Behavior Research Methods
2010, 42 (3), 877-883
doi:10.3758/BRM.42.3.877

SANLab-CM: A tool for incorporating stochastic operations into activity network modeling

Evan W. Patton and Wayne D. Gray
Rensselaer Polytechnic Institute, Troy, New York

The Stochastic Activity Network Laboratory for Cognitive Modeling (SANLab-CM) is a new tool that incorporates stochastic operations into activity network modeling (Schweickert, Fisher, & Proctor, 2003). In this article, we discuss the core functionality of SANLab-CM and walk through a case study that expands a previously published single, static path model of telephone operators interacting with customers via a workstation (from Gray, John, & Atwood, 1993) into a stochastic model that generates 55 unique paths with different frequencies and a variety of qualitative properties. Without SANLab-CM, it would have been easy to mistake some of the more frequent critical paths as evidence for alternative strategies for task completion. With SANLab-CM, these critical paths can be shown to be simple emergent properties of variability in elementary cognitive, perceptual, and motor processes.

The Stochastic Activity Network Laboratory for Cognitive Modeling (SANLab-CM) is a tool for designing activity networks for routine, interactive behavior where activities have times assigned stochastically on the basis of parameters set by the modeler. Activity networks are directed acyclic graphs in which each node represents some process in time, and connections between nodes represent execution order; they were first applied to modeling cognition by Schweickert (1978). Providing stochastic mechanisms extends the activity network approach and results in various critical paths, paths of activities that explain the entire task execution time, through a single network. Varied execution times and the resulting set of critical paths have the potential to advance many research issues, including implicit versus explicit strategies (e.g., Cary & Carlson, 2001), scheduling processes in working memory (Ehrenstein, Schweickert, Choi, & Proctor, 1997), the influence of external factors on the overall time to do a mental task (Dzhafarov, Schweickert, & Sung, 2004), suboptimal yet stable performance (Fu & Gray, 2004), and the interleaving of interactive routines (Gray & Boehm-Davis, 2000; Gray, Sims, Fu, & Schoelles, 2006). SANLab-CM lowers the barrier to this type of cognitive modeling by providing a simple graphical interface to construct complex stochastic models of interactive behavior.

A Brief History of Activity Network Modeling in Cognitive Science

Schweickert's (1978, 1980) generalization of the additive factor method (Sternberg, 1969a, 1969b) is the earliest use we know of activity networks in modern cognitive science. Fisher and Goldstein (1983) further generalized Schweickert's method to independent random variables drawn from exponential distributions, and later generalized to interruptible systems involving a wider range of distributions (e.g., exponential, gamma; Goldstein & Fisher, 1991). A decade after Schweickert's (1978, 1980) generalization, John (1988, 1990, 1996) adapted the activity network representation as a control structure for the Model Human Processor theory of Card, Moran, and Newell (1983) that facilitated modeling the control and interleaving of cognitive, perceptual, and motor operations. This notation, CPM-GOMS,[1] was successfully used by Gray, John, and Atwood (1993) to develop cognitive engineering models that predicted real-time performance of telephone operators on new workstations. The models were used by the phone company to save approximately $100 million in 1989.

The tools for developing such models have been limited to mathematical notations (Schweickert, 1978), spreadsheets (Schweickert, Fisher, & Proctor, 2003), or business productivity software such as MacProject (Gray & Boehm-Davis, 2000; Gray et al., 1993). Such tools demand a high level of sophistication on the part of their users, provide an extremely awkward and error-prone environment in which to develop any but the simplest of models, and/or were developed with a specific application in mind (simple scheduling for production or business) that works against the needs of the cognitive modeling community.[2] Most computer-based approaches focus on time scales of minutes and hours without stochastic manipulations, whereas cognitive modelers are often interested in phenomena that occur on the order of tenths of a second, where individual execution times might vary greatly. The one attempt that we know of to develop a higher level authoring tool for such modeling had some important proofs of concept (John,

E. W. Patton, pattoe@rpi.edu

© 2010 The Psychonomic Society, Inc.

Vera, Matessa, Freed, & Remington, 2002; Vera, John, Remington, Matessa, & Freed, 2005) but was focused at a specific application (developing CPM-GOMS models) and seems to have had no lasting influence. In contrast, SANLab-CM attempts to provide a general-purpose tool for cognitive modeling of any type, in a graphical user interface developed for modelers, with a core suite of tools that provide immediate overviews of the model and the results of modeling that support the need of cognitive modelers. SANLab-CM makes no assumptions about a particular cognitive paradigm (e.g., serial vs. parallel cognition), allowing modelers to develop models in the paradigm of their choosing. In addition, SANLab-CM provides a key stochastic component that had been underutilized in prior uses of activity networks in cognitive science.

SANLab-CM: A New Approach to Activity Network Modeling

SANLab-CM provides tools to investigate how variability in the operations used in activity network models affect task performance. Millisecond-level variations in execution time can lead to different behavior patterns, given the same task and cognitive workload (Gray & Boehm-Davis, 2000; Schweickert et al., 2003). In the following sections, we will discuss each of the components of SANLab-CM and how they help achieve this goal.

Activity network representation. SANLab-CM represents the interactions within and between agents and the external world as a network of elementary cognitive, perceptual, and motor operations with durations ranging from approximately 30 to 300 msec. These elementary operations can be composed to form a series of interleaved atomic actions (referred to in SANLab-CM as *interactive routines*; see Gray et al., 2006) with durations of approximately 300 msec to 3 sec. Example interactive routines include pressing a key on the keyboard and moving the eyes from one point in space to another. Interactive routines can be further composed into unit tasks (Card et al., 1983) with durations of approximately 3–30 sec. Example unit tasks might include finding, moving to, and clicking on an icon on a computer screen, or, if you were a telephone toll and assistance operator, handling one customer call. Edges in the network represent dependencies of operators on other operators.

The representations used in SANLab-CM are adaptations from scheduling theory (see Modor & Phillips, 1970; Weist & Levy, 1977) and existing literature in experimental psychology (Ehrenstein et al., 1997; Schweickert, 1978, 1980) and cognitive science (Gray & Boehm-Davis, 2000; Gray et al., 1993). For the purposes of SANLab-CM, an elementary operation is defined as an atomic process utilizing a single resource (cognition, perception, motor, etc.). Each operator has an associated type that identifies what resource it utilizes and dictates its default distribution and parameters. Operators are color coded by their type to make it easy to identify which operators belong to what operator type, and can be adjusted along with all of the default properties associated with that operator type. SANLab-CM provides a number of standard operator types (e.g., cognitive, perceptual, and motor operators, and system resource operators) and can be extended to include new operators as needed.

The user interface. Four main windows are used when SANLab-CM models are developed and run. The primary window, the model editor (see Figure 1A), is used for creating, selecting, and manipulating operators, connecting operators to create dependencies, and creating interactive routines. It is often used in conjunction with the toolbar (see Figure 1B), which lists all of the operator types in SANLab-CM as well as interactive routines (discussed below). Selection of an operator type in the Toolbox allows the user to point and click in the editor window to place new operators. To change an instantiated operator, one clicks on the appropriate property and the field becomes editable. After entering a new value, pressing the Return key confirms the change and updates all activities in the current selection. Operator types can also be changed after instantiation by right-clicking and selecting the "Change Operator Type" mechanism.

Once operators have been created, pairwise execution dependencies are established in simple from–to relationships by selecting the Connect tool (the "+" at the top of the Toolbox window), clicking and holding the mouse button down inside the "from" operator, dragging the mouse (with an animation of a line) to the "to" operator, and releasing the mouse button. In the logic of activity networks, the second operator cannot execute until the first operator has completed execution.

SANLab-CM provides an at-a-glance view of the model by rendering it at 1/10th its size in the Model Overview window (see Figure 1C). This window facilitates the creation and visual analysis of the model. The gray area in the overview represents the current view displayed in the editor and can be dragged around to move to a different point in the model. Additionally, double-clicking on any point in the overview will center the display on that location, to make it easier to move from one end of the model to the other.

Finally, after the software finishes executing a model, a Results window is displayed (see Figure 2). By default, it renders the predicted task execution time for each computed trial in the form of a histogram. Below the histogram is a list of all of the critical paths, the longest paths through the network on any individual trial, and their respective means and standard deviations. Modelers can look at a subset of the data by selecting a critical path from the list of paths. In this mode, only trials that resulted in the chosen critical path will be rendered. The Results window also renders an average trial Gantt chart where the average starting time and duration of each operator is used to compute the size of the operator. Similar to the histogram view, choosing a critical path in this view gives a Gantt chart that visualizes only the trials that resulted in the specified critical path. The underlying trial data may also be exported into an Excel file to be used in external tools for visualization or statistical processing.

Interactive routines. To facilitate the creation of CPM-GOMS models, SANLab-CM provides a small set of preconfigured groups of interconnected operators.

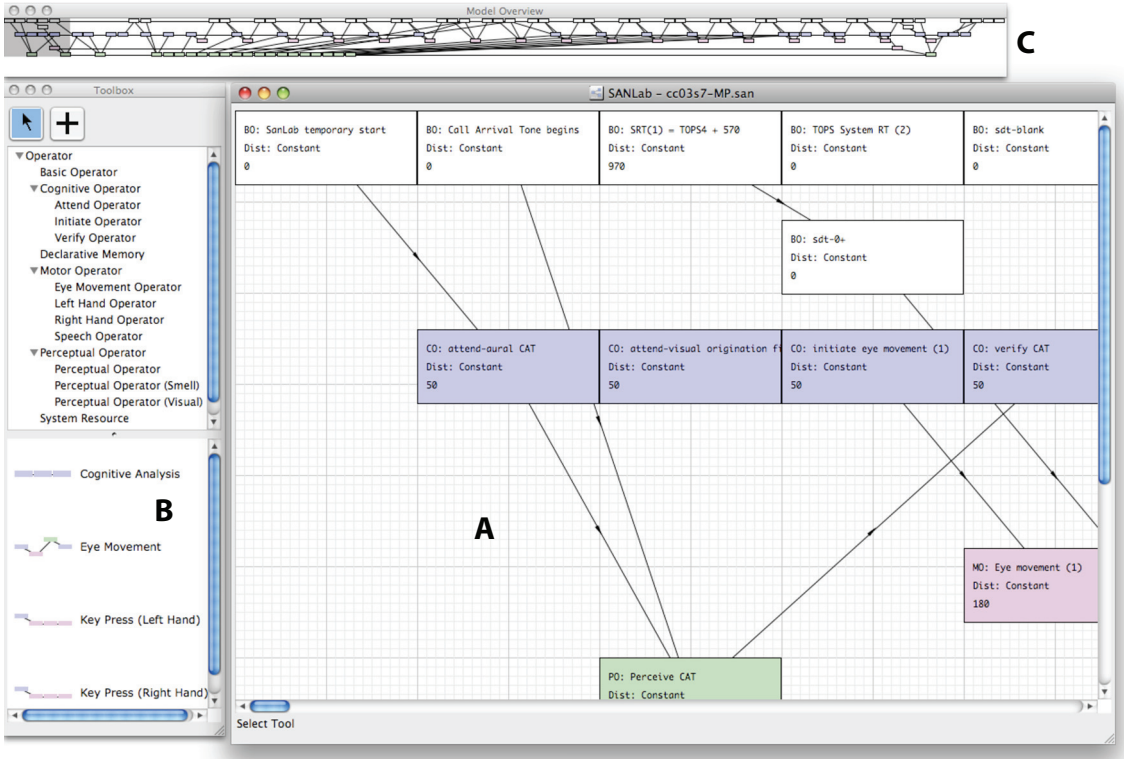

Figure 1. The SANLab-CM Interface: (A) The main editor (large window at the center) where operators are placed and connected. (B) The Toolbox (vertical window on the left), which lists all of the available operators and interactive routines for creating models. (C) The Overview window (horizontal window at the top), which displays the model in miniature and allows the modeler to jump to any point in the model.

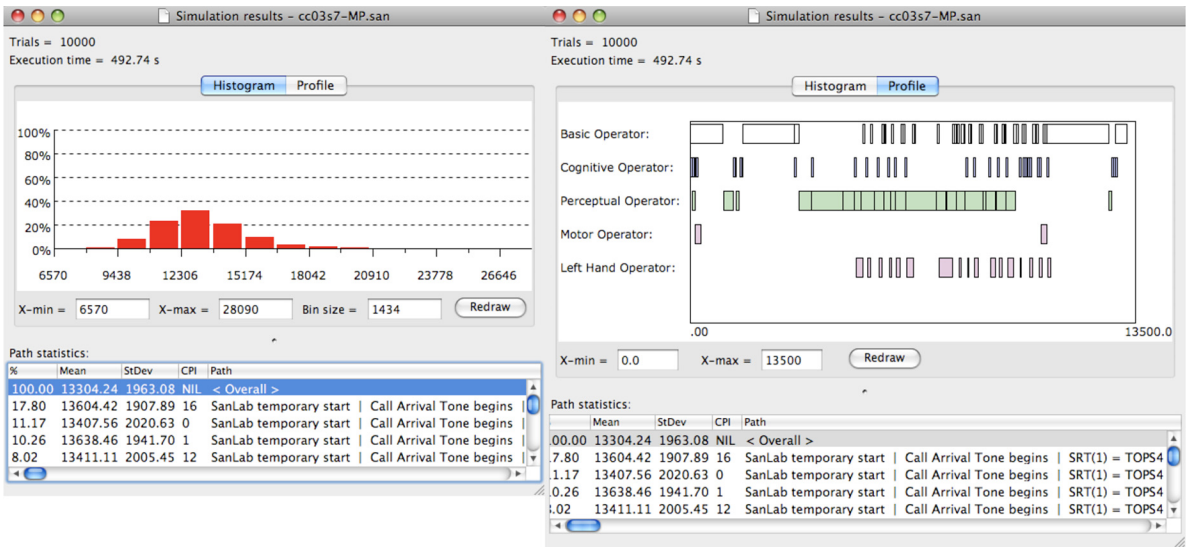

Figure 2. The two tabs of the Results window: Histogram (left) and Profile view (right). The Results window provides three critical functions. First, it displays a histogram, which summarizes the model execution times (left). The modeler can also see a Gantt chart of the operator execution versus time in the Profile view (right). Below the Histogram and Profile views is the list of every critical path that appeared in the run of the model. Selecting a critical path segments the data and will rerender the histogram and Gantt charts using only the data selected.

When the operators combine cognitive, perceptual, and motor operations at a certain temporal resolution, they are called *interactive routines* (see Gray et al., 2006). The default SANLab-CM package includes interactive routines for mouse clicks, eye movements, and keypresses, and can be enriched by a modeler to include additional routines useful to her area of research. When the modeler requires an interactive routine, she drags the routine from the toolbar to the main editor window. On release of the mouse button, SANLab-CM will prompt for a string to label this routine (strings need not be unique). The provided string is appended to the name of each operator in the routine for easy identification. For example, a modeler interested in typing could drag a "Key Press" routine to the window and append the string "J key" if the particular task required the participant to press the "J" key on the keyboard.

Once the necessary information has been supplied to the editor, a ghost image of the routine will follow the cursor, allowing the researcher to visualize how the interactive routine fits into the existing model. Clicking the mouse button will place the routine, instantiating all of its operator nodes and the connections between them. This also sets the default distribution and parameters (as discussed below) to those specified within the interactive routine (as opposed to the defaults specified for the activity type). Interactive routines are color coded in the editor and in the Overview window. This provides an easy way to identify which routines are being used within the model and how often they are used. This visual information can be combined with the critical path information displayed after model execution to allow modelers to quickly identify which interactive routines are most often on the critical path.

Distributions: Supplying Values to Nodes

SANLab-CM supports a plethora of distributions for calculating execution times for individual operators. Most of the default set of distributions are commonly used probability distributions like the normal and gamma distributions (see Luce, 1986; Schweickert et al., 2003). There are also variants of these distributions that take a mean and a coefficient of variation (CV) instead of the standard distribution parameters. For convenience, SANLab-CM provides a "constant" distribution that behaves in the same manner as the Dirac delta function to support nonstochastic models. The one-dimensional version of Fitts's law and a variant with Gaussian noise are also included for calculating execution times for pointing operations (e.g., mouse movement and homing to keys). SANLab-CM provides an editor (see Figure 3) for creating and manipulating distribution functions, which allows modelers to extend the underlying code. Functions must be written in ANSI Common Lisp and are read and compiled when the application starts. SANLab-CM stores all of the distributions on disk, making modified and new distributions easily available for sharing with the research community.

Model Execution and Data Visualization

Once a modeler has prepared a model, it must be executed by SANLab-CM to produce critical paths. At the start of a run, the modeler will be prompted for the number of iterations to calculate. After the number is entered, SANLab-CM checks that the current version of the model has been saved and that it conforms to certain rules. The trials then are performed in the following manner: (1) The system resets all internal values and calls the distribution function for each operator instance in the model to assign a duration; (2) the critical path is computed through the model and is stored as a record to be returned after all trials are computed; (3) additional statistics are computed, keyed to each critical path (e.g., conditional mean times, discussed below).

Exemplar: Adding Variability to Existing Models

In order to demonstrate the tools SANLab-CM provides, we utilize the CPM-GOMS model developed by Gray et al. (1993) for call type three in their Project Ernestine

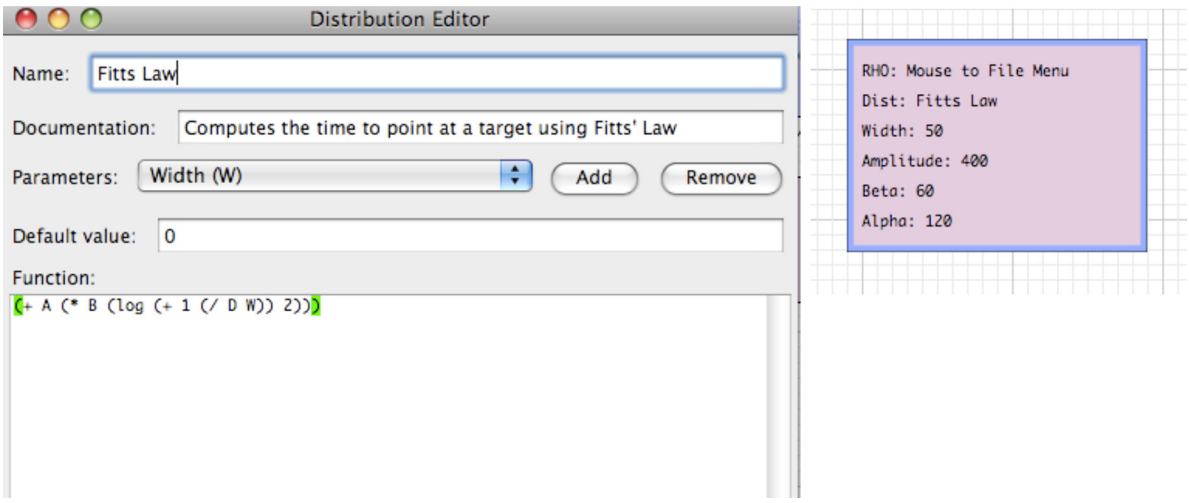

Figure 3. The Distribution Editor, left, allows for SANLab-CM to be extended with new timing calculations. A sample node, right, uses this new Fitts's law distribution to calculate the time required to point at a menu item.

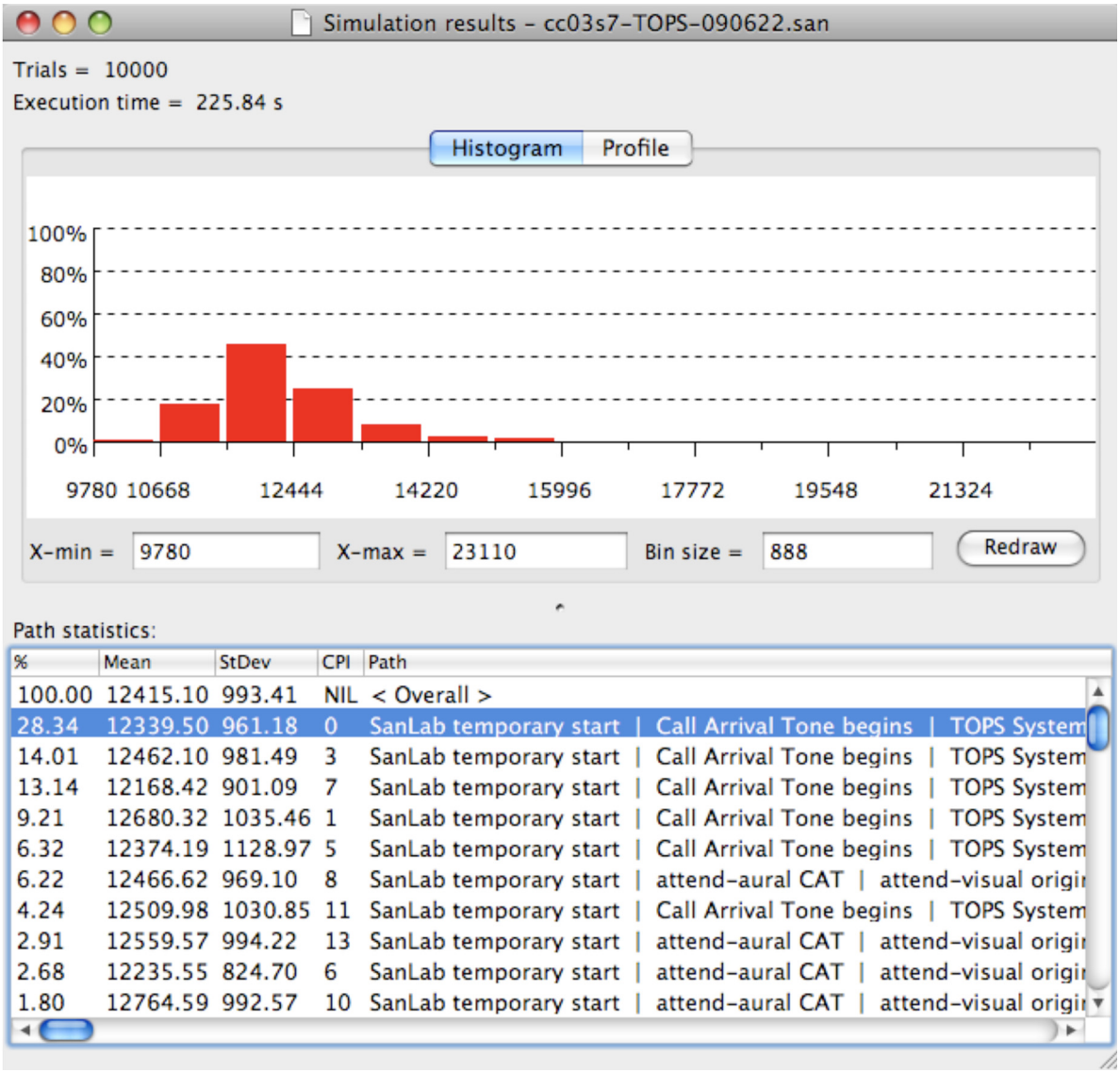

Figure 4. A histogram (above) of trial execution times produced after running the model 10,000 times, showing a clear positive skew. Below the histogram is the list of critical paths, ordered by percentage of occurrence, along with their respective means and standard deviations.

study. This particular task required a telephone operator to obtain a calling card number from a customer and enter it into her terminal, check that the number was valid for billing, and then proceed to let the customer place his call, all of which requires a rich mixture of perception, cognition, and action. The model was imported into SANLab-CM from MacProject, and some minor adjustments were made by assigning operator types to the network nodes. Executing the standard CPM-GOMS evaluation at this point determined that the critical path without variability was 12.240 sec for this model (which agreed with the times obtained in the original study).

Human variability is important in understanding the likelihood that two models predicting different task completion times are a reliable prediction of differences in human response times (RTs). Variability is also important in that slight differences in times for individual operators might result in multiple critical paths. SANLab-CM allows the modeler to assign different distributions for different operators or different operator types (these assignments can be based on empirical data or theoretical considerations). In the absence of other ways of determining an operator's distribution, Schweickert et al. (2003) followed Luce (1986) in suggesting that human operations are well modeled by a gamma distribution. Likewise, parameter settings for a given distribution (including the gamma distribution) may be based on empirical or theoretical considerations. For the gamma distribution, Schweickert et al. recommended using a CV of between 0.1 and 1.0. Since the original Project Ernestine models included means but no estimate of variability, we followed Schweickert et al.'s suggestions and set all of our operators to the "Gamma

CV" distribution. For the CV, we used Lisp code to set the CV to between 0.1 and 1.0 of the mean and to randomly sample a different CV from this range on each trial.

Results

Given these modifications, we ran the model for 10,000 trials. The most frequent critical path, used on 28.34% of trials, has a mean execution time of 12.339 sec (compared with 12.240 sec for the nonstochastic model) and a standard deviation of 0.981 sec. This path matches the original nonstochastic critical path, with the nonstochastic path within 0.101 standard deviations of the observed mean. Overall, 57 unique critical paths were taken through the model, with 15 of those accounting for 95% of trials (see Figure 4). Many of these critical paths are very similar to each other, often differing in the inclusion or exclusion of a small number of operators. However, across our collection there were some dramatic differences between critical paths (see Figure 5). Between the two most frequent critical paths, the one taken most often is dominated by the rate at which the customer speaks the digits in his billing number, whereas the second begins with dependencies on perceptual operators but eventually switches to being dependent on the typing speed of the operator. The particular operation that causes the paths to diverge takes only 116 msec on average in the first critical path, whereas it takes 179 msec on average to execute in the second critical path, an increase of 63 msec, or 54%. In some cases, the completion time of the task is driven almost exclusively by the speed with which the telephone operator can type the digits into the system. These examples show how the ability of the user (motor speed) and external factors (the rate at which the customer speaks) can affect which resources are most critical to task completion.

Summary and Conclusions

SANLab-CM's ability to incorporate variability into activity network models and its built-in support for clusters of co-occurring operations (e.g., interactive routines) provides a fresh look at how slight variations in operator execution can affect the ultimate critical path an agent has to take to complete a task and will offer new ways to test theories about the time distribution of cognitive, perceptual, and motor operators. The Project Ernestine model with probability distributions showed that many different critical paths can be obtained when operations vary and that these critical paths may differ greatly from each other. Being able to examine and analyze these large variations in resource utilization may shed light on human performance under information overload and other high-stress situations. The opportunity SANLab-CM provides to rapidly build activity network models might also help to extend prior work on interacting with computer software (e.g., Gray & Boehm-Davis, 2000) or scheduling processes in working memory (e.g., Ehrenstein et al., 1997). As the model execution revealed, one large variation at a single node in the network can dramatically affect which resources become critical to task completion. By helping identify key bottlenecks in a task, we might better design systems that reduce the variance of different operators, eliminating diverging paths and resulting in more predictable use of mental and external resources.

Future Work

Although SANLab-CM has many features, it is a work in progress and there are a number of tools that could help expand its research capabilities. Tools to investigate the criticality of operators in the network and provide quantitative comparisons between critical paths are in progress and will appear in future versions of the software. More advanced distributions will also be made available in the future, including integration to the ACT–R memory model to compute recall time for a memory retrieval and guidelines for exporting a run of an ACT–R model to SANLab-CM to facilitate the exploration of variability on ACT–R's predictions.

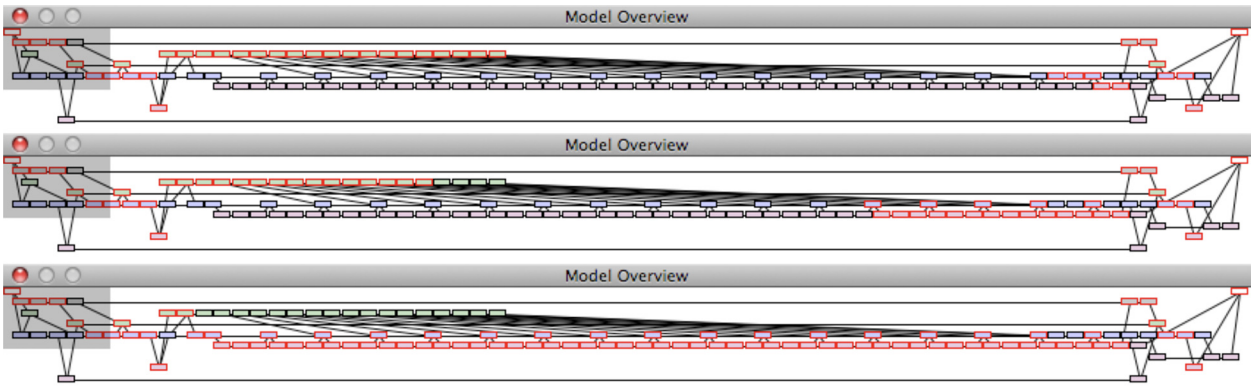

Figure 5. Three different critical paths produced through the model. The first is the most often occurring path, through which perception controls the critical path (the perception operators are red-highlighted toward the top of the model). The second occurs 14% of the time and shows what happens when the telephone operator falls behind on typing characters about two thirds of the way through handling one phone call (compared with the top critical path, more motor operators are red-highlighted toward the end of the model). The third occurs rarely (only 0.43% of the time), but shows that slower motor movements can dominate the critical path (the red-highlighted operators toward the bottom of this model are all motor operators).

Distribution of SANLab-CM

We are currently planning a beta test of SANLab-CM at a half dozen universities this fall and in early spring. Researchers who plan to use SANLab-CM in their laboratories or in courses that teach cognitive modeling and methods have signed up for our trial from Purdue University, Rice University, University of Illinois at Urbana-Champaign, University of Groningen, University of Nottingham, and (of course) Rensselaer Polytechnic Institute. For those who might modify or extend SANLab-CM, we are providing copies of our LGPL-licensed Lisp code. For others, we provide applications that can run under either the Macintosh OS X or Microsoft Windows. Please contact the authors to be included in our listserv and to gain access to the SANLab-CM Wiki (documentation, discussion, and sample class materials) and download site (software).

AUTHOR NOTE

E.W.P. worked on this project while being funded, in part, by the Lockheed Martin Advanced Technology Laboratory. The work was supported, in part, by Grant N000141010019 to W.D.G. from the Office of Naval Research, Dr. Ray Perez, Project Officer. Correspondence concerning this article should be addressed to E. W. Patton, Department of Computer Science, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 (e-mail: pattoe@rpi.edu).

Note—This article is based on a presentation at the meeting of the Society for Computers in Psychology, November 2009.

REFERENCES

Card, S., Moran, T. P., & Newell, A. (1983). *The psychology of human–computer interaction*. Hillsdale, NJ: Erlbaum.

Cary, M., & Carlson, R. A. (2001). Distributing working memory resources during problem solving. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **27**, 836-848.

Dzhafarov, E. N., Schweickert, R., & Sung, K. (2004). Mental architectures with selectively influenced but stochastically interdependent components. *Journal of Mathematical Psychology*, **48**, 51-64.

Ehrenstein, A., Schweickert, R., Choi, S., & Proctor, R. W. (1997). Scheduling processes in working memory: Instructions control the order of memory search and mental arithmetic. *Quarterly Journal of Experimental Psychology*, **50A**, 766-802.

Fisher, D. L., & Goldstein, W. M. (1983). Stochastic PERT networks as models of cognition: Derivation of the mean, variance, and distribution of reaction time using order-of-processing (OP) diagrams. *Journal of Mathematical Psychology*, **27**, 121-151.

Fu, W.-T., & Gray, W. D. (2004). Resolving the paradox of the active user: Stable suboptimal performance in interactive tasks. *Cognitive Science*, **28**, 901-935.

Goldstein, W. M., & Fisher, D. L. (1991). Stochastic networks as models of cognition: Derivation of response-time distributions using the order-of-processing method. *Journal of Mathematical Psychology*, **35**, 214-241.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, **6**, 322-335.

Gray, W. D., John, B., & Atwood, M. (1993). Project Ernestine: Validating GOMS for predicting and explaining real-world task performance. *Human Computer Interaction*, **8**, 237-309.

Gray, W. D., Sims, C. R., Fu, W.-T., & Schoelles, M. J. (2006). The soft constraints hypothesis: A rational analysis approach to resource allocation for interactive behavior. *Psychological Review*, **113**, 461-482.

John, B. E. (1988). *Contributions to engineering models of human–computer interaction*. Pittsburgh: Carnegie Mellon University Press.

John, B. E. (1990). Extension of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. *Proceedings of SIGCHI, 1990* (pp. 107-115). New York: ACM.

John, B. E. (1996). TYPIST: A theory of performance in skilled typing. *Human–Computer Interaction*, **11**, 321-355.

John, B. E., & Gray, W. D. (1992, June). *GOMS analysis for parallel activities tutorial*. Paper presented at the ACM CHI '92 Conference on Human Factors in Computing Systems, Monterey, CA.

John, B. E., & Gray, W. D. (1994, April). *GOMS analysis for parallel activities tutorial*. Paper presented at the ACM CHI '94 Conference on Human Factors in Computing Systems, Boston.

John, B. E., & Gray, W. D. (1995, May). *GOMS analysis for parallel activities tutorial*. Paper presented at the ACM CHI '95 Conference on Human Factors in Computing Systems, Denver.

John, B. E., & Gray, W. D. (1996). *GOMS analysis for parallel activities*. Fairfax, VA: George Mason University Press.

John, B. E., Vera, A., Matessa, M., Freed, M., & Remington, R. (2002, April). *Automating CPM-GOMS*. Paper presented at the ACM CHI '02 Conference on Human Factors in Computing Systems, Minneapolis, MN.

Luce, R. D. (1986). *Response times: Their role in inferring elementary mental organization*. New York: Oxford University Press.

Modor, J. J., & Phillips, C. R. (1970). *Project management with CPM and PERT* (2nd ed.). New York: Van Nostrand.

Schweickert, R. (1978). A critical path generalization of the additive factory method: Analysis of a Stroop task. *Journal of Mathematical Psychology*, **18**, 105-139.

Schweickert, R. (1980). Critical-path scheduling of mental processes in a dual task. *Science*, **209**, 704-706.

Schweickert, R., Fisher, D. L., & Proctor, R. W. (2003). Steps toward building mathematical and computer models from cognitive task analyses. *Human Factors*, **45**, 77-103.

Sternberg, S. (1969a). The discovery of processing stages: Extensions of Donders' method. *Acta Psychologica*, **30**, 276-315. (Also published in W. G. Koster [Ed.], *Attention and Performance II*. Amsterdam: North-Holland.)

Sternberg, S. (1969b). Memory-scanning: Mental processes revealed by reaction-time experiments. *American Scientist*, **57**, 421-457.

Vera, A. H., John, B. E., Remington, R., Matessa, M., & Freed, M. A. (2005). Automating human performance modeling at the millisecond level. *Human–Computer Interaction*, **20**, 225-265.

Weist, J. D., & Levy, F. K. (1977). *A management guide to PERT/CPM: With GERT/PDM/DCPM and other networks* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

NOTES

1. CPM represents both "critical path method" and "cognitive, perceptual, and motor operators." GOMS is an acronym for Goals, Operators, Methods, and Selection Rules.

2. Although anecdotal, this assessment is based on the experiences of John and Gray (1992, 1994, 1995, 1996) in giving 4- to 8-h tutorials at the annual ACM-SIGCHI conference, and on Gray's attempts to teach the spreadsheet alternative in 2001 and 2002.

(Manuscript received November 11, 2009;
revision accepted for publication April 5, 2010.)