Task Decomposition

James S. Albus
Robot Systems Division
National Institute of Standards and Technology
Gaithersburg, Maryland 20899 USA

Abstract

A hierarchical reference model architecture for real-time intelligent control has been defined. At each level generic task decomposition modules accept commands and use stored task knowledge to decompose the commands into subcommands to be sent to subordinate task decomposition modules. Both spatial and temporal task decomposition are addressed.

Introduction

A task is an activity performed by one or more agents on one or more objects in order to achieve a goal. Task decomposition is the process that the agent(s) performs in order to achieve a goal. Task decomposition consists of three elements: spatial decomposition, temporal decomposition, and execution.

1) Spatial decomposition consists of the assignment of agents to jobs, and the allocation of resources to agents, for the duration of the task.
2) Temporal decomposition consists of each agent planning a sequence of subtasks to accomplish its respective job assignment.
3) Subtask execution consists of each agent implementing its respective plans.

A reference model architecture for real-time intelligent control has been defined [1,2,3]. The architecture is hierarchically layered, with computational nodes consisting of sensory processing, world modeling, behavior generating, and value judgment modules at each level. The hierarchical layers are defined by:

1) temporal and spatial decomposition of tasks into levels of detail,
2) spatial and temporal integration of sensory data into levels of abstraction, and
3) spatial resolution of objects and maps, and temporal resolution of events and trajectories in the representation of knowledge in the world model.

Temporal resolution is defined by sampling rate, state-change intervals, and loop bandwidth. Spatial resolution is defined by the size of pixels in visual and tactile receptor arrays, the signal-to-noise ratio in analog sensors, and the number of resolution elements in analog-to-digital converters.

Temporal integration is defined by length of historical memory traces and planning horizons. Spatial integration is defined by clustering of pixels into regions, features into objects, and objects into groups.

In the reference model architecture, task decomposition is performed by task decomposition modules, each of which consists of:

1) a Job Assignment submodule that performs spatial decomposition,
2) a set of Planners that generate plans for the agents, and
3) a set of Executors that execute the plans for the agents.

Figure 1 is a diagram of a generic task decomposition module at level(i). The Executor submodule from level(i+1) computes a task command and a probable next task command based on its inputs, which are the current step in its plan plus the current feedback from its view of the world model. The world model feedback informs the Executor of the current best estimate of the state of the world, including the state of the control system itself.

The Job Assignment (JA) submodule at level(i) performs a spatial decomposition resulting in a job assignment wherein each agent is assigned a job and the resources required for that job. The job assignment operation is a process, and the job assignment itself is a data structure.

The job assignment data structure for the current and probable next task, together with feedback from the world model forms the input to the agent Planners. Feedback from the world model allows the planners to perform real-time planning. If there is no feedback from the world model to the Planners, then the planning process can be done anytime prior to task execution. In the case of off-line planning, there is no requirement for real-time feedback from the world model.

Figure 1. A generic task decomposition level. Rectangular boxes represent processes. Circles and ovals represent data structures. Arrows indicate the flow of data or control.

In Figure 1, the Planners are shown as a single process, rather than as a set of individual processes, one for each agent. This is because there are often mutual constraints, or requirements for coordination, between plans for the set of agents assigned to a task. For example, the agent may need to time-share their assigned resources, or they may need to synchronize their planned sequence of actions in order to accomplish the goal of their mutual task. This implies communication between planning processes for each of the agents that is indicated in the diagram by a single Planners process box.

U.S. Government work not protected by U.S. copyright 49
The activity of the Planners is a process. The resulting plan, or set of coordinated plans for each of the agents, is a data structure. Most plans can be represented by a state-graph. Coordinated plans for multiple agents can be represented by cross-coupled state-graphs, in which the edges (or conditions for state transition) in the state-graph of one agent are defined by states or edges in the state-graphs of other agents.

For each agent at level(i), there is an Executor that computes a task command and a probable next task command for level(i-1) based on its inputs. The inputs of each Executor submodule consist of the current state in its plan state-graph, plus the current feedback from its view of the world model.

When the world model indicates to an Executor that a subtask in its current plan is successfully completed, the executor steps to the next subtask in that plan. When all the subtasks in the current plan are successfully executed, the executor steps to the first subtask in the next plan. If the feedback indicates the failure of a planned subtask, the executor branches immediately to a preplanned emergency subtask. Its planner simultaneously begins work selecting or generating an error recovery sequence which can be substituted for the former plan which failed.

Output subcommands produced by Executors at level(i) become input commands to Job Assignment submodules in TD modules at level(i-1). Planners constantly operate in the future, each generating a plan to the end of its planning horizon. The Executors always operate in the present, at time t=0, constantly monitoring the current state of the world reported by feedback from the world model. At each level, each Executor submodule closes a reflex arc, or servo loop, and the Executor submodules at the various hierarchical levels form a set of nested servo loops. The Executor loop bandwidth decreases about an order of magnitude at each higher level. Each hierarchical level has a typical frequency of execution that is determined by the dynamical properties of the system being controlled.

**Task Knowledge**

Fundamental to task decomposition is the representation and use of task knowledge. A task is a piece of work to be done, or an activity to be performed. For any TD module, there exists a set of tasks that the TD module knows how to do. Each task in this set can be assigned a name. The task vocabulary is the set of task names assigned to the set of tasks each TD module is capable of performing.

Knowledge of how to perform a task may be represented in a task frame data structure. An example task frame is shown in Figure 2. The name of the task is a string that identifies the type of activity to be performed. The goal may be a vector that defines an attractor value, set point, or desired state to be achieved by the task. The goal may also be a map, graph, or geometric data structure that defines a desired "to-be" condition of an object, or arrangement of components. The object is an identifier that points to a database that may describe the geometry, position, orientation, surface characteristics, material composition, physical properties, and class to which the object belongs.

Parameters are properties of the task. The requirements define the information required from the world model during the task. This may consist of a list of state variables, maps, and/or geometrical data structures that convey actual, or "as-is" conditions that currently exist in the world. Requirements may also include resources, tools, materials, time, and conditions needed for performing the task.

The procedure section contains either a set of pre-computed plans or scripts for decomposing the task, or one or more planning algorithms for generating a plan, or both. For example, the procedure section may contain a set of IF/THEN rules that select a plan appropriate to the "as-is" conditions of the world model. Alternatively, the procedure section may contain a planning algorithm that computes the difference between "to-be" and "as-is" conditions. This difference may then be treated as an error that the task planner attempts to reduce, or null through "Means/Ends Analysis" or A* search. Each subsystem planner would then develop a sequence of subtasks designed to minimize its subsystem error over an interval from the present to its planning horizon. In either case, the Executor would act as a feedback controller, attempting to serve its respective subsystem to follow its plan. The procedure section also contains emergency procedures that can be executed immediately upon the detection of a disabling condition.

![Figure 2. An example of a task frame.](image-url)
Summary and Conclusions

Task decomposition is central to intelligent control. A reference model architecture that partitions the real-time control problem into hierarchical layers has been defined. In this architecture, task decomposition modules at every level decompose higher level tasks into coordinated concurrent sequences of lower level tasks to be performed by one or more agents on one or more objects.

Robotic systems based on this architecture have been implemented for a wide variety of applications that include loading and unloading of parts and tools in machine tools, controlling machining workstations, performing robotic deburring and chamfering, and controlling space station telerobots, multiple autonomous underwater vehicles, unmanned land vehicles, coal mining automation systems, postal service mail handling systems, and submarine operational automation systems. A methodology for designing open-architecture real-time intelligent control systems based on this reference model is currently being developed by the Robot Systems Division at NIST.

References


