In Proceedings of ICCM - 2007- Eighth International Conference on Cognitive Modeling. 133 - 138. Oxford, UK: Taylor & Francis/Psychology Press.

The Costs of Multitasking in Threaded Cognition

Jelmer Borst (jpborst@ai.rug.nl)[1,2,3]
Niels Taatgen (taatgen@cmu.edu)[1,2]
[1]Department of Psychology, Carnegie Mellon University, USA
[2]Artificial Intelligence, University of Groningen, The Netherlands
[3]School of Behavioral and Cognitive Neurosciences, University of Groningen, The Netherlands

Abstract

Most multitasking models make use of executive processes to assign resources to tasks (Kieras et al., 2000). An alternative is to have no executive, but constrain individual processes so that they share resources in a plausible way. Salvucci and Taatgen (under revision) in their theory of threaded cognition have shown how peripheral resources and declarative memory are shared between processes without an executive. In this paper we will extend this work by showing how two tasks share a resource to store the problem representation in a dual-task paradigm where either task sometimes needs a problem representation and sometimes not. Threaded cognition predicts extra interference when both tasks need a problem representation, which is what we found in the experiment.

Introduction

Human beings are amazingly adept at performing multiple tasks concurrently, and at combining previously unrelated tasks. This stands in sharp contrast to the current situation in cognitive modeling, where most models of multitasking make use of a so-called *Customized Executive* (Kieras et al., 2000). This is an, often complicated, control process specialized for the tasks at hand. It determines how the tasks will be interleaved, and at which point one of the tasks takes precedence. A consequence of this is that for every two tasks a different control structure is required, which, in turn, implies that we would have to learn a new control structure for every new combination of tasks. A more plausible solution would be to have a *General Executive* that could interleave any two tasks (Kieras et al., 2000; Salvucci, 2005). There have been several proposals for such a General Executive in cognitive architectures (e.g., Kieras et al., 2000 (EPIC); Salvucci, 2005 (ACT-R)). However, these proposals have not been equally successful in accounting for multitasking data as customized executive approaches.

Yet another possibility is to have no executive at all (e.g., Liu, Feyen, & Tsimhoni, 2006), which is the underlying idea of the new multitasking theory 'Threaded Cognition' of Salvucci and Taatgen (under revision). The essence of threaded cognition is that it has no central executive, but instead makes sure individual tasks in a multitasking situation interleave without top-down control. This interleaving of individual tasks sometimes leads to additional costs. Salvucci and Taatgen have already shown how declarative memory can be a contended resource, and that competition for this resource can explain differences between novices and experts on a task. In the current paper we will show evidence for a second shared central resource: the problem representation. We will use a dual-task situation with two relatively complex tasks: driving and operating a navigation device. The experimental manipulation is to have two variations of each of the two tasks, one that does require a problem representation, and one that does not.

First, we will outline threaded cognition, and show what kind of multitasking costs the theory predicts. Second, we will test in an experiment whether this prediction is correct, and finally compare the results of the experiment to a model designed with threaded cognition.

Threaded Cognition

Threaded cognition posits that each task (in a multitasking context) is represented by a cognitive thread (Salvucci & Taatgen, under revision). Each of these threads has its own control structure: there is no central executive; threads are independent and can be run in isolation. Threaded cognition can therefore account for the flexible way humans combine previously unrelated tasks, and for the fact that many tasks can be learned in isolation first and performed together later.

All threads are processed together on a single processor, which can only execute one rule at a time, and will therefore present a bottleneck (this in contrast to the approach of Kieras et al., 2000). At any given time, production rules of all threads can be selected, when multiple rules (of different threads) match, the rule belonging to the thread that has least recently been processed will be executed. This makes sure none of the threads will starve as long as it has matching production rules.

While the central processor presents a first bottleneck, it is not the only one. The threads have to share resources like memory and vision, which creates additional interference. For instance, if two threads need to retrieve a fact from declarative memory, the one that comes first can request retrieval, and the second thread will have to wait. A second consequence of resource sharing is that threads have to be polite, in that they should not 'steal' resources from another thread, as this could result in an infinite loop.

Costs of multitasking

As explained above, possible bottlenecks in the model are the central processor and resource sharing. In the current paper we investigate interference of sharing the problem representation resource. If a thread has to keep a problem state in mind, for instance a partial solution to a problem, and another thread has to keep track of its own problem state, both threads will have to restructure their problem state every time they take control (assuming only one

problem state can be maintained at a time). Thus, threaded cognition predicts additional interference in case two threads both have to keep track of their own problem state. Additional in the sense that the problem representation has to be restored on every task switch, in contrast to the use of the visual or memory resource where threads only have to wait sometimes, but can carry on afterwards.

The current paper tests this prediction by comparing two tasks in two conditions, an easy condition in which no problem state is necessary, and a hard condition in which it is. Thus, suppose performance is 100% if both tasks are easy, and 90% when one of the two tasks is hard (because of perceptual / motor / memory resource sharing), threaded cognition predicts a task performance lower than 80% in the condition when both tasks are hard.

Threaded Cognition & ACT-R

Because threaded cognition strives to be an "integrated" theory, it is implemented in the cognitive architecture ACT-R (Anderson et al., 2004). ACT-R is a cognitive architecture consisting of specialized modules functioning around a central production rule system. This production system works on a single goal at a time, for which it sequentially executes production rules. In order to achieve multitasking, a control structure is needed that switches between the multiple goals at the appropriate moments, essentially requiring a customized executive for each combination of tasks.

A possible solution for this problem could be, as stated above, threaded cognition. This is implemented in ACT-R in the following way. Instead of only one goal, ACT-R is now allowed to have multiple goals. Each goal represents a thread, and will have a number of dependent production rules. However, as in standard ACT-R, only one rule can fire at any given time. If production rules related to different goals match at the same time, threaded cognition will select the rule belonging to the least recently processed goal.

In ACT-R, the problem representation has to be stored in the imaginal buffer, which has to be shared by multiple tasks. In combination with threaded cognition this clearly predicts strong interference if two tasks have to keep track of a problem represenation.

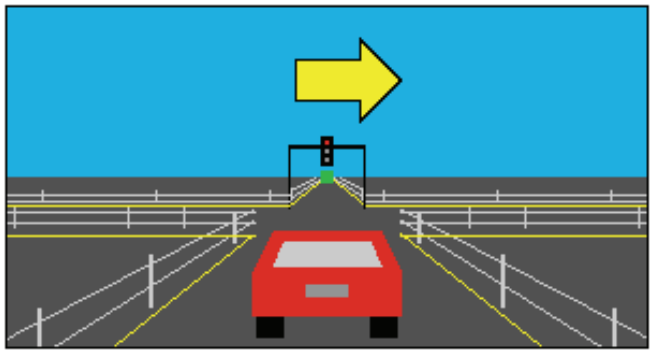

Figure 1. Example of an 'easy' intersection.

The Experiment

To test our hypothesis we modified the discrete driving task of Salvucci, Taatgen and Kushleyeva (2006). This is a task in which participants have to steer a car down a road on the left side of the screen, while entering information into a navigation device on the right. As explained above, for our current purposes we needed two tasks, both with a hard condition in which participants have to keep track of a problem state, and an easy condition in which this is not the case. To this end we modified both parts of the discrete driving task. We will describe both tasks in detail below.

Driving

In the driving part of the experiment the participants' main task is to keep the car in the middle of the road. Every few moments (0.5, 0.75, or 1.0 seconds, with equal probability) the car is perturbed 10 pixels to the right or to the left. It can be steered back to the middle of the road by pressing 'a' or 'd' (left or right, respectively), which also resets the perturbation timer. When the car is in the middle of the road, it will move to the left or to the right with equal probability. When it is already on one of the sides, it will move in 2/3 of the cases further to that side.

Every 15 seconds the car reaches an intersection, where it can either go left, straight, or right (keys 'q', 'w', 'e'). In the easy condition, participants are shown where to go by an arrow above the intersection, as in Figure 1. They only have to press the corresponding key on the keyboard, and do not have to keep track of past or upcoming intersections. In the hard condition, four arrows are shown at the first intersection of a set of four, and none on the other three. This means that participants have to (1) remember where to go on a series of three intersections, and (2) keep track of how many intersections they have already passed in the current set. The four arrows are shown for a maximum of 3 seconds.

Navigation

Navigating is done using the mouse, and while it has to be performed concurrently with steering the car, participants use their left hand to steer the car with the keyboard and use their right hand for navigation with the mouse.

The navigation task starts with an initial screen with five buttons: street number, street name, city, state, and done. These buttons are used to choose the category to be entered, but as only one of them is active (and highlighted) at a time, this part of the task is trivial. When one of the buttons is clicked a keyboard appears, as in Figure 2 (in case of street name, city, or state the keyboard is completely alphabetic).

Figure 2 shows an example of the easy task. In this case the to-be-entered character is present in the display, the only thing a participant has to do is to click the corresponding key on the keyboard. As soon as the click is registered a new stimulus appears; this continues until the whole number/name is entered (the participant has no access to the full name, and can therefore not plan ahead). After all

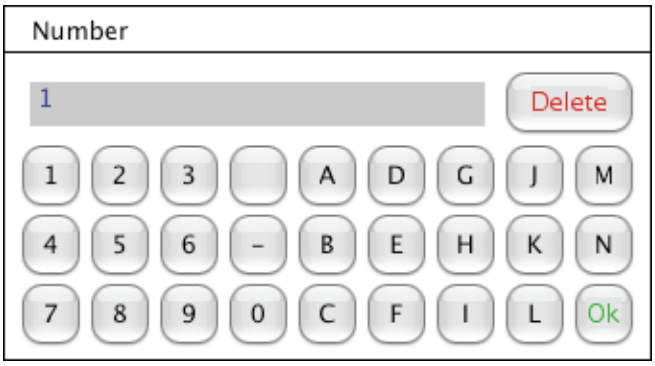

Figure 2. Navigation display in the easy variant.

characters of a name have been entered 'OK' is shown in the input field, when the participant clicks the OK-button the task returns to the initial display and the next category is highlighted. When all four parts of the address have been entered the Done-button is highlighted, when that is clicked the display disappears for 10 seconds, after which a new display appears. When the car reaches an intersection, the buttons of the navigation device become inactive, to become active again as soon as the participant steered.

In the hard condition a whole number/name is shown in the input field at once, however, it disappears as soon as the participant starts typing. Also, no feedback is offered to the participant as to what they have entered; only a 'click' can be heard every time a button is clicked. This means that the participant has to keep in mind what word they are typing and which character of the word has to be entered next.

In both conditions the numbers were three digits long, the street names six letters, the city names contained nine letters and the states were the normal two-letter abbreviations. In the hard condition, real street / city / state combinations of well-known cities were used. In the easy condition the characters of these names were scrambled to prevent participants from guessing the word.

Eye-tracking

To investigate which of the two tasks the participants were focused on at a particular moment, we used an Eyelink II head-mounted eye-tracker (SR Research) to record eye movements.

Participants

27 people agreed to participate in the experiment for monetary compensation. As one of them left halfway through the experiment because of a fierce headache, there are 26 complete datasets (11 female, age range 18-34, mean age 23.4). All of the participants had normal or corrected-to-normal visual acuity. Informed consent was obtained before testing. Due to technical difficulties the eye-tracking data of 6 participants could not be analyzed.

Experimental set-up

The experiment started with five practice blocks: easy driving: 2 blocks of 4 intersections; hard driving 2x4 intersections; easy navigation: 2 complete addresses; hard navigation: 2 addresses; combination: one set of each condition combined: 4 sets of 4 intersections and a complete address. This might sound a bit overdone as the single tasks are quite easy, but as the response of many participants indicated at the combination practice ("this is impossible!"), it was necessary.

After the practice block the participants were asked to do the single tasks in isolation, to measure their base level performance (3 sets of 4 intersections in the two driving conditions, 3 addresses in the two navigation conditions). The main part of the experiment existed of two blocks of 12 4-intersection sets and addresses each, thus 24 sets in total. At the end of the experiment the single tasks were once again administered, to control for learning effects. Between the different blocks participants could take a break, which they usually only did halfway the main phase. The complete experiment lasted approximately 1.5 hours.

The Model

To model this task we used threaded cognition and ACT-R. The experiment consists of two tasks that can be performed in isolation: driving and navigation. Thus the model will have two threads, which we describe in turn below.

Driving thread

As long as the driving thread is the only active thread, it can constantly attend the road, and act promptly to every perturbation. However, most of the time a navigation thread is also present which needs to attend the navigation device. To know when it has to focus attention back on the road the driving thread needs a sense of time, which we implemented using the previously validated temporal module (Taatgen, Van Rijn, & Anderson, in press).

As long as the car is not on the center of the road, the driving thread will use the visual resource. It will give it up as soon as the car is on the middle of the road. As soon as it notices that the visual module is used by another thread and attends something else than the road (in this case the navigation device), the driving thread will start the timer of the temporal module. While the navigation thread is busy entering information into the navigation device, the driving thread tries to decide whether it is time to look at the road by retrieving past timing experiences, stored at the current timer value. If it retrieves an experience that says it is time to drive again, the driving thread attends the road, and steers the car back to the middle. It can also retrieve an experience saying it is still safe to continue navigation, in which case that is exactly what it does. If it fails to retrieve a past experience it will continue navigating half of the time, and go back to driving in the other half of the cases.

Where do these timing experiences come from? Every time the driving thread starts steering the car, it first stores whether this was already necessary or not (i.e., whether the car was far out of the middle of the road, or whether it was still driving safely in the middle) together with the timer value on which it looked back to the road; this forms a

timing experience. It should be noted that while the driving thread is combined with a navigation thread in this particular example, this is by no means necessary. Without making any changes to the driving thread, it can be combined with any other behavior performed while driving, like using a cell phone.

The driving thread steers the car back to the middle of the road by looking whether the car is to the left or to the right of the center, and pressing the corresponding key. When the car stops at an intersection, the model tries to find an arrow. If there is only one arrow, it presses the corresponding key. If there are four arrows, the model starts memorizing them by attending them in left to right order, until the arrows disappear after 3 seconds. It also changes its problem state to represent where it is in the current set of intersections. If it now arrives on an intersection with no arrows it retrieves the arrow corresponding to the current problem state from memory, and steers into that direction. Every time the driving threads steers the car back to the middle of the road it will also retrieve the arrow for the upcoming intersection, and, if necessary the problem state.

Navigation thread

Navigation starts with selecting a category: finding an active button and clicking it. If the task is easy, the model now perceives the stimulus and clicks the corresponding key. However, if the task is hard the model puts the to-be-entered information in its problem state and starts typing the first character. As soon as it clicks a button it starts searching for the next character of the word, and so on until the whole word has been entered.

It should be noted that both tasks are polite in the sense that they will only take over control when all resources are free, except for the problem state. There is one exception to this general rule: the driving task can request visual attention back immediately. This mimics real driving in the sense that when someone is paying attention for some time to entering information in a navigation device, at some point they will look back to check the state of the road, independent of whether they had finished entering all the information.

Whenever the model switches to the navigation task and notes that it is in a hard condition and does not have the right problem state, it will first request this from declarative memory, effectively pushing the problem state of the driving thread into declarative memory. Similarly, whenever the model switches to driving in the hard condition, it will restore the driving problem state.

Results

A visual inspection of the data showed that all learning took place before the main phase of the experiment: there was no noticeable difference between the base level measurements before and after the experiment. Therefore the rest of this paper will only be concerned with the main two blocks of the experiment. All reported *F*- and *p*-values are from ANOVAs, all error bars depict standard errors.

Task durations

The average duration of periods spent on one of the two subtasks can be seen in Figure 3 (driving sequence) and Figure 4 (navigation sequence). These durations are approximations, calculated in the following manner: the length of a driving sequence is defined as the time between two *navigation* actions (button clicks), with at least one driving action in between. Similarly, the length of a navigation period is the time between two *driving* actions with a navigation action in between.

Driving Figure 3 shows that the length of driving periods decreases when the navigation task becomes hard, *but only when driving is easy*. When navigation is hard, people know what they are going to type next ("philadel…"), which means that they do not have to find the stimulus first, but can start right away with entering navigation information.

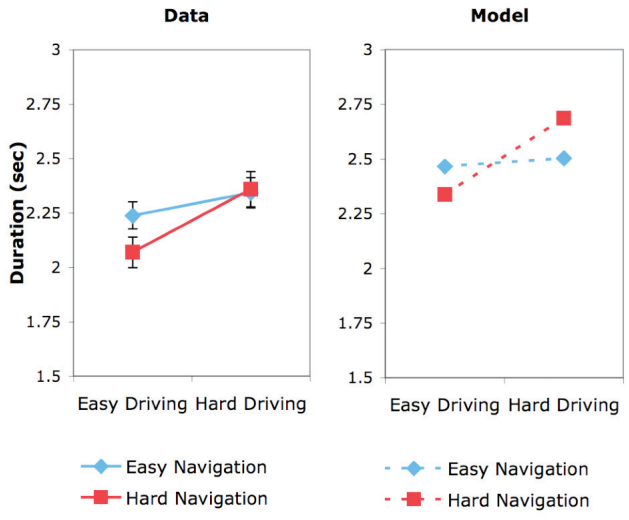

Figure 3. Duration of driving periods.

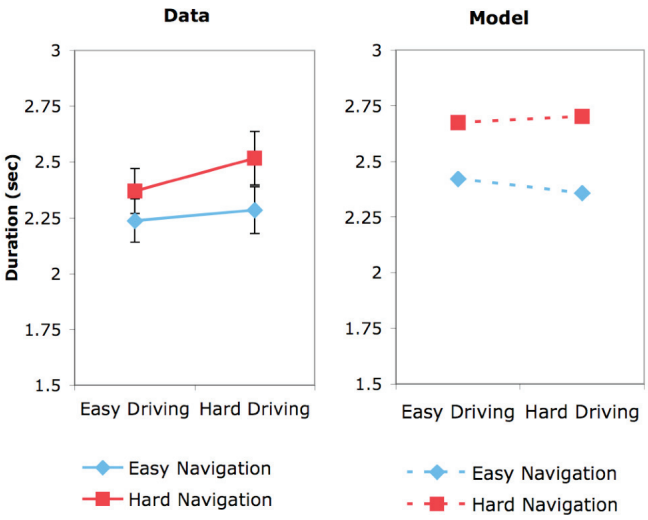

Figure 4. Duration of navigation periods.

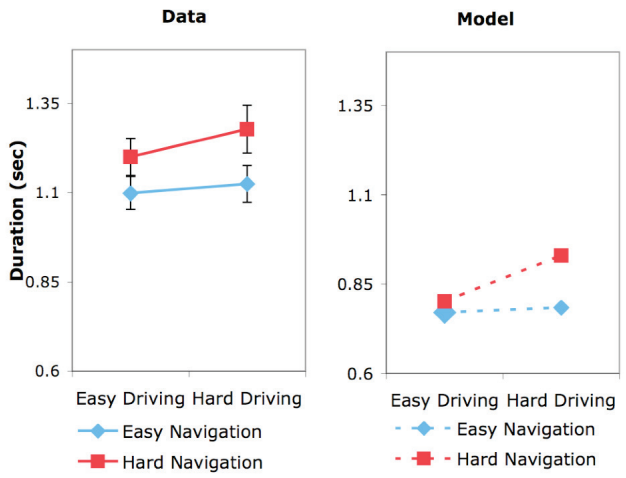

Figure 5. Duration of eye-tracking driving periods.

Because of the fact that the length of a driving sequence is measured as the time between two navigation actions with a driving action in between, the length of the driving sequence decreases when navigation becomes hard. However, this effect disappears when both navigation and driving are hard – it seems as if people have to reconstruct their problem state before they can start navigating, which increases the length of the driving periods. Overall can be seen that the length of driving periods increases with driving difficulty. An ANOVA showed indeed a main effect of driving ($F(1,25) = 65.414$, $p < .001$) and an interaction effect of driving x navigation ($F(1,25) = 13.906$, $p < .001$).

Navigation In Figure 4 can be seen that the duration of the navigation periods increases with task difficulty of navigation ($F(1,25) = 16.755$, $p < .001$). Driving has no significant effect on the length of the navigation periods, neither is there an interaction.

Model The model shows the same pattern as the experimental data: in the driving task (Figure 3, right panel) there is a significant interaction, while in the navigation task (Figure 4, right panel) there is no significant interaction. This is what threaded cognition predicted: there will be interference as soon as people have to keep track of a problem state in both tasks.

Task durations measured with eye-tracking

Figure 5 again shows the duration of periods spent on the driving task, but now as measured by the eye-tracker. The length of a period is now determined by where a participant was looking: as long as participants were looking at the right side of the screen it was recorded as navigation, as long as they were looking at the left side as driving. These measurements are arguably more accurate than the ones before: periods without any key-presses or mouse clicks are taken into account as well. This explains why the average length of the periods is about a second shorter than what we saw earlier.

Driving Interestingly, instead of decreasing, the length of driving periods now increases with navigation difficulty ($F(1,19) = 9.1367$, $p < .01$). This can be explained by the fact that finding and reading the stimulus in the easy condition no longer contributes to the driving periods. The reason for the increase is probably that participants tried to finish parts of a word ("phi…"), before going back to driving, an effect that will not occur in the easy driving condition and will make for longer navigation periods. The longer participants spend on navigation, the longer they need to steer the car back to the middle of the road. There is also a significant effect of driving difficulty ($F(1,19) = 14.455$, $p < .01$). Besides these two main effects, we found an interaction effect of driving x navigation as well ($F(1,19) = 14.931$, $p < .01$). This could be explained by the fact that people have to reconstruct their problem state before entering navigation information, and this preparation is done while looking at the driving display, as people can still control the car in that case.

Navigation No significant effects were observed in the duration of the periods spent on navigation (no graph is shown).

Model The model showed the same patterns, it only predicted the duration of the driving periods to be about 250 ms shorter (Figure 5, right panel). On the other hand, the duration of the navigation periods is predicted correctly (2.25 sec), without effects of condition.

Deviation

Due to space limitations we cannot show graphs of the average deviation of the middle of the road, but will describe it shortly. Deviation increases with task difficulty, this is both significant for driving, $F(1,25) = 21.010$, $p < .001$ and for navigation, $F(1,25) = 18.967$, $p < .001$. No interaction effect was found. The values range between 10 and 12 pixels.

However, the model shows an interaction effect. There is too much deviation in the easy driving/easy navigation condition. The duration of the navigation periods in the easy/easy condition is overestimated as well (Figure 4), and these two phenomena are connected. Because the model spends a little too much time in the easy/easy condition on navigation, it will also deviate further from the middle of the road. Furthermore, the model performed better than the participants, with deviation ranging between 6 and 8 pixels. However, about one third of our participants actually performed on that level, while some others were far worse than average. The model must be seen as a 'perfect' participant, in that it always manages to steer the car back to the middle of the road in exactly the right number of key-presses.

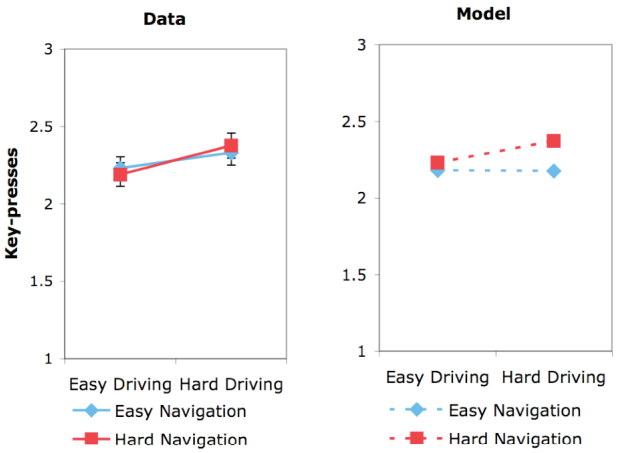

Figure 6. Mean number of key-presses per driving period.

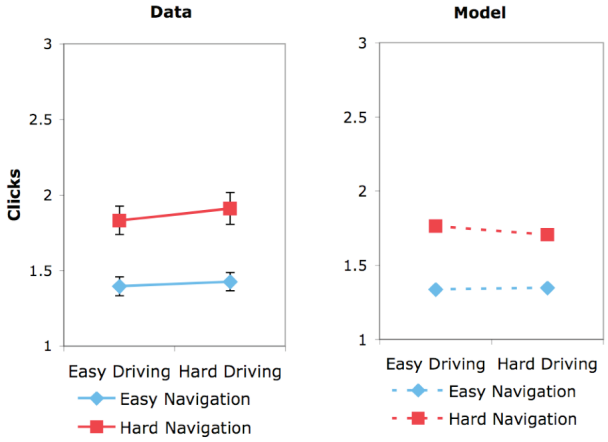

Figure 7. Mean number of clicks per navigation period.

Number of clicks / key-presses per period

In Figure 7 and Figure 8 is respectively shown how many times participants pressed a key during a driving period, and how many times they clicked a button during a navigation period. There is only one significant effect on the number of key-presses, which is driving ($F(1,25) = 13.475$, $p < .01$). The opposite is true for the number of clicks during a navigation period, this gives a highly significant effect of navigation ($F(1,25) = 229.54$, $p < .001$), and only a marginal effect of driving ($F(1,25) = 6.070$, $p = .02$).

The model shows the same effects, as can be seen in the right panels of both figures.

Discussion & Conclusion

As explained above, threaded cognition predicts an extra drop in performance when it is necessary to keep track of a problem state for two tasks. The results of the experiment clearly showed that this is in fact the case: we found a significant interaction effect in the length of driving periods. By modeling this task in ACT-R with threaded cognition, we showed exactly why these costs are connected to the driving task: the preparation of a problem state for both the driving and the navigation task is done while driving, and the need to reconstruct a problem state therefore increases the duration of driving periods. Eye-tracking measurements confirmed those results.

Threaded cognition is one of the first theories of multitasking without a control structure to interleave the subtasks. Salvucci & Taatgen (under revision) showed the value of this theory in a multitude of task combinations, they validated the theory on tasks ranging from simple laboratory tasks to real-world tasks, and showed the effects of sharing perceptual and memory. In the current paper we investigated whether threaded cognition can account for the costs of sharing another internal resource: the problem state. As we have made clear, threaded cognition predicted correctly in which conditions we had to expect extra costs of sharing a problem state.

Acknowledgments

This work was supported by Office of Naval Research grant N00014-06-1-005. We would like to thank Dario Salvucci for comments on an early version of the experiment.

References

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.

Kieras, D.E., Meyer, D.E., Ballas, J.A., & Lauber, E.J. (2000). Modern computational perspectives on executive mental processes and cognitive control: Where to from here? In S. Monsell & J. Driver (Eds.), *Control of cognitive processes* (pp. 681-712). Cambridge, MA: MIT Press.

Liu, Y., Feyen, R., & Tsimhoni, O. (2006). Queueing network-model human processor (qn-mhp): A computational architecture for multitask performance in human-machine systems. *ACM Transactions on Computer-Human Interaction (TOCHI), 13*(1), 37-70.

Salvucci, D.D. (2005). A multitasking general executive for compound continuous tasks. *Cognitive Science, 29*, 457-492.

Salvucci, D.D., & Taatgen, N.A. (under revision). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*.

Salvucci, D.D., Taatgen, N.A., & Kushleyeva, Y. (2006). Learning when to switch tasks in a dynamic multitasking environment. In D. Fum, F. D. Missier & A. Stocco (Eds.), *Proceedings of the seventh international conference on cognitive modeling* (pp. 268-273). Trieste, Italy: Edizioni Goliardiche.

Taatgen, N.A., Van Rijn, D.H., & Anderson, J.R. (in press). An integrated theory of prospective time interval estimation: The role of cognition, attention and learning. *Psychological Review*.