The Wickens's task: from Operand to Soar-7

Y. Lallement & B. E. John

Carnegie Mellon University
Human Computer Interaction Institute

This work was supported by the Office of Naval Research, Cognitive Science Program, Contract Number N00014-89-J-1975N158. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research or the U. S. Government.

Contents

1. Motivations

2. Current numeric results

3. Similarities and differences

4. Why the differences?

5. Golden nugget and lump of coal

Ron Chong's model was in Operand

→ One o-supported series of activation per decision cycle

→ Current operator redecided if o-supported items are created

→ Top state operator redecided on new input

Wickens's task in Soar-7: why?

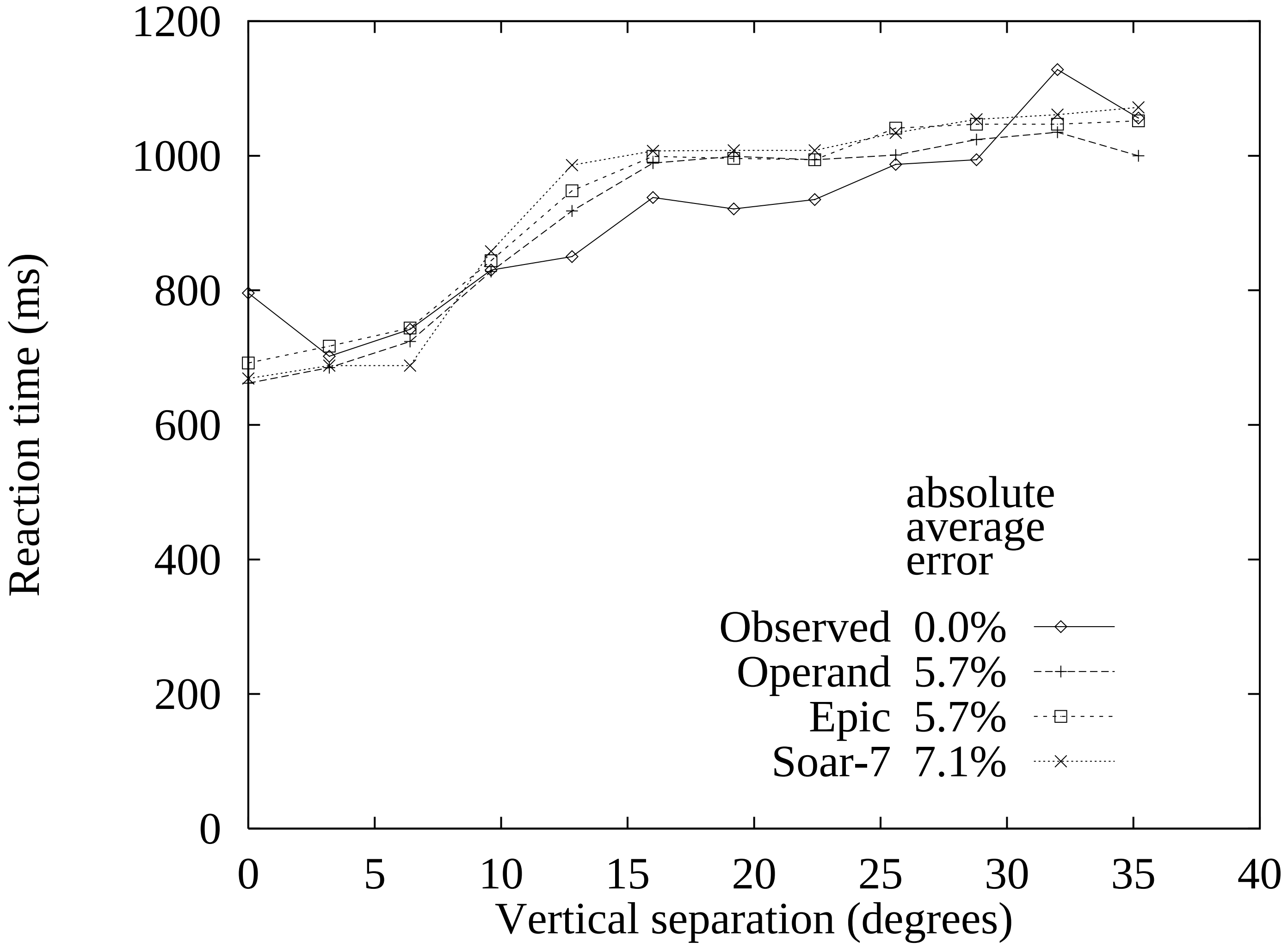
We wanted:

- Epic-Soar for a more complex task
- Wickens's task as a "previous skill" for our more complex task

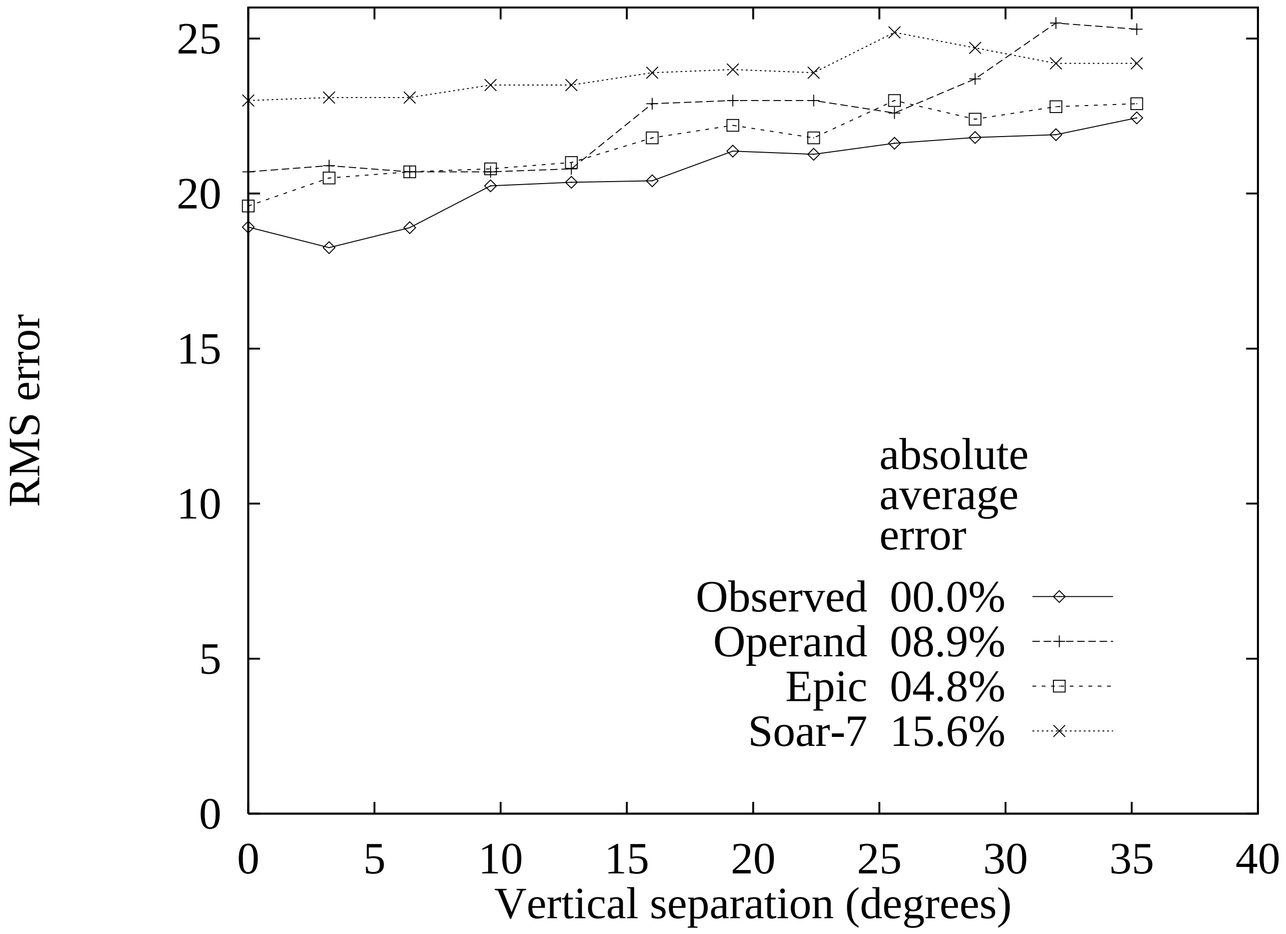
We have:

- Experience with Soar-7
- Experience with PEACTIDM-style models
- Operand is still an experimental architecture

Reaction time results



Tracking error results



Similarities and differences between the models' results

- Similar reaction times on choice task
- Different tracking errors
  - → Soar-7 model has positive slope without *disable-track* knowledge needed in Operand model
  - → Soar-7 tracking error consistently too high

Operand *vs* Soar-7 Wickens's task

	Operand	Soar-7
Best model	Concurrent + track-express + disable-track	Concurrent + track-express +
Impasses	ONC	SNC
Chunks	op. implement.	op. proposal
Parallelism	yes	no
Jam avoidance	learned	not necessary
Command removal	based on Operand	Soar-7 hack

Parallelism issue in Operand + ONC programming

- single `dual-task` operator proposal
- several op. implementation chunks can fire

→ 2 tasks in parallel in cognition

→ can launch 2 separate Epic motor processors at once

→ jam-repair required when same motor processor used

Parallelism issue in Soar-7 + SNC programming

- single operator proposed by a chunk
- implemented by its hand-coded application production

→ no parallelism in cognition

→ can only launch one Epic motor processor at a time

→ no jams

Does parallelism cause the difference in tracking error?

- Launching two motor processors in a single decision cycle does occur in Operand
- But lots of slack time in traces

→ Still not clear

Other possible causes

During initial recognition of the environment:

- Architectural: Operand can recognize several objects at a time

→ See parallelism issues

- Non architectural: Soar-7 sometimes looks at the wrong thing first

→ Is this a real difference?

Golden nugget and lump of coal

- Likely result (almost there!):
  Wickens's task can be modeled as well in Soar-7-Epic as in Operand-Epic.
- Novice Soar hacker's lament:
  stop the world, I want to back up!

Extra info for Soar hackers: command removal in Epic-Soar

Commands to Epic must be removed after output phase;
In Operand:

- o-supported `remove` production...
- that will fire only at the next cycle...
- and will be the only o-supported production to fire at that cycle

In Soar-7:

- obvious solution: `remove` operator; not PEACTIDM
- solution employed: place command on the output link, but also on the operator at proposal, and on the state at termination. Remove command on output link if operator and state commands are different. (Thanks Gary!)