Performance Optimization in Signal Processing Systems
Amit Misra, Ben Abbott, and Janos Sztipanovits
Department of Electrical Engineering
Vanderbilt University
Nashville, TN 37235

Abstract
This paper discusses techniques to optimize the utilization of underlying computer resources with respect to a dynamic signal processing system executing on them. An example system is used to illustrate these techniques: a structurally adaptive solution to the sonar problem, direction of arrival. The signal processing system runs in a distributed, parallel environment, the Multigraph execution environment. Above the executing real-time signal processing system, a controller guides and coordinates overall goals (e.g. tracking). It also manages the available system resources taking into account the memory and time requirements of the signal processing algorithms and their priorities. Further, a user interface allows priorities and various operating parameters of the system to be changed dynamically.

Introduction
In recent years, considerable amount of research has been focused on Adaptive Signal Processing and parallel, distributed Signal Processing with higher level control. Typically, these signal processing applications have to perform a variety of tasks in real-time while operating with limited processing resources (e.g. processing speed and available memory). To maintain an acceptable real-time performance level, these tasks need to be allocated the limited processing resources according to the importance of the task, the time constraints and other resource constraints. Thus, an application needs to pay attention to these performance optimization issues along with pure signal processing issues. The problem is further compounded by the fact that the priorities of these tasks are more than likely to change with time. Thus, there is a need to do the resource allocation dynamically, and yet maintain signal flow consistency. This implies that the structure of the system will change during run time according to some performance criteria and resource constraints.

A structurally adaptive system is one which is able to change its processing structure during run time [1]. Most of the work done in the field of adaptive systems has been focused on devising adaptive signal processing algorithms that change the operating parameters of the systems. Structurally adaptive systems have not received as much attention. But, in complex, unstable, changing environments, parametric adaptation can be insufficient. The structure of the system itself may need to be changed. In recent years, some attempts have been made to provide environments to implement structurally adaptive systems running in a distributed processing environment. At Vanderbilt University, one such environment, namely the Multigraph Programming and Execution Environment has been developed which provides a solution to the problems usually encountered during the development of such systems [2]. The Multigraph Environment provides a parallel, distributed execution environment using a graph computational model, hierarchical modeling tools with graphical editing, allows one to combine numeric and symbolic processing, and provides an interface to the underlying computational structure so that the user application may work with the abstract model of the signal processing system.

Direction of Arrival
The direction-of-arrival (DOA) problem has its use in many applications: SONAR, RADAR, Imaging, Biomedical, Astrophysical Exploration [3]. The basic problem in DOA applications is to find the directions, from which incident spatially propagating sources impinging on the sensing devices may be arriving. The user is usually interested in the location of the sources emitting these signals in terms of their azimuth and elevation angles, and sometimes their range. For this example, two DOA techniques were implemented: Beam-
forming by way of a Spatial Transform and MUSIC estimate.

Beamforming is perhaps the most widely used technique for finding the DOA. The term beamforming derives from the fact that early spatial filters were designed to form a pencil beam in order to receive a signal radiating from a specific location and attenuate signals from other locations [3]. A beamformer uses an array of sensors to sample the arriving signals and typically, linearly combines the spatially sampled time series from each sensor in much the same manner that an FIR filter linearly combines temporally sampled data.

Figure 1 shows the signal flow graph used for computing the spatial transform of the signals impinging on an array of sensors. The input to the system is a $p \times N$ sensor signal matrix $X(t)$, where $p$ is the number of sensors and $N$ is the number of samples. The sensor locations are described by the $p \times 3$ matrix $Z$. The incident waves are assumed to have the same speed and frequency. The search range consists of the azimuth ($\theta$) and elevation ($\phi$) angles which the system examines to determine the DOA of the signals. The search range is such that

$$\theta_0 < \theta_1 < \theta_2 \ldots < \theta_{n-1}$$

and

$$\phi_0 < \phi_1 < \phi_2 \ldots < \phi_{m-1}$$

The value of $n$ is given by

$$n = (\theta_{n-1} - \theta_0)/\text{res}_\theta$$

where $\text{res}_\theta$ is the resolution in azimuth angles with which the search is performed. Similarly $m$ is given by

$$m = (\phi_{m-1} - \phi_0)/\text{res}_\phi$$

The system computes the energy estimate $\hat{B}(\theta_i, \phi_j), 0 \leq i \leq n - 1, 0 \leq j \leq m - 1$ for all the combinations of azimuth and elevation angles. We will refer to one combination of azimuth and elevation angle as one direction.

The functions of the various modules shown in the signal flow graph are: Pos-matrix supplies a $p \times 3$ sensor position matrix $Z$. Array-corr computes an array correlation matrix

$$\hat{R}_z = \frac{1}{N}[X(t)X(t)^*]$$

Ko-vector computes a direction vector

$$\hat{e}_{\theta, \phi_i}$$

Sko-vector computes a steering vector

$$s(\theta_0, \phi_i) = e^{j\phi_i \hat{e}_{\theta_0, \phi_i}}$$

Compute-energy computes $\hat{B}(\theta_i, \phi_j)$ from the correlation matrix and steering vector. Collect-energy collects the computed energies in different directions, combines them into one matrix and gives out the result.

The MUSIC method for estimating DOA of signals uses an eigenspace approach. It is based on the Multi-pIe Signal Characterization (MUSIC) algorithm [4] [5] [6]. MUSIC is a high resolution technique which can be used to distinguish closely placed sources. It performs fairly well under low SNR conditions and works well with incoherent sources. In the example it is assumed that the sources are incoherent.

The MUSIC signal flow graph resembles that of the spatial transform. The only differences are that the compute-energy module is replaced by a MUSIC-estimate module and that array-corr is followed by a noise-vectors module. The noise-vectors module finds out the noise subspace eigenvectors of $\hat{R}_z$. The MUSIC-estimate module computes the MUSIC estimate $\hat{E}(\theta_i, \phi_j)$.

See [3] for a formal analysis and discussion of the signal processing techniques involved with Spatial transform and MUSIC.

Spatial Transform vs. Music

In a SONAR application, tradeoffs between the two approaches cause the optimum choice of Spatial trans-
form or MUSIC to depend on both the current tracking situation and computer resources. The issues include:

- Spatial transform is computationally faster than MUSIC estimate.
- Spatial transform is a low resolution technique. Increasing resolution below 2 degrees barely increases the accuracy.
- The spatial transform gives some false peaks in the spectrum.
- MUSIC estimate is a high resolution technique, so much so that if we make the search range 3 degrees or greater, it misses the peaks in the spectrum more often than not.
- MUSIC estimate rarely gives false peaks.

In our case, a scenario was imagined in which a ship at sea is looking for sources. The basic search strategy was to use the spatial transform to search in a larger area with low resolution, and consider the peaks returned as locations of possible sources. MUSIC is used to search in a small area around these possible sources. Since MUSIC rarely gives false peaks, using MUSIC around possible sources will either confirm or reject these sources. The confirmed sources are called detected sources. Furthermore, MUSIC gives a better fix on the location of the sources. The search resolution for spatial transform is 3.0'. The MUSIC estimate algorithm is used to search with a resolution of 2.0' in an area ±8.0' azimuth angle and ±8.0' elevation angle around the possible sources. Apart from this the user can select a detected source for focusing. MUSIC is used to search around the focus sources with a resolution of 0.5' and search range of ±4.0'. This gives an even more accurate location of the focus sources. Furthermore, the area around the focus sources is searched continually to track them.

Note that the search resolutions and search ranges given above are the default values. The user can change all of them on the fly. Also, the search strategy described in this section is not the only one possible. There can be many search strategies depending on the application. What is of note here is that the search strategy can be changed quite easily without making any changes in the rest of the system. How this is possible and how it can be done will become clearer in the later sections.

The Multigraph Architecture

This example was developed within the Multigraph Architecture (MA). Multigraph provides symbolic and numerical programming techniques, different models of parallel computations, and various programming paradigms.

MA is a layered architecture composed of a knowledge-based layer, module layer, system layer and hardware layer. The hardware layer consists of the computer hardware system, upon which the whole structure is built. The system layer includes an operating system providing standardized access mechanisms to the hardware resources. The module layer is an intermediate layer between the knowledge-based and system layers. Its primary function is to provide a virtual machine for the Multigraph Computational Model (MCM). MCM is a parallel, graph model of computation which supports medium or large computational granularity. The module layer includes a run-time system for MCM, called the Multigraph Kernel which supports the dynamic configuration and control of computational graphs and provides a data/demand-flow scheduler which allocates the processing resources. The knowledge-based layer supports symbolic computations. A general programming methodology has been developed for this layer: declarative languages are used for the very high-level, structural models of systems. The declarations are mapped into an appropriate configuration in the Multigraph execution environment.
System Architecture

In the case of this example, the primitive components of the signal flow graph (e.g., `collect-energy`) were implemented in C. The structures of Spatial Transform and MUSIC were modeled by way of a graphical editor [7]. The top level controller was implemented in a concurrent object oriented language, ACO. It adjusted model parameters and invoked HDL [8] to create instances of the Spatial Transform and MUSIC from their declarations as needed. HDL interfaces with the Multigraph kernel and builds the necessary, real-time, computation graph from the model. Figure 2 shows a screen dump of one layer of the model hierarchy of the Spatial Transform model.

The overall structure of the system is shown in Figure 3. It consists of many different objects which are:

- **Actual-sources**, **Possible-sources**, **Detected-sources**, **Focus-sources**: These objects keep a list of actual sources used for simulating sensor signals, possible sources returned by spatial-module and detected sources returned by music-module. These objects have methods to add and delete a source, give all source positions, to return a source which is in a particular region, etc.

- **Spatial-module** and **Music-module**: These objects interface to the HDL layer and handle one instance of spatial transform and music modules each, respectively. They have methods to change the parameters of the modules including optimum size and search range, to create, destroy and run the modules. Note that if the optimum size or the search range changes, the HDL may change the structure of the underlying graph computation.

- **Focus-module**: This object is basically the same as music-module except that it is associated with the focus searches.

- **Spatial-manager**, **Music-manager**, **Focus-manager**: These objects manage a set of spatial-module's and music-module's respectively. They are told by the controller to perform a search in a particularly area with a particular resolution, the number of modules that can exist together and the number of these modules that can run simultaneously (due to memory and processing resources). It adds or deletes its search objects, asks them to set their parameters and then asks them to perform the searches. The search results are passed to the controller.

- **Simul**: This object provides an interface between the controller and HDL. It allows the controller to build and modify the simulation environment (signal sources).

- **Scenario**: This object interfaces with the HDL to provide a display of the sources on the screen and allow a mouse driven interface to the user. When this object is asked to draw the scenario picture, it asks actual-sources, possible-sources and detected-sources to give all the source positions and then uses the positions to draw a picture. If the user wants to select a source for focus or if he wants to know the location of a source in the picture, scenario provides a mouse driven interface to do so.

- **Controller**: This is the top level object which controls and coordinates the searches. It has three parts:
  1. **Search controller**: This part deals with running a search, looking at the results of a search, adding or deleting sources from one of the source lists, taking care of focus searches, displaying the scenario etc.
  2. **Resource manager**: This part deals with managing the number of copies of spatial transform and music computation graphs that exist and run in parallel. It looks at the searches that are waiting, the amount of memory these searches will require to run, the amount of memory that the HDL modules that perform these searches will take up, the amount of memory available and the priority of the searches. It gives the highest priority to focus searches, next to music searches and lowest to spatial transform searches. Then, using simple if-then-else rules, it decides the number of modules of each kind that can exist and the number that can...
run in parallel.

3. User interface.

Implementing the controller in an object-oriented language gives us some advantages which are typical of programming in object-oriented languages. These advantages include hierarchical breakdown and clean separation of the functionalities of different objects which allows us to easily make changes in the system.

There are default values for all the parameters that the controller needs to run the searches, but they can also be changed dynamically. This will result in a change in the functioning and perhaps in the structure of the system. The changes are introduced through the user interface which allows the user to

- Select a source: the user may want to select a source to delete it, to look at its attributes or to start or stop focusing on the source. The system allows the user to select a source by pointing to it in the scenario picture using a mouse. Then it takes the appropriate action, i.e., delete, display attributes, focus or unfocus.
- Change the system parameters like the available memory, processing speed and communications overhead. These parameters can also be set by another program.
- Change the search resolutions and search ranges for the three kinds of searches.
- Start and stop the general search, i.e., searches using spatial transform.
- Search all sectors using spatial transform or focus on one sector only.

Note that the system can have more than one search manager for searches with spatial transform, with MUSIC and focus searches. All that is needed is to add the corresponding search manager. If the operating environment consists of distributed processors, then the signal processing algorithms will run in parallel on distributed processors, courtesy of the Multigraph Kernel. What is more, since the ACO objects also run on Multigraph, the controller can also be a distributed controller.

**Conclusion**

The direction of arrival finding problem was implemented under the Multigraph Architecture. It demonstrated:

- There is a need for optimizing the real-time performance of large signal processing systems.
- Performance optimization can be done by allocating resources according to the tasks that need to be performed, their relative priorities and the computing resources that they use.
- Structural adaptivity is needed to optimize the performance. Traditional adaptivity, which only allows changes in the operating parameters of the system, does not suffice.

Furthermore, it was demonstrated that Multigraph provides the tools to build large, structurally adaptive, signal processing systems.

**References**


