

Cognitive Efficiency Considerations for Good Graphic Design

Stephen M. Casner
University of Pittsburgh

Jill H. Larkin
Carnegie Mellon University

ABSTRACT

Larkin and Simon's (1987) analysis of how graphical representations support task performance is applied to designing graphical displays that streamline information-processing tasks. Theoretically this streamlining is done by designing external data structures that (a) allow users to substitute less effortful visual operators for more effortful logical operators, and (b) reduce search for needed information. A design program called BOZ is used to produce four alternative displays of airline schedule information to support a task of making airline reservations. We postulate several procedures that use visual operators to perform the task using the different graphics. The number of times each operator is executed provides one measure of task difficulty (for a procedure and graphic). A second measure is the difficulty of executing each operator. Seven subjects performed the airline reservation task using each of the four graphics. Response times for the different graphics differ by a factor of two, which is statistically highly significant. Detailed data analyses suggest that these differences arise through substitution of visual operators for logical ones and through the use of visual cues that help reduce search. These analyses provide quantitative estimates of the time saved through operator substitutions.

INTRODUCTION

Empirical studies of graphics find little support for any general superiority of graphical representations. Instead, graphic displays seem to vary in usefulness depending on the task involved. Twenty-nine studies (Jarvenpaa and Dickson, 1988) found graphics to be more useful than tabular presentations for some tasks, but less useful for others. These results are consistent with the theoretical analysis of Larkin and Simon (1987) that a display (graphic or otherwise) is a data structure. Its utility depends on the nature of the task it supports and the nature of the procedures employed by the human implementer to perform the task. When procedures and data structures match well, there is better cognitive efficiency than when they do not.

Larkin and Simon (1987) suggest that the following forms of cognitive efficiency are offered by good graphical displays.

Substituting Visual Operators: Graphical displays often allow users to substitute less demanding visual operators in place of more complex logical operators. Visual operators (e.g., distance and color comparisons, spatial coincidence judgments) can often give users the same information as more complex non-visual operators. This advantage arises when a display represents explicitly information that is only implicit (or computable) in an alternate representation.

Reducing Search: Effective graphical displays often arrange information so as to reduce the number of items the user must look at in order to find something useful, or they group into one location information required to draw a particular inference. Graphical techniques like shading and spatial arrangement can help guide the eye to relevant information or past irrelevant information.

This paper describes BOZ, a computer-implemented algorithm for designing graphical displays (Casner, 1989). BOZ (described in the second section) systematically exploits the hypothesized advantages of graphical displays, substituting visual operators for logical ones, and constraining the grouping of related information. BOZ analyzes a formal description of the operators required to execute a task and searches a catalog of visual operators to find visual operators that can serve as substitutes for the logical operators. BOZ then proposes graphic displays that support performance of these operators. A single task description typically gives rise to many graphic displays, each supporting different substitutions of visual for logical operators. The next section describes four alternative graphical displays proposed by BOZ to support the task of finding an airline reservation satisfying time and cost constraints. For each of the four graphics, we hypothesize search and information-generation procedures using the different display-supported operators. Simulations of these procedures count the number of times each operator executes for each procedure and graphic. The final section describes an experiment in which participants used the four BOZ-designed graphics. Comparisons of participants' response times with the operator counts support two mechanisms through which these graphics improve cognitive efficiency: (1) substituting visual operators for logical ones, and (2) reducing search by using visual cues to ignore items.

BOZ: DESIGNING EFFECTIVE VISUAL DATA STRUCTURES AND PROCEDURES

A Logical Operator Description Language. BOZ begins with a description of the logical operators (LOPs) required to perform a task. Logical operators are general information-

processing activities independent of a particular representation. For example, the following LOP (`findLayover`) describes finding the layover between two connecting airline flights. It takes two flights as arguments and returns the layover time—the difference between the arrival time of `flightA` and the departure time `flightB`.

```
(LOP findLayover (flightA flightB)
  (DIFFERENCE
   (findDeparture flightB) (findArrival flightA)))
```

A Catalog of Visual Operators. BOZ contains a catalog of visual operators that describe information-processing activities that occur within the context of a graphical display. Visual (or perceptual) operators (POPs) include spatial position and coincidence judgments, interval and distance judgements, comparisons of color, shape, size, slope, length, height, width, etc. POPs are encoded using the same formalism as LOPs. For example, the operator for estimating horizontal distance between two graphical objects is:

```
(POP findHorzDistance (objA objB)
  (DIFFERENCE (findHorzPos objA) (findHorzPos objB)))
```

Matching Logical to Visual Operators. A matching algorithm considers each logical operator in a task description and searches the catalog of visual operators for substitutes. A visual operator qualifies as a substitute if renaming can map the visual operator into the logical operator. For example, `findHorzDistance` and `findLayover` are equivalent because they both compute a difference between two numbers (end times of flights and horizontal locations). Although not discussed here, if no single visual operator matches a LOP, BOZ attempts to match it using two or more visual operators and a set of combination, composition, and repetition rules.

Visually Structuring Related Data. For each proposed substitution of a visual for a logical operator, a data structuring algorithm assesses the information required to perform the operator and tries to ensure that this information is presented in the same spatial locality and in a form that supports easy perceptual performance of that visual operator. For example, if `findHorzDistance` replaces `findLayover`, then the data structuring algorithm requires that: (1) all times are encoded along the same axis, allowing a human to substitute estimating horizontal distance between two objects for the logical operator of subtracting their coordinates; and (2) all time information about a flight is encoded using the same graphical object.

EXAMPLE: GRAPHICAL DISPLAYS FOR AIRLINE RESERVATIONS

We used BOZ to design a set of graphical displays to support the following airline reservation task that manipulates information about flights, their origins and destinations, departure and arrival times, and costs.

Find a pair of connecting flights that travel from Pittsburgh to Mexico City. You are free to choose any intermediate city as long as the layover in that city is no more than four hours. Both flights that you choose must be available. The combined cost of the flights cannot exceed \$500.

The task description given to BOZ contained the following logical operators:

findFlight(origin: city1, destination: city2) Sequentially searches a list for a flight with origin and destination equal to city1 and city2 (one of the cities may be left unspecified). Returns the first flight meeting this criterion, together with the name of any unspecified city.

checkAvailability(flight) Returns true if a flight has seats available.

checkLayover(flightA, flightB) Returns true if the layover between two flights is acceptable (non-negative and less than 4 hours).

checkCost(flightA, flightB) Returns true if the cost of the two flights is acceptable (less than \$500).

Figure 1 shows four of the displays produced by BOZ from these logical operators. We consider these displays in turn, describing how BOZ created them, and correspondingly their hypothesized advantages to a user.

DISPLAY 1: A CONVENTIONAL AIRLINE SCHEDULE

In substituting visual for logical operators, BOZ can select operators for finding and interpreting text. Therefore, among BOZ's representations is Display 1 (Figure 1), a tabular presentation that supports substituting the following visual operators.

findFlight(origin: city1, destination: city2) searches the rows of the table stopping at a row that has the specified endpoint(s), and returning the flight listed in that row.

readAvailability(flight) returns true if second column reads "ok"; else false

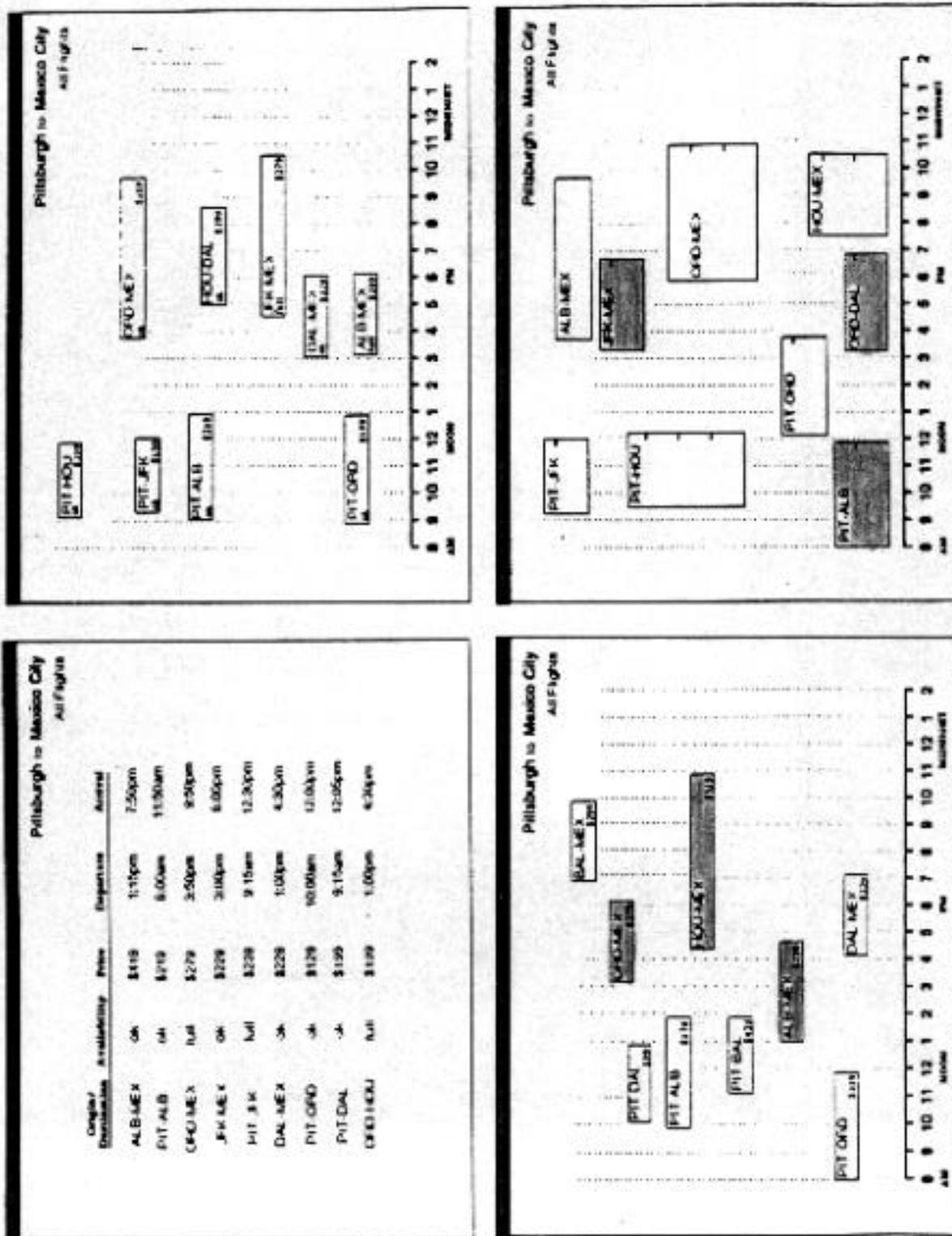


Figure 1: Four BOZ-based displays for the reservation task. (1) A table. (2) Horizontal distance encodes time. (3) Shading encodes availability. (4) Height encodes cost.

subtractTimes(flight1, flight2) finds departure time in the flight2 row (column 4) and arrival time in the flight1 row (column 5); subtracts arrival time from departure time; returns true if greater than zero and less than 4 hours; else false

addCosts(flight1, flight2) finds cost in flight2 row (column 3) and cost in flight1 row (column 3); adds the two; returns true if less than \$500; else false

We define the following search procedure (rowSearch) using the four operators and considering flights sequentially in the order they appear in the rows of the table. It exploits the row and column indexing of information, the only spatial structure available in Display 1.

```
procedure rowSearch
  repeat
    findFlight(origin: pit; destination: any);
      returns flight1, city1.
    if readAvailability(flight1)
      then: findFlight(origin: city1,
destination: mex); returns flight2.
      if readAvailability(flight2)
        then: if subtractTimes(flight1,
flight2)
          then: if addCosts(flight1,
flight2) then:
            report answer
  until answer found
```

DISPLAY 2: HORIZONTAL DISTANCE ENCODES TIMES

Display 2 substitutes the visual findHorzDistance operator for the logical checkLayover operator. If two connecting flights have ends within four units of each other, then the layover is less than four hours. As required by the data structuring algorithm, all times are encoded as horizontal positions, and the two times associated with one flight are encoded by the same graphical object, i.e., a box.

Display 2 also supports two variations of the rowSearch procedure. rightOfSearch is the same as rowSearch, but omits consideration of flight boxes

that are not to the right of the end of the current flight box. `rightOfSearch` thus prunes search by eliminating automatically flights leaving before the arrival of `flight1`. `closeSearch`, users first consider those pairs of flights that not overlapping, but are closest together (have the shortest layovers). `closeSearch` thus prunes search by considering first flight pairs most likely to meet the layover criterion.

DISPLAY 3: SHADING ENCODES AVAILABILITY

Display 3 adds to display 2 support for the visual `judgeShaded` operator in place of the `checkAvailability` operator. Additionally, Display 3 lets users prune any search procedure by skipping all shaded flight boxes. These procedures are indicated by `rowSearchU`, `rightOfSearchU`, and `closeSearchU`, where the final "U" indicates searching only unshaded boxes.

DISPLAY 4: HEIGHT ENCODES COST

Display 4 adds to display 3 support for the visual `judgeHeights` for the `checkCost` operator. A user can judge whether the combined heights of two flight boxes is greater than 5 (\$500), instead of adding the numerical costs of the two flights. Display 4 also supports pruned search procedures (`cheapSearch` and `cheapSearchU`) in which the user considers first the cheapest (least tall) flight boxes, thereby making it more likely to satisfy the cost constraint early in search.

DISPLAY SUMMARY

Operators. Display 1 (the table) supports only the arithmetic and reading operators listed at the left in Table 1(a). Each other display, compared to the previous one supports substitution of one additional visual operator for these read and compute operators. Table 1(a) lists the operator substitutions and the displays in which each is available.

Search. Table 1(b) shows the eight search procedures, four standard strategies, each with a variant involving skipping shaded boxes indicating filled flights. With each is listed the displays for which it can be applied. The central search procedure is `rowSearch`, possible for all displays (see Table 1(b)). Displays 2 - 4 (in which horizontal distance encodes time) allow a user to short-cut `rowSearch`, when finding a connecting flight, by skipping rows unless the flight box begins to the

Table 1(a): Operator substitutions.

<i>Table Operators</i>	<i>Other Display Operators</i>	
subtractTimes	findHorzDistance	2, 3, 4
readAvailability	checkShaded	3, 4
addCosts	judgeHeights	4

Table 1(b): Three search strategy groups.

	<i>Standard</i>	<i>Ignore unshaded</i>
rowSearch	1, 2, 3, 4	3, 4
rightOfSearch	2, 3, 4	3, 4
closeSearch	2, 3, 4	3, 4
cheapSearch	4	4

right of the initial flight box. This `rightOfSearch` procedure thus produces a search sequence that is a consistently ordered subset of items considered by `rowSearch`. These two procedures therefore yield closely related search results. Displays 2-4 also make possible a form of best-first search, by considering first connecting flight boxes with left ends closest to (but right of) the initial flight box. This `closeSearch` algorithm (compared with `rowSearch` and its variant `rightOfSearch`) produces a search sequence with a different ordering of items. Similarly, Display 4, in which box height encodes cost, supports a best-first search with respect to cost (`cheapSearch`), that yields a search sequence different from that of either `rowSearch` or `closeSearch`.

Displays 3 and 4, in which shading encodes availability, support a search variant in which shaded boxes are skipped. Each `-SearchU` variant produces a consistently ordered subset of the search sequence produced by the corresponding search without use of shading.

To compare search procedures concretely, we used 40 displays, ten instances of each type. A LISP simulation of each search procedure counted the number of search steps for each example. We computed the correlation between the number of search steps for a display for each pair of search strategies. Only pairs within one group in Table 1(b) had non-negligible correlations. Consider first a procedure and its `-U` variant in which unshaded boxes are skipped. In the number pair (b, R^2) b is the regression coefficient for the number of search steps with the `-U` strategy on the number of steps with the non-`U` strategy. For three of the pairs, these numbers are: `rowSearch` (.737,.703)

`rightOfSearch` (.502,.646), and `closeSearch` (.741,.446) [based on 30 cases]. Thus skipping unshaded boxes cuts search consistently for each strategy by amounts from 70% to about 50%. For non -U procedure pairs, non-negligible correlations between number of search steps occurred only for `rowSearch` and `rightOfSearch` ($b=.66$, $R^2=.703$). The results of this simulation thus verify the grouping of search procedures in Table 1(b).

EMPIRICAL TEST OF DESIGN EFFECTIVENESS

METHOD

Participants. Eight employees of the Learning Research and Development Center at the University of Pittsburgh. One participant's data is currently missing from the analysis.

Materials. There were a total of 40 problems, ten instances of each of the four displays. Examples of each of the four displays are shown in Figure 1.

Apparatus. Displays were presented as 9 x 12 inch screen images on a Xerox 1186 computer. Response times were computed using the system clock when the mouse was clicked.

Procedure. Subjects performed the reservations task forty times, ten times using each display. To counterbalance learning and practice, eight orders (one for each participant) of display presentation were used (1234, 2341, 3421, 2341, 4321, 3214, 2143, 1432). At the start of the experiment, all the visual operators were explained. Participants were shown the `rowSearch` procedure but were told that they could follow any strategy they wished. Their task was to find a flight that satisfied the criteria (not necessarily the flight that minimized any measure). There was one practice trial with each display version. Participants were told not to guess, to work as quickly as possible but not to compromise accuracy, and that they could rest between any two graphics. Time to complete the experiment was typically 40 minutes.

EMPIRICAL PREDICTIONS

Global Efficiency. Each graphic supports the advantages of the previous one, as well as the ones it introduces. Therefore the first prediction is that cognitive efficiency should be linearly ordered as in Figure 1 with the conventional table worst and Display 4 best.

Decrease in operator times. For every combination of display and search procedure, we can count the number of times each operator is executed. If response times are expressed as a function of the number of executions of each operator, a regression analysis yields estimates of the times associated with each operator. If substituting visual operators for

reading and computing improves efficiency, then the times associated with the operators (`checkAvailability`, `checkLayover`, and `checkCost`) should be smaller for graphics that support substitution of visual operators.

RESULTS AND DISCUSSION

Global Efficiency

The mean response times for each display (excluding five times differing by more than three standard deviations from the problem mean and the 3 to 6 erroneous responses for each graphic) are: Table: 19.3 (8.4); Horizontal distance encodes times: 10.1 (4.7); Shading encodes availability: 7.2 (2.7); Height encodes cost: 7.4 (2.4)

Graphic version had a highly significant effect on response time ($F(3, 239) = 52.719, p < .0001$), and also on the variance of response time ($F(3, 24) = 18.649, p < .0001$). Fischer's PLSD for pairwise comparison indicates no significant difference between version 3 and 4 and differences significant at the .05 level between other version pairs for both mean response times and standard errors of the mean. Displays 3 and 4 produce both the lowest response times and the least variable performance. Displays 2 and 1 each in turn produce significantly higher response times and greater variability.

The visual operators supported by Displays 2 and 3 thus had the predicted effect on global efficiency. But allowing users to perform `judgeHeights` (instead of `addCosts`) produced no observable effect. This should perhaps not surprise us since it is the one visual operator that requires integrating quantitative estimates from two different locations.

Decrease in Operator Times.

Preliminary comparisons of the three procedure groups (`rowSearch`, `closeSearch`, and `cheapSearch`) with the subject response times for each of the four displays suggest that only the `rowSearch` procedures provide reasonable fits to the data. Thus it seems that, with the practice available, subjects did not adopt the best-first strategies, but used the `rowSearch` group shown in Table I(b).

Based on these preliminary results, we used the following process to assess the effect of substitution of visual for logical operators: We assumed for each graphic the most efficient `rowSearch` procedure supported by that graphic, i.e., `rowSearch` for the table, `rightOfSearch` for display 2 (with flight boxes), and `rightOfSearchU` for Displays 3 and 4 (with shaded boxes). We considered two alternative operators for assessing layover and cost. The `subtractTimes` and `addCosts` operators correspond to subtracting or adding numbers. These operators were assumed for Displays that did not support alternative procedures (the table for `subtractTimes`, and

Displays 1, 2, and 3 for `addCosts`). For the remaining displays, we assumed use of the more efficient visual operators `findHorzDistance` and `judgeHeights`. We assumed that the time for one search step was the same in all graphics (although the number of such steps varied with the search procedure supported).

Using these assumptions we computed for each of the 40 graphic exemplars the number of search steps and the number of cost and layover computations. A regression of response times on these numbers produced a well-fitting statistical model with $F(4, 238)=73.108$, $p = .0001$, $R^2 = .48$. Removing from the model the counts for either search or checking layovers dramatically reduced the fit. In contrast, removing the counts for checking costs had no effect on the fit. This model yielded the following parameter estimates:

One search step requires $330 + 35$ milliseconds.

The `findHorzDistance` operator is $2 + .25$ seconds faster than the `subtractTimes` operator.

The `judgeHeights` operator is negligibly ($100 + 300$ milliseconds) slower than the `addCosts` operator.

These results are consistent with the global time differences given above. The reduced performance time with successive graphics arises for two reasons. First, attending only to boxes to the right of the current box and to unshaded boxes reduces the number of items that must be searched. Second, substitution of `findHorzDistance` for `subtractTimes` produces a substantial saving in time. In contrast `judgeHeights`, which requires integrating visual information from two separate locations, provides no such advantage. These two effects are sufficient to account for the response time differences between Displays 1-3, and for the lack of difference between Displays 3 and 4).

SUMMARY

BOZ is a computer algorithm that starts with the logical operators required to perform a task and designs graphic displays supporting substitution of visual operators for logical ones, and pruning of search through visual cues. In an initial experimental test, four BOZ-designed graphics each included one additional visual operator and corresponding opportunities for pruning search, by using visual cues to ignore certain items or by restructuring search to consider more promising items earlier. Analysis of subjects' response times indicate strongly that two out of these three enhancements dramatically and significantly improved response times to the task. The unhelpful enhancement

required integration of information from two separate locations. More detailed analyses suggest that these improvements were due to operator substitution and using visual cues to omit items from search, but not due to restructuring search.

Importantly BOZ is a synthesis algorithm. It starts with an abstract task description, and produces a collection of graphics which should, on the basis of information-processing principles, reduce human processing effort for the task. This work is therefore a start on the practically important effort of putting cognitive science to work in practical applications.

REFERENCES

Casner, S.M. (in press) A task-analytic approach to the automated design of graphic presentations. To appear in *ACM Transactions on Graphics*.

Jarvenpaa, S.L. & Dickson, G.W. Graphics and managerial decision making: Research Based Guidelines. *Communications of the ACM*. 31(6) 764-774.

Larkin, J.H. & Simon, H.A. (1987). Why a diagram is (sometimes) worth 10,000 words. *Cognitive Science*, 11(1), 65-100.

ACKNOWLEDGEMENTS

This work is supported Office of Naval Research, University Research Initiative Contract Number N00014-8~K-0678 and by a grant from the James S. McDonnell Foundation to the second author. William Oliver and Stellan Ohlsson provided helpful comments.