Human Performance on Hard Non-Euclidean Graph Problems: Vertex Cover

Sarah Carruthers,[1] *Michael E. J. Masson,*[2] *and Ulrike Stege*[3]

Abstract:

Recent studies on a computationally hard visual optimization problem, the Traveling Salesperson Problem (TSP), indicate that humans are capable of finding close to optimal solutions in near-linear time. The current study is a preliminary step in investigating human performance on another hard problem, the Minimum Vertex Cover Problem, in which solvers attempt to find a smallest set of vertices that ensures that every edge in an undirected graph is incident with at least one of the selected vertices. We identify appropriate measures of performance, examine features of problem instances that impact performance, and describe strategies typically employed by participants to solve instances of the Vertex Cover problem.

Keywords:

computational complexity, vertex cover, human performance

[1, 2, 3] University of Victoria

http://dx.doi.org/10.7771/1932-6246.1142

Introduction

In the past two decades, there has been an increased interest among cognitive scientists in investigating human performance on instances of computationally hard visual problems[1]. For the purpose of this study, we define computationally hard problems as those that have been shown to be NP-hard according to a formal definition from complexity theory. These are problems that likely cannot be solved by polynomial-time algorithms[2] (Garey & Johnson, 1979). Studies of human performance on one such problem, the Euclidean Traveling Salesperson Problem (E-TSP), have indicated that humans may be able to find close to optimal solutions in near linear-time (Chronicle, MacGregor, Ormerod, & Burr, 2006; Dry, Lee, Vickers, & Hughes, 2006; Graham, Joshi, & Pizlo, 2000; MacGregor, Chronicle, & Ormerod, 2004; MacGregor, Chronicle, & Ormerod, 2006; Kong & Schunn, 2007; Tak, Plaisier, & van Rooij, 2008). This seems to be in keeping with the best known approximation algorithms[3] that can also achieve near optimal results in close to linear time, as well as fast heuristics (Arora, 1997). In the computer science field, a heuristic is an algorithm that has no guarantee of correctness or quality of bounds for a given computational problem (Cormen, Leiserson, Rivest, & Stein, 2003). This definition differs slightly from that in the cognitive science domain given by Gigerenzer and Gaissmaier (2011), which states that "[h]euristics are efficient cognitive processes, conscious or unconscious, that ignore part of the information" (p. 451). Alternatively, according to Todd and Gigerenzer (2000):

> Up to about 1970, "heuristics" referred to useful, even indispensable cognitive processes for solving problems that cannot be handled by logic and probability theory alone. After 1970, a second meaning of "heuristics" emerged in the fields of psychology and decision-making research: limited decision-making methods that people often misapply to situations where logic and probability theory should be applied instead. (pp. 738–739)

[1] A *computational problem* is typically a formally described question or task that is to be solved for a given input of a specified format. Computational problems come in different flavours, such as *decision problems*, *search problems* and *optimization problems*.

[2] An *algorithm* is a finite set of well-defined steps that transforms an input into an output. Furthermore, for a given computational problem, in order for an algorithm to be considered correct, it must, on any input, come to a stop with the correct output (Cormen, Leiserson, Rivest, Stein, 2003). For the remainder of the paper when referring to an algorithm, unless otherwise specified, we mean a correct algorithm for the computational problem discussed.

[3] An *approximation algorithm* for a computational problem is a well-defined set of steps that transforms an input into an output that guarantees the optimality of the solution to be within specific bounds of the correct solution (Vazirani, 2003). Typically, the goal when designing an approximation algorithm is that the approximation achieved by the algorithm is optimal up to a (small) constant factor (for example, $\leqq 2\times$optimal). This still holds when there exists more than one optimal solution, as the bound is in terms of the optimality of the solution.

The computer science definition of heuristic stems from the correctness of the result, whereas this cognitive science definition appears to stem from the input that is (not) used. It is not clear whether humans employ strategies that are similar or related to approaches from the computer science literature, namely, optimal algorithms, heuristics or approximation algorithms. It is, however, important to view human performance results in context. Studies of human performance on E-TSP typically use inputs of relatively small size (<100 points), and generally use randomly generated instances. Both of these factors may reduce the complexity of the particular instances of the problems given to participants.

While this paper focuses on human performance on computationally hard graph problems, we recognize the wealth of previous work on other hard problems that do not use graph representations, like the 15-puzzle (or n-puzzle), water jug problems, optimal stopping, and bandit problems to name but a few (e.g., Atwood, Masson, & Polson, 1980; Bellman & Brock, 1960; Lee, 2006; Hemmati & Smith, 2011; Lee, Vickers, & Brown, 1997; Lee, Zhang, Munroe, & Steyvers, 2011; Pizlo & Li, 2005). A study of human performance on non-Euclidean TSP found that solutions were no longer near-optimal (Saalweachter & Pizlo, 2008). This study is a next step in investigating human performance on non-Euclidean computationally hard problems.

Because of the interdisciplinary nature of these studies, we begin with a discussion of the notions of problem solving in the fields of computer science and cognitive psychology. We then introduce the computationally hard Minimum Vertex Cover problem, and discuss the purpose of this study. Finally, after presenting the data from our experiments, we discuss the results of this pilot study, and some directions for future studies.

Solving a Problem

Problem solving is an important part of human thinking, and takes place in diverse situations of everyday life. In this section, we try to clarify possible differences in the meaning of "problem solving" in the fields of computer science and cognitive psychology. Neither of the definitions below is complete or comprehensive. Our goal is to highlight general differences across these disciplines that may, if unobserved, lead to misinterpretation.

Problem Solving in Computer Science

Problem solving is a key aspect of computer science. Computer scientists refer to finding a solution to a computational problem with an algorithm, exemplified in Introduction to Algorithms (Cormen et al., 2003, p. 6), which states "a correct algorithm solves the given computational problem." In this case, the known algorithm is used to go from any given input to its corresponding output. Given a computational problem and any particular input, an algorithm outlines a set of steps that will yield a correct output, as shown in

Figure 1. Depending on the type of problem, there may be more than one correct output since, according to the corresponding measure of correctness, they may be equivalent. For example, in finding the shortest route from point *A* to point *B*, there may be multiple paths of equal (and optimal) length that correctly lead from point *A* to point *B*.

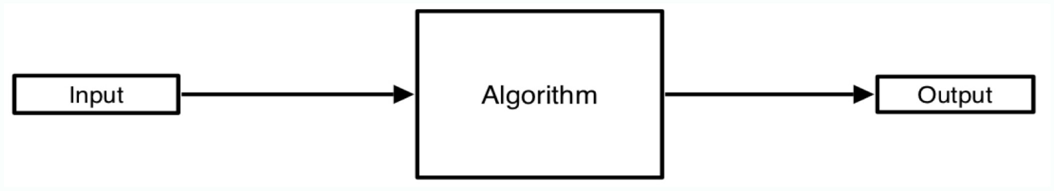

Figure 1. An algorithm solves a given problem.

Problem solving is also used when attempting to identify an algorithm to solve a known problem, as shown in Figure 2. Where it has been shown that there is likely no polynomial time algorithm, computer scientists will search for approximation algorithms, fixed-parameter tractable algorithms[4] or heuristics.

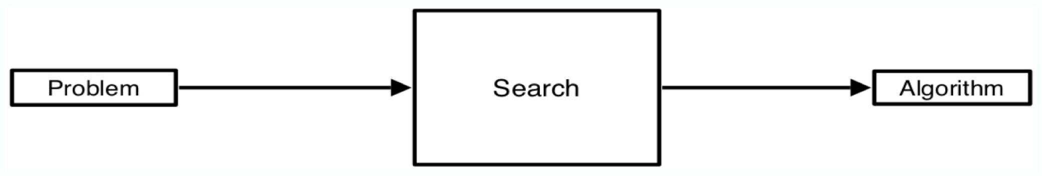

Figure 2. Searching the problem space for an algorithm.

Human Problem Solving

We contrast the previous definition of problem solving to a model of problem solving in *Human Problem Solving* (Newell & Simon, 1972, p. 137): "The problem solver is given a description of the solution . . . and is required to find an alternative process description that will specify how to generate it." This reflects one theory of human problem solving. Given some start state, the problem solver's aim is to reach some specified goal state, and as shown in Figure 3, they reach that goal through a search process (Newell & Simon, 1972). For some problems, this may require only one step, but for complex problems where a path to the goal state is not necessarily clear, the problem solver may identify *sub-goals* between the start state and final goal. In general, we observe that these sub-goals fall into at least two different categories, *correct* and *opportunistic*. Sub-goals are correct when they always lie on a path to a correct solution. Other sub-goals are opportunistic, in that they meet a short-term goal, but do not necessarily lead to a correct final goal state. Participants

[4] A fixed-parameter tractable (FPT) algorithm solves a problem with input of size n and a specified parameter of a fixed value k, in time that is polynomial in terms of n, although it can be exponential or worse in terms of k (Downey & Fellows, 1999). Parameterized problems that are fixed-parameter tractable are members of the complexity class FPT.

may choose an opportunistic sub-goal, for example, because they have incorrectly interpreted the final goal, or because no correct sub-goal can be identified (e.g., due to time constraint, motivation, difficulty). In this case, the application of opportunistic sub-goals is a *heuristic* step. We remark, however, that both correct and opportunistic sub-goals can be carried out either correctly or incorrectly.

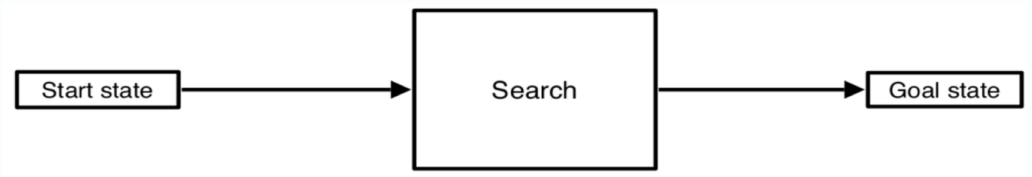

Figure 3. Human problem solving.

The types of problems investigated by Newell and Simon are simpler than the problem presented in this paper. It is important to question whether or not the means-ends analysis model of investigating problem solving still holds for more complex problems like Minimum Vertex Cover. It is of interest in future work to investigate whether other models, for instance Gigerenzer's Adaptive Toolbox, may also be relevant (Gigerenzer & Selten, 2001).

We highlight the similarities between the ideas presented in Figures 2 and 3. In both cases, the individual is searching for a path to the goal. In this sense, human problem solving and problem solving in the field of computer science are similar. However, in computer science, this search is usually more rigorous: in computer science we require a general solution that works for all instances of the problem, whereas humans may not always achieve such a general solution as they are typically asked to solve particular instances of a problem.

The Vertex Cover Problem

Given an undirected graph *G* with edges *E*, and vertices *V*, a *vertex cover* is a subset *U* of *V* such that every edge in *E* is incident to a vertex in *U*. A *minimum* vertex cover of *G* is one where the number of vertices in *U* is minimized.

For example, Figure 4 shows an undirected graph with five vertices (*A, B, C, D*, and *E*) and six edges ((*A, D*), (*E, D*), (*D, B*), (*D, C*), (*B, C*), and (*E, C*)). The vertices {*E, D, B*} constitute a vertex cover. The vertices {*D, C*} are a minimum vertex cover.

In this article we concern ourselves only with undirected graphs. Unlike directed graphs, in undirected graphs, no directions are associated with the edges. The *degree of a vertex* is the number of edges that are incident to it. Vertex *D* is of degree four, whereas vertex *A* is of degree one.

There are many real world applications of the Vertex Cover problem (VC). For example, the Minimum Vertex Cover (MVC) problem comes into play in scheduling problems. A scheduling problem can be modeled as a graph, where the vertices represent tasks or times, and an edge between vertices means that a conflict exists between those times or tasks. Finding the minimum number of tasks that needs to be removed in order to resolve all conflicts is equivalent to finding a minimum vertex cover.

The *Minimum Vertex Cover* problem, namely finding a minimum vertex cover for a given undirected graph, is NP-hard. This means that likely no polynomial time algorithm for the problem exists. However, the parameterized decision version of Minimum Vertex Cover—in which we ask, for a given graph G and a positive integer k, whether there exists a vertex cover that contains at most k vertices—has been shown to be in FPT for parameter k. In this case, there exists, for example, an algorithm that runs in time $O(2^k n)$, that is, it is polynomial in terms of n, and exponential in terms of k (Downey & Fellows, 1972). Though in FPT, the Minimum Vertex Cover problem is also NP-hard.[5]

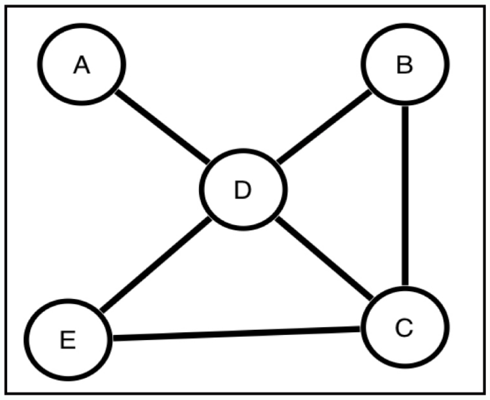

Figure 4. An undirected graph.

For parameterized problems that are shown to be in FPT, there typically exist polynomial-time rules that can be applied to an instance of the problem leading to a reduction in the size of the instance. For Vertex Cover, we propose two of these rules, the Degree 1 rule (D1) and the Degree 2 rule (D2) (see, e.g., Stege, 2000), as possible techniques humans may apply to instances of VC. These rules were inspired by known rules from the FPT literature and have the advantage of being both simple, and optimal. Other possible rules exist as well, which we will discuss briefly in the *Future Lines of Study* section, but for this study we focus on these two only.

[5] We observe a direct connection between the decision version and the optimization version of Vertex Cover: If the decision version of Vertex Cover was not NP-hard but solvable in polynomial time, then the Minimum Vertex Cover Problem also would be solvable in polynomial time as one has only to try the decision version for all possible values of k.

Degree 1 Rule

Where there are one or more vertices of degree one connected to a vertex *v*, it is never wrong to select *v* as part of a minimum vertex cover. For example, in the graph in Figure 4 vertex *D* is such a vertex, and is part of a minimum vertex cover.

Degree 2 Rule

Given a triplet of vertices, (*u*, *v*, *w*), that are connected by the edges (*u*, *v*), (*v*, *w*) and (*u*, *w*), at least two of these vertices must be in a minimum vertex cover. Furthermore, if one of these vertices, say *u*, is of only degree two, then it is optimal to include v and w in the vertex cover. For example, in Figure 4 vertices *B*, *D* and *C* are such a triplet, and vertices *C* and *D* are in the minimum vertex cover.

Both the D1 and D2 rules are a special case of a stronger rule. Although we chose to examine only these two special cases, it is possible that people may be able to recognize other cases of this generalized rule, under certain circumstances. Given a graph *G* with vertices *V*, and edges *E*:

> For any subset *U* of *V* that forms a clique, at least ($|U| - 1$) vertices must be in the minimum vertex cover. A clique is a set of *k* vertices where each vertex is connected by an edge to every other vertex in the clique. The D2 and D1 rules are special cases where $k = 3$ or $k = 2$, respectively.

Purpose of the Study

The main purpose of this study is to identify what effect problem-instance size and layout have on human performance on the Minimum Vertex Cover problem. Like the E-TSP, this problem can easily be presented in the plane. However, it differs in that it is a non-Euclidean problem. A secondary purpose of the study is to determine whether individuals adopt specific techniques while solving instances of the Minimum Vertex Cover problem. These techniques include the D1 and D2 reduction rules identified in the previous section, as well as the opportunistic technique of adding vertices of highest degree to the cover, which in the literature is referred to as the *greedy method* (Cormen et al., 2003). All these techniques can be likened to sub-goals that participants may identify when the path to the goal is not readily apparent, and whether or not participants apply them may give some insight as to sub-goals they identify. Finally, the study aims to investigate the quality of human solutions to instances of the Minimum Vertex Cover problem.

Graph Layout

How does the layout of instances of Vertex Cover relate to human performance? Two manipulations were used in the study: symmetry and planarity of the graphs. We hypothesized that participants' performance would be negatively impacted by instances of the graphs that were depicted with crossings. Crossings were believed to increase problem difficulty as they may impact individuals' ability to discriminate and maintain distinctions between edges and the connectivity of specific vertices. Similarly, it was hypothesized that instances presented in a symmetric or ordered format would be less challenging because participants could use redundancy in the layout to their advantage, for example by replicating successful selections applied to one part of a graph in other, similarly configured parts.

Techniques

Participants, as novices, are asked to solve the given MVC instance—without the knowledge of an a priori algorithm or procedure that will always yield the correct result. That is, given the initial state, they must choose to add vertices to the Vertex Cover until the goal state is achieved. The initial state is a particular instance with no vertices yet in the vertex cover. The goal state, as given in the instructions, is a set of vertices that meets the following criteria: all edges in the graph are incident to at least one vertex in the set (called the vertex cover), and the set includes the fewest number of vertices possible.

Three techniques were suggested that participants might identify to solve instances of the Minimum Vertex Cover problem. Two techniques were chosen based on known reduction rules employed on FPT algorithms solving the parameterized Vertex Cover problem, and one technique was identified based on observation of humans solving instances of Vertex Cover.

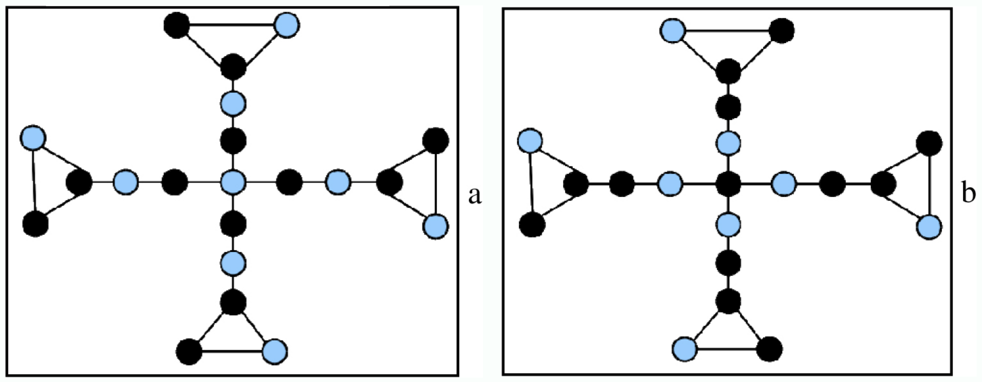

Figure 5. Graph 21.1a with optimal (a) and non-optimal (b) vertex cover in black.

The first two techniques are the D1 and D2 rules described above. The third technique is one that was identified as a candidate based on the hypothesis that a likely sub-goal is to cover as many edges as possible in one step. Choosing high-degree vertices that are not yet included in the vertex cover satisfies this goal. It is important to note, however, that this technique does not always lead to an optimal solution. This generally non-optimal technique will be referred to as *opportunistic*. Graph 21.1a in Figures 5a and 5b is resistant to the opportunistic approach, and is shown in Figure 5a with a minimum vertex cover of size 12. The highest degree vertex in the graph is not included in the cover. In Figure 5b, this graph is depicted with a sub-optimal minimum vertex cover of size 13, with the highest degree vertex included in the cover. Conversely, the minimum vertex cover of Graph 17a, in Figure 6, includes the highest degree vertex at the centre. This graph, unlike Graph 21.1a is not resistant to the opportunistic technique.

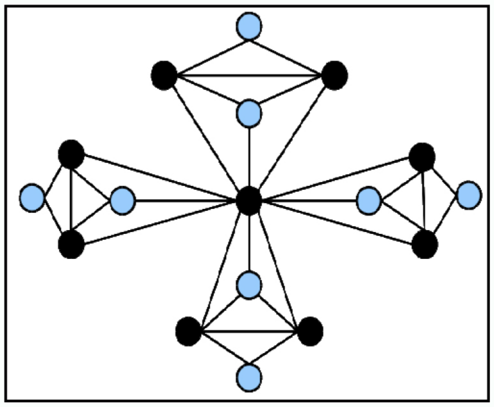

Figure 6. Graph 17 with optimal vertex cover in black.

When the path to the goal state may not be apparent for a given initial state, novices are known to apply the general search strategy *means-ends analysis* (Newell & Simon, 1972). Under this strategy, sub-goals are chosen based on eliminating differences between the current state and the goal state. For the Minimum Vertex Cover problem, these differences may be dependent on how the subject interprets the problem. Participants may see the task of "covering all edges" as the main goal, and may interpret the task as optimally choosing a vertex to add to the cover such that the number of edges covered by this vertex is maximized. Participants may also use a more global strategy to identify vertices that are best added to the cover, independent of the magnitude of their degree. These sub-goals may be related to specific techniques that individuals adopt and they may not be mutually exclusive. It is hypothesized that the *opportunistic* technique may be an indication of the application of an *edge covering* sub-goal, whereas the D1 and D2 rules may be indicative of a *vertex adding* sub-goal.

As part of our goal to examine sub-goals that problem solvers develop as they gain experience with Vertex Cover instances, we randomly assigned participants to two groups that differed with respect to instance order. Group A received an instance set with instances resistant to the opportunistic technique in the first half of the set. We anticipated that this group might be more likely to be cautious when considering the opportunistic technique as they proceeded through the latter part of the set. Group B received the instance set reversed: instances resistant to the opportunistic techniques were only in the second half of the set of instances. We were also interested whether early experience with instances resistant to the opportunistic approach would affect performance on other measures of problem solving optimization.

Quality of Solutions

In studies of human performance on computationally hard optimization problems, the quality of solutions can be measured in a number of ways, including optimality, speed, and classes of solutions that exhibit different qualities. For the purpose of this study, quality was measured by forming a ratio between the number of vertices included in the cover and the minimum number of vertices actually required to complete a cover (that is, the optimal solution). Further, we investigate how this measure of performance is related to instance size, layout, and order of problem set presentation.

Methodology

Participants

Fourteen University of Victoria students participated in the study to earn extra credit in an undergraduate psychology course. Each participant was given a set of graphs on which they were asked to find a minimum vertex cover. Participants were randomly assigned to one of two groups, Group A (8 participants) and Group B (6 participants). The only difference between the groups was the order in which the instances were given, and all participants completed the same 24 instances.

Instrument

Six different graphs, with 8, 16, 17, 21, 21 and 28 vertices, were created digitally by hand. Four different versions of each graph were created, resulting a total of 24 graphs. Each graph was presented on a separate page of paper, and participants were given a pen to write with. Graphs are named according to the number of vertices they contain. Note that two graphs with 21 vertices are named 21 and 21.1 for clarity. The properties of each of these graphs are outlined in Table 1. Note that *n* refers to the number of vertices in the graph, and *e* refers to the number of edges. Two graphs were generated with D1-candidate vertices (Graphs 16 and 21). These graphs are shown in Figures 8a and 7a. The remaining graphs were generated with D2 candidate vertices (graphs 8, 17, 21.1 and 28) shown in Figures 7b, 7c, 7d, and 7e. One of these graphs (21.1) was generated such that including the highest degree vertex in the vertex cover would also guarantee a sub-optimal solution. The four versions of this graph are termed resistant to the opportunistic technique.

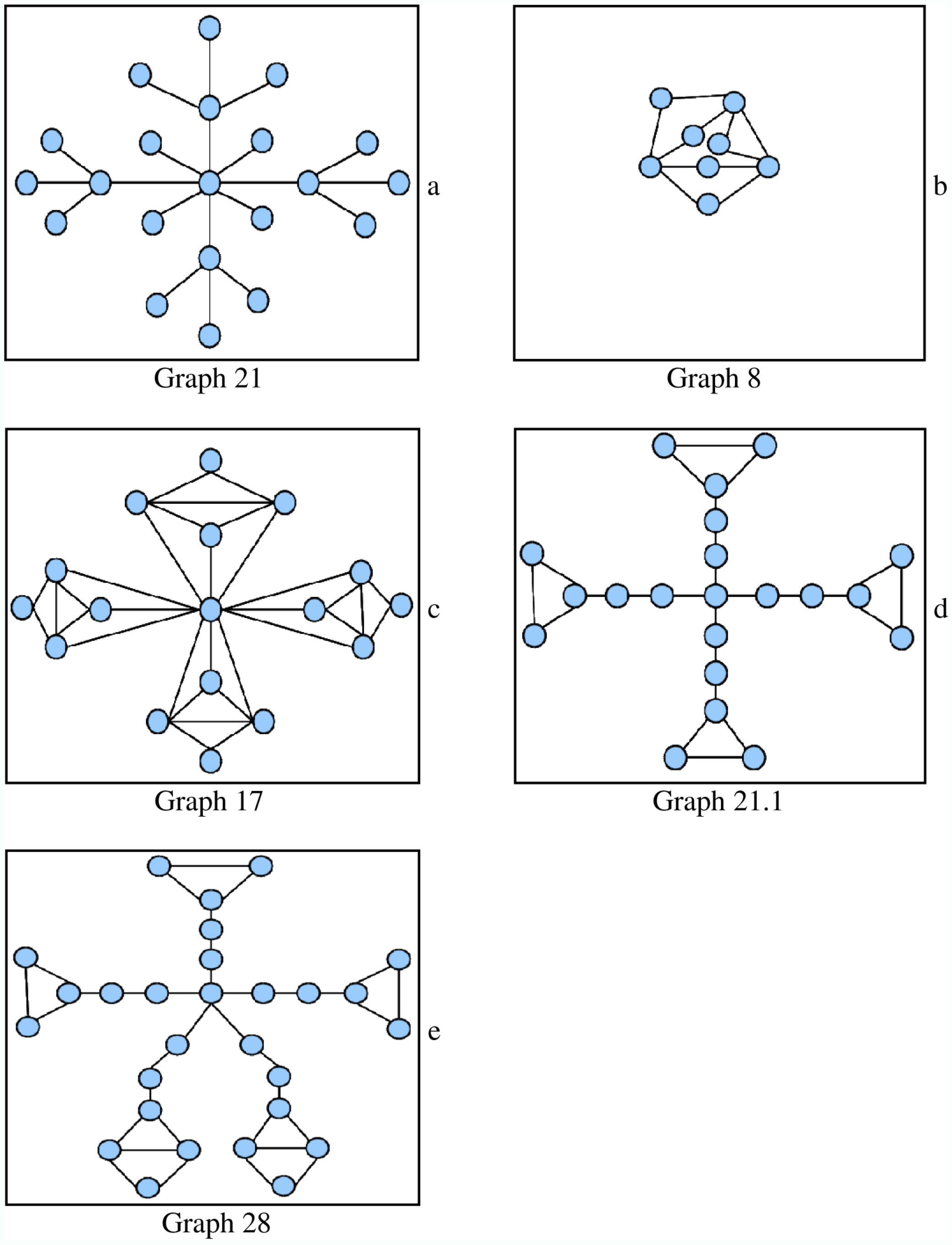

Figure 7. Graphs 21, 8, 21.1, 17 and 28.

The four versions of each graph were generated according to a factorial manipulation of two variables: planarity (plane and nonplane) and neatness (neat and nonneat). Plane graphs were arranged so that edges did not cross one another. Neat graphs were laid out so that the vertices were positioned, as much as possible, in an orderly and symmetrical layout. The four variants of Graph 16 are shown in Figure 8 and described in Table 2.

ID	*n*	*e*	Optimal VC	Opportunistic
8	8	11	3	yes
16	16	21	8	yes
17	17	32	9	yes
21	21	20	5	yes
21.1	21	24	12	no
28	28	34	16	yes

Table 1. Properties of the six main graphs.

ID	Neat	Plane
16a	yes	yes
16b	yes	no
16c	no	yes
16d	no	No

Table 2. Four variants of Graph 16.

These 24 graphs were divided into two sets: Set X and Set Y. Set X contained all four variants of graph 21.1 that were resistant to the opportunistic technique and Set Y contained no graphs that were resistant to that technique. Within each set, a single fixed random order for presentation of the 12 graphs was established and used for all participants. The only variation in presentation order for the graphs was that Group A first received Set X followed by Set Y, and Group B received Set Y followed by Set X. Thus, Group A experienced instances resistant to the opportunistic technique in the first half of their instance set, whereas Group B did not encounter any until the second half of the instance set. The specific order of instances in Set X and Set Y are presented in Table 3.

X	21.1a	16a	28a	21.1b	21.1d	28d	16d	28b	17c	28c	14c	21.1c
Y	8a	21c	17a	8b	21b	16b	17d	8c	21d	17b	21a	8d

Table 3. Graph order in sets X and Y.

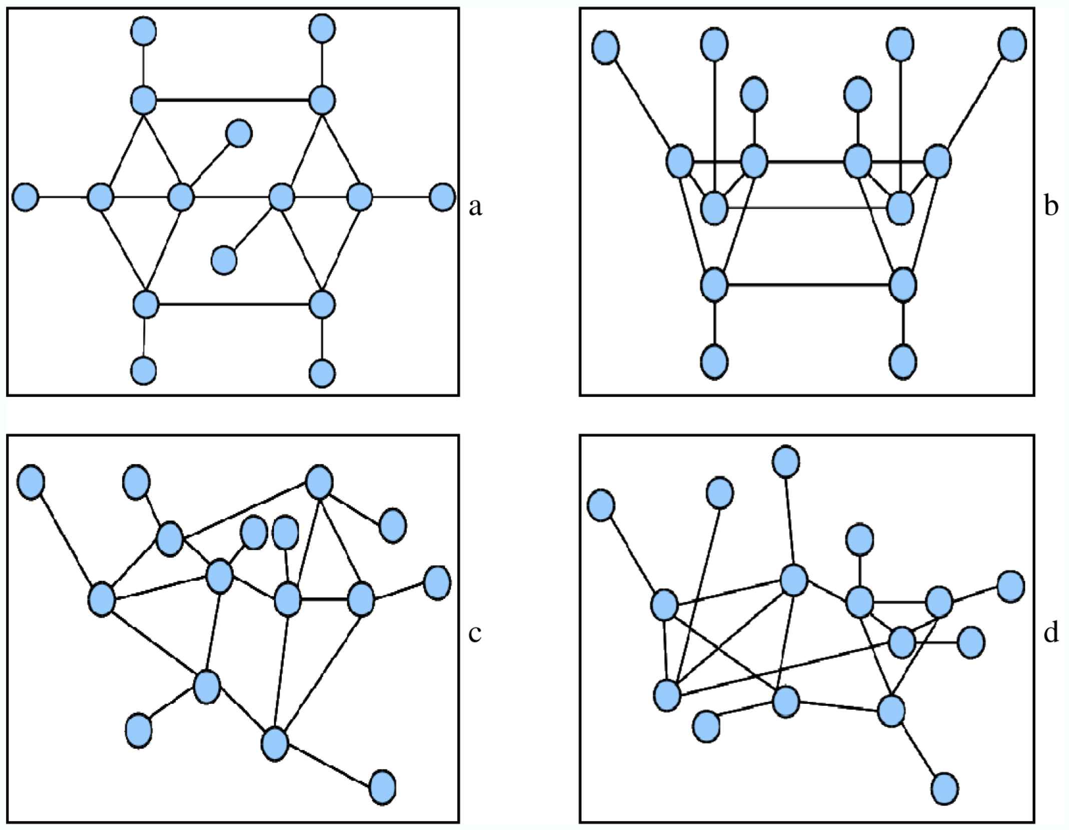

Figure 8. Four variants of Graph 16.

Procedure

In order to make the Vertex Cover problem relevant, it was described to participants as *The Guard Problem* as follows: *Find the fewest guards necessary to protect the art in the corridors of the art gallery. On each page is a drawing of the layout of an art gallery. The circles represent guard-posts, and lines represent corridors containing priceless art.* That is, participants were asked to place the fewest guards necessary to protect all the corridors. This problem description is equivalent to the Minimum Vertex Cover problem. Participants were asked to number the guards consecutively as they placed them, starting at one. They were allowed to indicate that they changed their mind about a choice by crossing out a previously numbered vertex. Participants were asked to not reuse crossed out numbers but rather to continue numbering where they had left off. Participants were allowed to make notes or other marks on the instances as they worked out their solution. Three trial instances were presented to each participant to ensure that he or she understood the problem described. A time limit of two minutes per instance was given to the participants, but not enforced. All participants were given one hour to complete all the instances (including instructions and trial instances). No feedback was given between instances.

Analysis

Participants' responses on each instance were coded, vertex by vertex, in the order that vertices were chosen for the cover. Each vertex was coded based on the state of the graph at the time that it was added to the cover as a candidate of each of the three techniques (D1 rule, D2 rule, or opportunistic). A vertex was deemed a candidate for the D1 technique if it was connected to vertices that were of degree one. A vertex was deemed a candidate for the D2 technique if it was in a triangle of vertices, of degree greater than two, and there was at least one other vertex in the triangle with degree two. Finally, each vertex was coded for greediness based on its degree relative to all other vertices in the graph. A score of 0 was given to a vertex with degree as high or higher than all other vertices. A score of 1 was given to a vertex with degree one less than a vertex of score 0, a score of 2 was given to a vertex with degree one less than a vertex with score 1, and so forth.

All solutions that included a valid vertex cover were included in the analysis. Solutions were deemed to have a valid vertex cover if the vertices selected by the participant covered all edges. A solution was not considered valid if it was impossible to interpret which vertices were in the cover, or for which the order in which the vertices were added was not clear (for example if a number was repeated).

Results

In total, 40 solutions (11.9%) were excluded from our analysis because they did not contain a valid vertex cover. For each participant, the percentage of excluded solutions was calculated for each graph type. Neither graph type nor group significantly influenced the percentage of exclusions.

Graph Layout and Type

To investigate possible variation in the quality of participant solution as a function of graph type, the number of covered vertices and the optimal number were used to compute the percent above optimal (PAO) for each solution. Then, for each participant, the mean PAO was calculated for each of the four graph types: neat-plane, neat-nonplane, nonneat-plane and nonneat-nonplane. These results are shown in Figure 9a. A repeated-measures analysis of variance was performed on these data with group (A or B) as a between-subjects factor. Strong effects were seen for neatness ($F = 8.41$, $p < 0.02$), and neatness also interacted with group ($F = 5.68$, $p < 0.05$). Surprisingly, performance was better overall on nonneat instances than on neat ones. This is in contradiction to our hypothesis that participants' performance would be positively impacted by neatness, as they could use redundancy to their advantage.

Mean PAO was also separately calculated for opportunistic and non-opportunistic graphs for each participant. These results are shown in Figure 9b. There was no significant difference between mean PAO on these two types of graph.

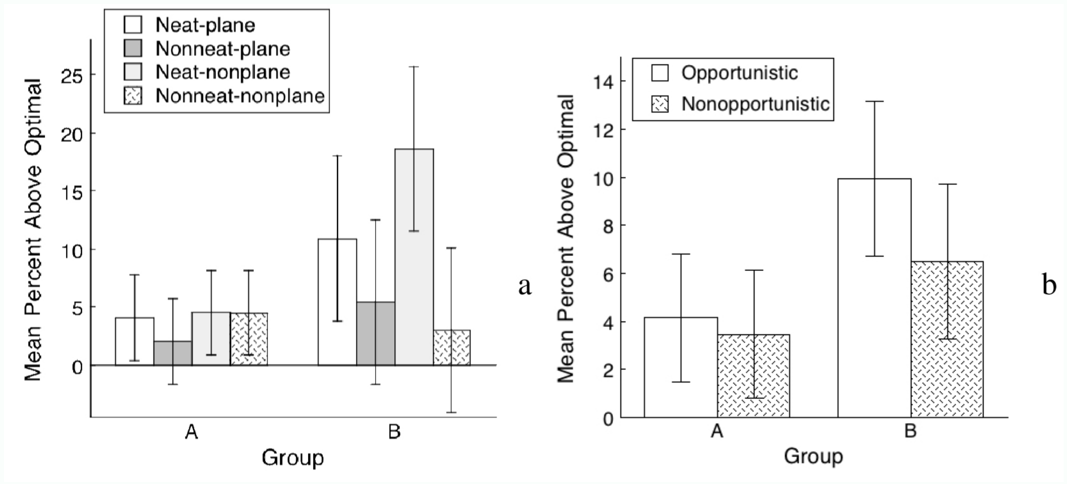

Figure 9. Mean percent above optimal coverage as a function of group and type of graph. Error bars are 95% within-subject confidence intervals which allow meaningful comparisons among the types of graphs.

Techniques

D1-Candidate Vertices.

In order to investigate whether or not participants were sensitive to D1-candidate vertices, their solutions were coded and analyzed as follows. For each of the six graphs, the first vertex added to the cover by participants was coded as either a candidate for the D1 technique or not. The results are presented in Table 4. The results for the remaining vertices are not considered here because adding a vertex to the cover disrupts the state (that is, the vertices and edges that remain to be considered) of the graph, so the type is no longer constant across all instances and all participants. On Graphs 17, 21.1 and 28 the number of D1-candidate choices was zero, as there were no D1 candidate choices at the onset. Participants, however, clearly tended to choose D1-candidate vertices on graphs 16 and 21, even though there were many other vertices in these graphs that were not D1-candidates. The number and ratio of D1-Candidate vertices on Graphs 16 and 21 are shown in Table 5. A chi-square test was computed for each of the two instances that contained D1 candidates to provide a statistical test of the preference for D1 vertices. These tests showed that for both instances, the preference for D1 candidates was substantially greater than would be expected if subjects had been selecting the initial vertex at random from among D1 and non-D1 vertices, *ps* < .001. This strong preference for D1 candidates, when available, was present in both Group A and Group B.

Graph	Type	D1-candidate choice	Non D1-candidate choice
16	D1	48	3
21	D1	55	0
8	Not D1	0	55
17	Not D1	0	44
21.1	Not D1	0	46
28	Not D1	0	45

Table 4. D1-candidate choices by graph type.

Graph	D1-candidates	Non D1-candidates	Ratio
16	8	8	0.5
21	5	16	0.24

Table 5. Ratio of D1-candidate vertices.

Graph	Type	D2-candidate choice	Non D2-candidate choice
8	D2	24	31
17	D2	43	1
21.1	D2	18	28
28	D2	11	34
16	Not D2	0	51
21	Not D2	0	55

Table 6. D2-candidate choices by graph type.

Graph	D2-candidates	Non D2-candidates	Ratio
8	2	6	0.25
17	8	9	0.47
21.1	4	17	0.19
28	7	21	0.25

Table 7. Ratio of D2-candidate.

D2-Candidate Vertices.

For each of the six graphs, the first vertex added to the cover by participants was coded as either a candidate for the D2 technique or not, and are presented in Table 6. Participants were prone to identify D2-candidate vertices when they were available for selection, although the effect was not as strong as for D1-candidate vertices. Nevertheless, chi-square tests for instances 8, 17, and 21.1 showed that participants strongly preferred to begin with a D2 candidate, relative to what would be expected by chance, based on the ratio of D2 to non-D2 candidate vertices as shown in Table 7, *ps* < .005. These effects were large across both Group A and Group B. Instance 28, despite containing D2 candidates, did not show this effect.

Opportunistic Choices.

The greediness score for the first vertex added to each cover was calculated relative to the degree of all other vertices in the graph. These scores were based on the degree range of each graph. For example, Graph 14's vertices have a maximum degree of 4, therefore the range for the greediness score for vertices on this graph was 0 (same degree as highest degree vertex) to 3 (3 fewer edges than the highest degree vertex). The score was then inverted, and divided by the maximum to give a scaled relative greediness score. The mean greediness score was calculated for each participant, across opportunistic and non-opportunistic instances. Repeated measures analysis of these variables revealed no significant results.

Instance Size and PAO

Mean PAO was calculated across all participants for each instance size (both in terms of $|E|$ and $|N|$), however no significant correlation between PAO and instance size was found.

Discussion

The goal of this study was to determine what impact graph layout and instance size have on human performance on instances of Minimum Vertex Cover problems. Further we investigated whether or not participants would identify specific techniques while working on instances of Minimum Vertex Cover problems. Lastly, we investigated if the introduction to instances resistant to one of these techniques has an impact on performance. In this section we review the implications of our results, and consider future avenues of study.

Graph Layout

The results support our hypothesis that graph layout would affect performance, but not in the manner anticipated. Participants' performance, as measured in terms of PAO, was impacted by neatness. Surprisingly, mean PAO was better on nonneat graphs than on neat graphs (regardless of the number of crossings in the instances) contrary to our hypothesis that neatness would have a positive impact on performance. Furthermore, Group A's PAO was better than Group B's on neat graphs. It is unclear what the implications of these results are. But participants in both groups mentioned in debriefing that they tended to work from the "outside" of the graph. Starting at the outside of the graph could yield different results on neat graphs than on nonneat. It may be that identifying the outside area is more difficult in nonneat graphs, and that human performance improves when avoiding outside areas, or at least concentrating on central areas of the graphs. In order to better determine how neatness impacts performance, further study is needed, perhaps by manipulating the types of vertices present on the periphery of the graph on neat and nonneat graphs. It appears that crossings do not negatively impact performance. This contradicts our hypothesis that crossings would make it harder for participants to discriminate between edges and the connectivity of specific vertices. It remains to be seen, however, if this holds for larger instances, both in terms of *n* and *e*.

Techniques

Of the three techniques we identified as candidates, both those based on FPT reduction rules were clearly adopted by participants.

D1-Candidate Vertices

Participants were highly likely to choose D1-candidate vertices to add to the vertex cover on instances where they were available. This indicates that participants are likely able to identify the D1 technique when attempting to solve instances of MVC, and provide insight into a sub-goal commonly identified by novices. Many participants mentioned during debriefing that they noticed that D1-candidate vertices were good to add to the cover. This technique appears to be easily identified by novices.

D2-Candidate Vertices

Participants were very likely to choose to add D2-candidate vertices where they were available, specifically on Graphs 8, 17, and 21.1. This trend was not present on instances of Graph 28, indicating that, although individuals may be able to apply the D2 technique under certain circumstances, some aspect of Graph 28 appears to have caused interference. The complexity of this technique may contribute to the lack of clarity, as the D2 technique relies upon identifying three mutually connected vertices: one of degree two, and two of higher degree. During debriefing, some participants stated that they noticed that in a triangle of vertices, two vertices had to be included in the vertex cover, but it was not articulated that they noticed that it was optimal to not include the degree two vertex in the cover. This technique is more complex, and may not be as readily apparent to all participants. Further study is needed to determine what factors contribute to participants' ability to apply the D2 technique. For instance, since all instances presented were exclusively D1 or D2 type graphs, it was not possible to determine if these techniques interfere with each other, or if one is more dominant than the other. The third technique, the opportunistic technique, was not clearly adopted by participants. However, its early introduction in Group A may be connected to that groups' lower overall PAO.

Opportunistic Graphs

Group A was given an instance set that included graphs resistant to the opportunistic approach in the first half of the instance set, whereas Group B was not introduced to these graphs until the second half of the instance set. There was no significant difference in the use of the opportunistic technique by Group A vs. Group B. This could be a result of the small number of instances (four) that were resistant to the opportunistic technique. The difference in PAO between the two groups, however, may indicate that early introduction to these opportunistic-resistant graphs lead Group A to shift their strategies, resulting in lower PAO. Being introduced to instances that contradicted a natural assumption, that high degree vertices should be considered for the vertex cover, may have influenced Group A's choice of techniques overall. Although they were not less likely to pick high degree vertices at the onset, this does not preclude the possibility that they avoided them further along in the process of selecting vertices for the vertex cover. The overall lower PAO of participants in Group A, indicates some difference in technique.

Instance Size and PAO

Although there were no significant differences in mean PAO based on either measure of instance size, the implication of this outcome is not immediately clear. Larger instance sizes may reveal a clearer relationship between instance size and difficulty.

Future Lines of Study

This study was a first look at human performance on the computationally hard problem Minimum Vertex Cover. As such, there are a number of avenues that remain to be investigated. We briefly address some of these lines of study.

This analysis focused mainly on PAO as a measure of performance, however there are other measures that should be considered as well. One aspect that was not considered in this study was the length of time taken to solve instances of Minimum Vertex Cover. It may be possible to classify the difficulty of instances based on mean time to complete. This, taken into consideration with instance properties such as D1- and D2-candidate vertices, may provide insight into what strategies participants are more prone to apply. Another measure of performance is the number of moves (forward and backward) required to solve instances, which may provide different insight into strategies that participants use to solve instances.

Given that Group A and Group B in this study showed significant differences, it is natural to wonder what impact experience and learning have on performance. By manipulating problem ordering, we can investigate whether individuals recognize non-optimal strategies, and learn to avoid them, or learn to consistently apply optimal techniques.

Large instances of this problem may be challenging in part due to limitations of working memory. This burden may be lessened through the use of cognitive aids. A next logical step is to develop a digital interface for the presentation of instances of this problem that would allow user interaction, providing cognitive aids to reduce the load on working memory and make it easier for participants to determine the quality of their solution. If we consider a means-end analysis approach, participants also need to make quality comparisons of the current state of the instance to the goal state. This process might be assisted by marking all incident edges when a vertex is selected, allowing participants to more clearly determine the impact and quality of the inclusion of a vertex in the cover. The inverse, being able to easily see the impact of *removing* a vertex from the cover, may also be useful. This would allow not only linear backtracking, but also *hyper-jumping:* the ability to undo any selected vertex not just the most recently chosen one. Finally, feedback at the end of solving each instance about the quality of solution may also provide participants with vital evaluative information regarding completed covers as well as motivation on subsequent instances.

Acknowledgments

This work is funded in part by the Natural Sciences and Engineering Research Council (NSERC).

References

Arora, S. (1997). Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of the 38th Annual Symposium on Foundations of Computers Science,* pp. 554–563. http://dx.doi.org/10.1109/SFCS.1997.646145

Atwood, M. E., Masson, M. E. J., & Polson, P. G. (1980). Further explorations with a process model for water jug problems. *Memory & Cognition, 8*(2). 182–192. http://dx.doi.org/10.3758/BF03213422

Bellman, R., & Brock, P. (1960). On the concepts of a problem and problem-solving. *The American Mathematical Monthly, 67*(2). 119–134. http://dx.doi.org/10.2307/2308525

Chronicle, E. P., MacGregor, J. N., Ormerod, T. C., & Burr, A. (2006). It looks easy! Heuristics for combinatorial optimization problems. *The Quarterly Journal of Experimental Psychology, 59*(4). 783–800. http://dx.doi.org/10.1080/02724980543000033

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2003). *Introduction to Algorithms* (2nd Ed.). Cambridge, MA: The MIT Press.

Downey, R. G., & Fellows, M. R. (1998). *Parameterized complexity.* Monographs in computer science. New York: Springer-Verlag.

Dry, M., Lee, M. D., Vickers, D., & Hughes, P. (2006). Human performance on visually presented traveling salesperson problems with varying numbers of nodes. *The Journal of Problem Solving, 1*(1). 20–32.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York : W. H. Freeman and Company.

Gigerenzer, G., & Gaissmaier, W. (2011). Heuristic decision making. *The Annual Review of Psychology, 62*. 451–482. http://dx.doi.org/10.1146/annurev-psych-120709-145346

Gigerenzer, G., & Selten, R. (2001). Rethinking rationality. In G. Gigerenzer, R. Selten (Eds.), *Bounded rationality* (pp. 1–12). Cambridge, MA: The MIT Press.

Graham, S. M., Joshi, A., & Pizlo, Z. (2000). The traveling salesman problem: A hierarchical model. *Memory & Cognition, 28*(7). 1191–1204. http://dx.doi.org/10.3758/BF03211820

Hemmati, M., & Smith, J. C. (2011). Finite optimal stopping problems: The seller's perspective. *Journal of Problem Solving, 3*(2). 72–95.

Kong, X., & Schunn, C. D. (2007). Global vs. local information processing in visual/spatial problem solving: The case of the traveling salesman problem. *Cognitive Systems Research, 8*(3). 192–207. http://dx.doi.org/10.1016/j.cogsys.2007.06.002

Lee, M. D., (2006). A hierarchical Bayesian model of human decision-making on an optimal stopping problem. *Cognitive Science, 30*(3). 1–26. http://dx.doi.org/10.1207/s15516709cog0000_69

Lee, M. D., Brown, M., & Vickers, D. (1997). Neural network and tree search algorithms for the generation of path-following (trail-making) tests. *Journal of Intelligent Systems, 7*(1–2). 117–144. http://dx.doi.org/10.1515/JISYS.1997.7.1-2.117

Lee, M. D., Zhang, S., Munroe, M. N., & Steyvers, M. (2011). Psychological models of human and optimal performance on bandit problems. *Cognitive Systems Research, 12*(2). 164–174. http://dx.doi.org/10.1016/j.cogsys.2010.07.007

MacGregor, J. N., Chronicle, E. P., & Ormerod, T. C. (2004). Convex hull or crossing avoidance? Solutions heuristics in the traveling salesperson problem. *Memory & Cognition, 32*(2). 260–270. http://dx.doi.org/10.3758/BF03196857

MacGregor, J. N., Chronicle, E. P., & Ormerod, T. C. (2006). A comparison of heuristics and human performance on open versions of the traveling salesperson problem. *The Journal of Problem Solving, 1*(1). 33–43.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Saalweachter, J. & Pizlo, Z. (2008). Non-Euclidean traveling salesman problem. In: T. Kugler, J. C. Smith, T. Connolly, & Y-J. Sun (Eds.) *Decision modeling and behavior in complex and uncertain environments* (pp. 339–358). Springer optimization and its applications (vol. 21). New York: Springer. http://dx.doi.org/10.1007/978-0-387-77131-1_14

Pizlo, Z., & Li, Z. (2005). Solving combinatorial problems: The 15-puzzle. *Memory & Cognition, 33*(6). 1069–1084. http://dx.doi.org/10.3758/BF03193214

Stege, U. (2000). *Resolving conflicts from problems in computational biology*. (Doctoral dissertation). Retrieved from Diss. Technische Wissenschaften ETH Zürich. (13364)

Tak, S., Plaisier, M., & van Rooij, I. (2008). Some tours are more equal than others: The convex-hull model revisited with lessons for testing models of the traveling salesperson problem. *The Journal of Problem Solving*, *2*(1). 4–28.

Todd, P. M., & Gigerenzer, G. (2000). Précis of *Simple heuristics that make us smart*. *Behavioral and Brain Sciences, 23*(5). 727–741. http://dx.doi.org/10.1017/S0140525X00003447

Vazirani, V. V. (2004). *Approximation algorithms.* Berlin: Springer-Verlag.