensor that operates in several modes. An example of this is a Doppler radar operating either using Doppler or not using it. The other type of pseudo-sensor is the cooperative use of multiple bearings-only sensors at different locations in order to obtain range and bearing measurements of a target. A sensor management model and simulation developed by McIntyre and Hintz[28] demonstrates the use of the enhanced OGUPSA algorithm.

The detailed algorithm for the enhanced OGUPSA is shown below:

```
Step 1:   Move the tasks with the highest UI and $t^i_{TNET} \le$ current time in INAQ to INAQHP
          if INAQHP is not full
              Calculate or Modify $SATS_i$, $i = 1, 2, ..., m$
Step 2:   Get the task with the highest UI in INAQHP.
          Assume it is task $T_k$, $k'$ is its task type. That is,
          $$UI_k = \max_{T_i \in T_{INAQHP}} UI_i \qquad (5)$$
Step 3:   Determine whether the deadline of $T_k$ can be met.
          $$tec_i = tc_i + \Delta t_{k'i}, \quad SS = \{S_j \mid tec_j = \min_{S_i \in SAST_{k'}} tec_i\} \qquad (6)$$
          If ($tec_j > t^k_{TNLT}$) then
              Go to Step 6
```

else

Select the least versatile sensor[1] in *SS*. That is,

$$|SATS_j| = \min_{S_i \in SS} |SATS_i| \tag{7}$$

Step 4: Assign task T_k to sensor S_j and modify HV. That is,

$$\begin{aligned} Q_j &\leftarrow Q_j \cup \{T_k\} \\ tc_j &\leftarrow tc_j + \Delta t_{k'j} \end{aligned} \tag{8}$$

If S_j is a cooperative pseudo-sensor

Assign tasks to individual sensors comprising pseudo-sensor and modify the HV

If S_j is a sensor that is part of a cooperative pseudo-sensor

Assign tasks to pseudo-sensor and modify the HV

If S_j is a multimode pseudo-sensor

Assign task to actual sensor queue and set operating mode flag

Step 5: Remove T_k from T_{INAHP}. That is,

$$T_{INAHP} \leftarrow T_{INAHP} - \{T_k\} \tag{9}$$

Go to Step 1.

Step 6: The entry of the preemptive mechanism.

$$SS \leftarrow SAST_{k'} \tag{10}$$

Step 7: Select the least versatile sensor in *SS*.

$$|SATS_j| = \min_{S_i \in SS} |SATS_i| \tag{11}$$

[1] Versatility means that a single sensor is capable of performing more than one type of task and its capabilities overlap that of another sensor. If this is the case, then a specific task admits of two different sensors satisfying that task. The one which is chosen is the one which has the least ``versatility" in that it does not have as much capability overlap with the rest of the sensors so as to maintain the largest capability in the, as yet, unscheduled sensors.

Step 8: Obtain Q^j_{LP}, a subset of tasks in Q_j, which are interruptable and have lower priorities than T_k.

Step 9: Determine whether the deadline of T_k can be honored. If it preempts the tasks in Q^j_{LP}.

$$tc_{j'} = tc_j - \sum_{T_q \in Q^j_{LP}} \Delta t_{q'j} \qquad \textbf{(12)}$$

```
If $(tc'_j + \Delta t_{k'j}) <= t^k_{TNLT}$ then
    Go to Step 11.
else
    $SS \leftarrow SS - \{S_j\}$
```

Step 10: If SS is not empty, then go to Step 7.

else Reject T_k and go to Step 5.

Step 11: Assign T_k to sensor j, shift all tasks in Q^j_{LP} to INAQHP and modify HV. That is,

$$\begin{aligned} Q_j &\leftarrow (Q_j - Q^j_{LP}) \cup \{T_k\} \\ tc_j &\leftarrow tc_{j'} + \Delta t_{k'j} \\ T_{INAHP} &\leftarrow T_{INAHP} \cup Q^j_{LP} \end{aligned} \qquad \textbf{(13)}$$

```
If $S_j$ is a cooperative pseudo-sensor
    Shift all tasks of individual sensors comprising the pseudo-sensor to INAQHP and modify HV of each individual sensor
If $S_j$ is a sensor that is part of a cooperative pseudo-sensor
    Remove all task of cooperative pseudo-sensor and modify HV
Go to Step 5.
```

It is easy to see that OGUPSA is a natural and intuitive algorithm and the computational complexity has been shown in the original paper[4] to be of polynomial time dependent on the number of tasks in the T_{INAHP} and the number of sensors available with the dominating computations in Step 1 and the loop from Step 6 to Step 10. It uses the prior existing information in AST and uses the accumulated dynamic information represented by the Horizon Vector (HV). A set of sets, $SATS_i$, $i = 1, 2, ..., m$, are used as future information. In OGUPSA there are three main policies used in order during scheduling. It first uses the Most-Urgent-First (MUF) policy to select a task, then uses Earliest-Completed-First (ECF) to select applicable sensors. If there is a tie among the sensors, the third policy of Least-Versatile-First (LVF) is used to select sensor. If a tie still exists, a random selection will be used.

The concept of an HV plays a major role in the design of OGUPSA. The HV can be interpreted as a measure of the dynamic load of the AQ. The greediness of OGUPSA is reflected by the second policy of ECF. By using the LVF policy OGUPSA leaves the most options for the unscheduled tasks in INAQHP. The ECF and LVF also improve the dynamic load balance among the sensors. The preemptive mechanism in OGUPSA does not guarantee that the later arriving higher priority task will be definitely executed before the earlier arriving lower priority task when both their deadlines are honored.

If all tasks in T_{INAHP} have the same UI and $\Delta t_{ij} = 1$ or ∞, the problem will reduce to a problem called the scheduling problem with constrained processor allocation. An example consisting of five sensors and nine tasks using the AST shown in Table 2 can be shown to obtain an optimal schedule.

$SAST_j$, $j = 1, 2, ..., 9$ and $SATS_i$, $i = 1, 2, ..., 5$ can be easily calculated from the AST. The change of the HV is:

$$\begin{aligned} HV = &\ (0,0,0,0,0) \rightarrow (1,0,0,0,0) \rightarrow (1,1,0,0,0) \rightarrow (2,1,0,0,0) \\ &\rightarrow (2,2,0,0,0) \rightarrow (2,2,1,0,0) \rightarrow (2,2,1,1,0) \rightarrow (2,2,1,1,1) \\ &\rightarrow (2,2,1,1,2) \rightarrow (2,3,1,1,2). \end{aligned}$$

The final result produced by OGUPSA is:

$Q_1 = \{ T_1, T_3 \}$,

$Q_2 = \{ T_2, T_4, T_9 \}$,

$Q_3 = \{ T_5 \}$,

$Q_4 = \{ T_6 \}$,

$Q_5 = \{ T_7, T_8 \}$

which is the same as the proved optimal result produced by Chang and Lee[29] with a minimum makespan equal to 3. McIntyre and Hintz[28] demonstrate the use of a reasonable non-UET AST for sensor scheduling that includes the use of multimode sensor and the cooperative use of multiple bearings-only sensors in a sensor management simulation.

4 Conclusion

An enhanced scheduling algorithm called OGUPSA is presented based on several scheduling schemes found in the OR, CS, and EE literature. OGUPSA provides a preemptive mechanism and uses a suitable urgency index function to meet the requirement of the most urgent task first. This algorithm can also detect the deadline failure of the tasks. It is able to generate an optimal schedule in the sense of a minimum makespan for a group of tasks with the same priorities by introducing the concept of a Horizon Vector. Moreover, it seems to lead to dynamic load balance among all of the sensors by using the policy of selecting the least versatile sensor first. It has been refined to accommodate multimode sensors and the cooperative use of multiple bearings-only sensors. Finally, from a computational standpoint, it is a polynomial time algorithm.

Table 1 **Applicable Sensor Table**

Sensor Task type	S_1	S_2	...	S_m
1	Δt_{11}	Δt_{12}	...	Δt_{1m}
2	Δt_{21}	Δt_{22}	...	Δt_{2m}
...	...	...	...	...
l	Δt_{l1}	Δt_{l2}	...	Δt_{lm}

Table 2 **The AST for the UET scheduling problem**

Sensor Task type	S_1	S_2	S_3	S_4	S_5
T_1	1	1	∞	∞	∞
T_2	1	1	∞	∞	∞
T_3	1	1	∞	∞	∞
T_4	1	1	∞	∞	∞
T_5	∞	∞	1	1	1
T_6	∞	∞	∞	1	1
T_7	∞	1	∞	∞	1
T_8	∞	1	∞	∞	1
T_9	∞	1	∞	∞	1

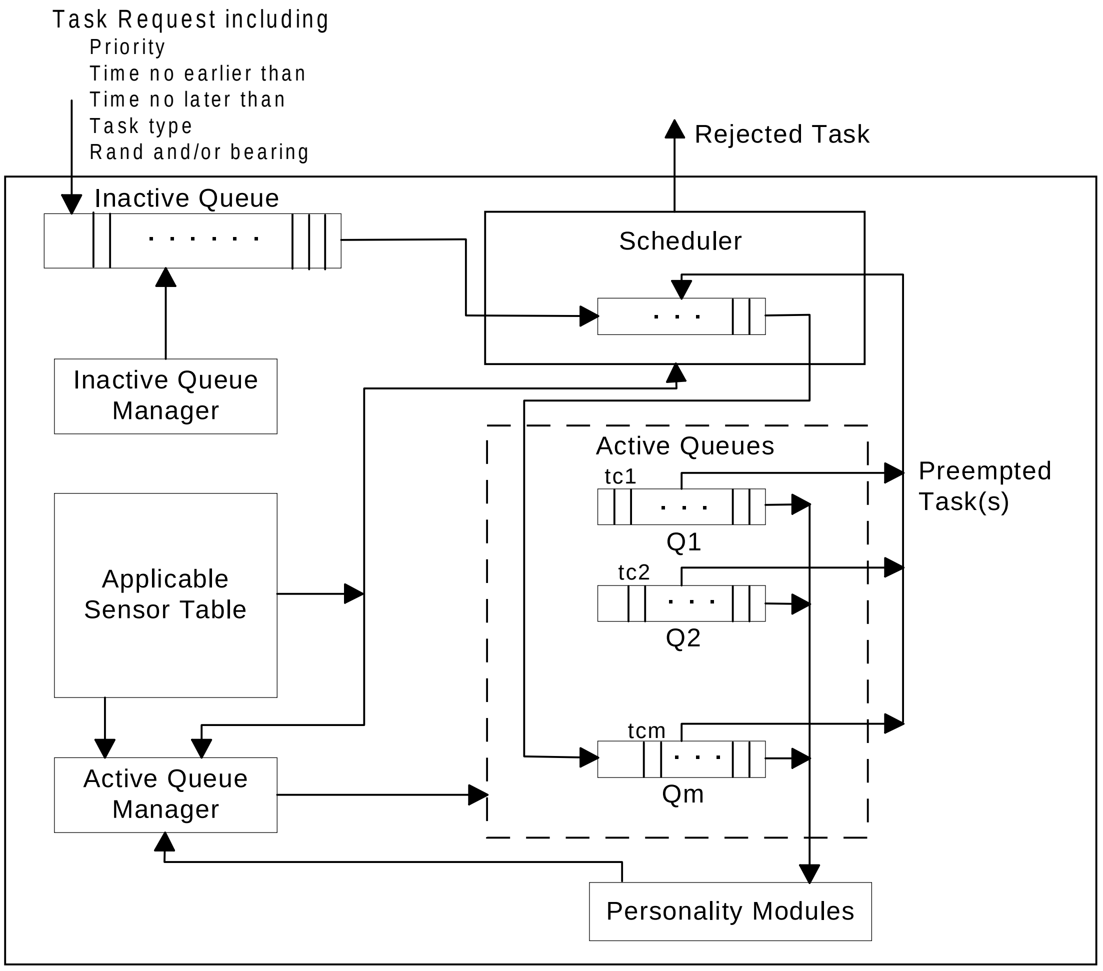

Figure 1 Enhanced OGUPSA architecture

BIOGRAPHIES

Gregory A. McIntyre received his B.A. degree in Math and Computer Science from Washburn University of Topkea in 1981 and his M.S. in Operations Research in from the Air Force Institute of Technology (AFIT) in 1986. He was a distinguished graduated of ROTC and received his commission in the Air Force in 1981 and is currently a Major in the Air Force. His assignments have taken him to Utah, Ohio and Washington D.C. While in Utah he performed real-time data analysis and computer support for several cruise missile and remotely piloted vehicle programs. Following Utah, he attended AFIT in Ohio where he was a distinguished graduate. For the last 10 years, he has been in several assignments in the Washington D.C. area as scientific analyst and budget analyst. He is currently on an Air Force fellowship and is a doctoral student in Information Technology at George Mason University.

Kenneth J. Hintz received his B.S. degree in Electrical Engineering from Purdue University in 1967 and his M.S. and Ph.D. in Electrical Engineering from the University of Virginia in 1979 and 1981 respectively. Since 1987 he has been an Associate Professor in the Department of Electrical and Computer Engineering at George Mason University. Before joining GMU, he was with the Naval Surface Warfare Center, Dahlgren, VA, working in electronic warfare and radar signal processing where he designed and built the AN/ULQ-16. He holds two patents and is a Senior Member of IEEE and a member of SPIE. His current research interests are in X-ray, thermal, and visual image processing, computer architectures and algorithms for real-time signal processing, and self-organizing machines including the application of genetic algorithms to the design of neural networks.

Figure 1 Enhanced OGUPSA architecture

Table 1 Applicable Sensor Table

Table 2 The AST for the UET scheduling problem

References

1. J. M. Manyika and H. F. Durrant-Whyte. *Data Fusion and Sensor Management : A Decentralized Information-Theoretic Approach*, Ellis Horwood: New York; (1994).
2. R. A. Adrian, “Sensor Management,” *Proceedings of the 1993 AIAA/IEEE Digital Avionics System Conference*, IEEE: New York, NY, 32-7 (1993).
3. R. Popoli, “The Sensor Management Imperative”, Chap. 10 in *Multitarget-multisensor tracking : applications and advances*, vol. 2, Y. Bar-Shalom, Ed., pp. 325-392, Artech House, Norwood, MA (1992).
4. Z. Zhang and K. J. Hintz, “OGUPSA Sensor Scheduling Architecture and Algorithm,” *Signal Processing, Sensor Fusion, and Target Recognition V. Proceedings of the SPIE - The International Society for Optical Engineering*, **2755**, SPIE, 296-303 (1996). K. J. Hintz and Z. Zhang, “Dynamic Resource Allocation and Sensor Management for OBATS,” Final Report, George Mason University, (1994).
5. D. E. Jeffcoat and R. L. Bulfin, "Simulated annealing for resource-constrained scheduling," *European Journal of Operational Research*, **70(**1), 43-51 (1993).
6. E. L. Mooney and R. L. Rardin, "Tabu search for a class of scheduling problems," *Annals of Operations Research*, **41**(1-4), 253-78 (1993).
7. L. Tao, B. Narahari and Y. C. Zhao, "Assigning task modules to processors in a distributed system," *Journal of Combinatorial Mathematics and Combinatorial Computing*, **14**, 97-135, Oct. 1993.
8. L. J. M. Cluitmans, "Using genetic algorithms for scheduling data flow graphs," Report No: EUT 92-E-266, Issued by Eindhoven Univ. Technol., Netherlands, (1992).

9. T. J. Starkweather, "Optimization of sequencing problems using genetic algorithms (scheduling)," PhD dissertation, 1993, Colorado State University.

10. R. Thawonmas, N. Shiratori and S. Noguchi, "A real-time scheduler using neural networks for scheduling independent and non-preemptable tasks with deadlines and resource requirements," *IEICE Transactions on Information and Systems*, **E76-D(**8), 947-55 (1993).

11. L. C. Gonzalez, J. C. Sutton and R. M. Felder, "The use of neural networks to solve resource-constrained process scheduling problems," *Proceedings. Fourth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-91)*, **1**, 338-44 (1991).

12. V. Cherkassky and D. N. Zhou, "Comparison of conventional and neural network heuristics for job shop scheduling," *Proceedings of the SPIE - The International Society for Optical Engineering*, **1710**, pt.1, 815-25, vol.2 (1992).

13. S. Vaithyanathan and J. P. Ignizio, "A stochastic neural network for resource constrained scheduling," *Computers & Operations Research*, **19(**3-4), 241-54 (1992).

14. T. Ibaraki and N. Katoh, *Resource allocation problems: algorithmic approaches*. The MIT press (1988).

15. R. Jain, J. Werth, J. C. Browne and G. Sasaki, "A graph-theoretic model for the scheduling problem and its application to simultaneous resource scheduling," *Computer Science and Operations Research. New Developments in their interfaces*, 321-48 (1992).

16. H. H. Ali and H. El-Rewini, "Task allocation in distributed systems: a split graph model," *Journal of Combinatorial Mathematics and Combinatorial Computing*, **14**, 15-32 (1993).

17. Chiun-Chieh Hsu and Pao-Jyh Lee, "Distributed task assignment using critical path

estimate," *Information Sciences*, **74(**3), 191-212 (1993).

18. D. Rafaeli, D. Mahalel and J. N. Prashker, "Heuristic approach to task scheduling: 'weight' and 'improve' algorithms," *International Journal of Production Economics*, **29(**2), 175-86 (1993).
19. H. Oudghiri and B. Kaminska, "Global weighted scheduling and allocation algorithms," *International Journal of Production Economics*, **29(**2), 491-5 (1993).
20. N. C. Audsley *et al*., "The DrTee architecture for distributed hard real-time systems," *Distributed Computer Control Systems 1991. Towards Distributed Real-Time Systems with Predictable Timing Properties. Proceedings of the 10th IFAC Workshop*, 49-53 (1991).
21. Q. Chen and J. Y. S. Luh, "Operational scheduling using truncated Petri net technique," *Proceedings IEEE International Workshop on Emerging Technologies and Factory Automation - Technology for the Intelligent Factory*, 230-5 (1992).
22. J. -M. Gourgand and S. Norre, "Petri net based methodology for task scheduling on multiprocessor architectures," *Simulation*, **61**(3), 185-92, (1993).
23. S. Balaji *et al*., "S-nets: a Petri net based model for performance evaluation of real-time scheduling algorithms," *Journal of Parallel and Distributed Computing*, **15(**3), 225-37 (1992).
24. K. Ramamritham and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems," *Proceedings of the IEEE*, **82(**1), 55-67 (1994).
25. D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Transactions on Automatic Control*, **35(**7), 814-824, (1990).
26. F. Lin and W. M. Wonham, "Decentralized control and coordination of discrete-event

systems with partial observation," *IEEE Transactions on Automatic Control*, **35(**12), 1330-1337, (1990).

27. R. V. Denton, E. I. Alcaraz, J. P. Knopow, J. Llinas and K. J. Hintz, "Towards modern sensor management systems," *DFS-93*, **1**, 659-78 (1993).

28 G. A. McIntyre and K. J. Hintz, “Sensor Management Simulation and Comparative Study,” *Proceedings of the SPIE - The International Society for Optical Engineering*, **3068**, (1996).

29 R. S. Chang and R. C. T. Lee, “On a scheduling problem where a job can be executed only by a limited number of processors,” *Computers and Operations Research* **15**(5), 471-8 (1988).