Where the *really* hard choices are: A general framework to quantify decision difficulty

Juan Pablo Franco[1], Nitin Yadav[1], Peter Bossaerts[1,2], Carsten Murawski[1*]

1 Brain, Mind & Markets Laboratory, Department of Finance, The University of Melbourne, Melbourne, Victoria, Australia

2 Florey Institute of Neuroscience and Mental Health, Melbourne, Victoria, Australia

* Corresponding author

E-mail: carstenm@unimelb.edu.au

Abstract

Current models of decision-making more often than not ignore the level of difficulty of choices or treat it only informally. Yet, difficulty has been shown to affect human decision quality. We propose instance complexity (IC), a measure of computational resource requirements, as a generalisable framework to quantify difficulty of a choice based on a small number of properties of the choice. The main advantage of IC compared to other measures of difficulty is fourfold. Firstly, it is based on the theory of computation, a rigorous mathematical framework. Secondly, our measure captures complexity that is intrinsic to a decision task, that is, it does not depend on a particular solution strategy or algorithm. Thirdly, it does not require knowledge of a decision-maker's attitudes or preferences. And lastly, it allows computation of difficulty of a decision task ex-ante, that is, without solving the decision task. We tested the relation between IC and (i) decision quality and (ii) effort exerted in a decision using two variants of the 0-1 knapsack problem, a canonical and ubiquitous computational problem. We show that participants exerted more effort on instances with higher IC but that decision quality was lower in those instances. Together, our results suggest that IC can be used as a general framework to measure the inherent complexity of decision tasks and to quantify computational resource requirements of choices. The latter is particularly relevant for models of resource allocation in the brain (meta-decision-making/cognitive control). Our results also suggest that existing models of decision-making that are based on optimisation (rationality) as well as models such as the Bayesian Brain Hypothesis, are computationally implausible.

Introduction

Most theories of decision-making ignore the difficulty of making a decision [1–3]. They assume that the decision-maker is always able to identify the best option—whether it is a choice between two flavours of ice cream or a choice of investment option for a retirement portfolio from thousands of available options. This is the case not only for rational choice theories of decision-making [4–6], but also for theories of bounded rationality [7–9] and theories of computational rationality [10, 11]. All of those theories assume, implicitly or explicitly, that an observed choice is the outcome of a (possibly constrained) optimisation problem.

Where decision difficulty has been taken into account, it has been done either informally or in a highly domain-specific way. An example of the former are approaches based on heuristics [12, 13]. In this line of research, it is proposed that decision-makers use simple rules or procedures as 'short cuts' to overcome various forms of cognitive limitations. These approaches do not usually demonstrate, however, if and in what ways the proposed heuristics overcome various cognitive limits.

Other work on decision difficulty is domain-specific and cannot necessarily be generalised. For example, it has been shown that the ability of human participants to find the optimal solution to the set cover and maximum coverage problems can be predicted from a set of mathematical properties of the graph representations of the problem instances [14]. Similarly, human performance in the 0-1 knapsack problem has been shown to vary according to a complexity measure based on the *Sahni* algorithm [15]. It is not obvious if and how the characterisation of difficulty could be transferred to other domains.

An important related question is how decision-makers detect the difficulty of decision tasks. This is important, in particular, for the allocation of limited cognitive resources during the decision-making process [1, 16]. It has been suggested that decision-makers learn the features of decision tasks that make them difficult and choose their strategy accordingly [17–19]. However, it is an open question whether there is a set of features that makes decision tasks difficult and why. Detecting this set of features might be as hard as, or indeed require, solving these problems, and exceed the computational resources available to people.

We propose that computational complexity theory (CCT) provides a general theoretical framework that lends itself to characterising difficulty of decisions. CCT is a branch of computing theory that studies the computational resource requirements for solving a task [20–22]. Traditionally, CCT has been used to characterise complexity of computational *problems*. An example of a computational problem is sorting of an array. Other well-studied computational problems include the travelling salesman problem, the subset sum problem or the satisfiability problem. An *instance* of a problem is a particular case of the problem, for example, a particular array of numbers to be sorted. The traditional way of defining the computational complexity of problems is only of limited use for the study of decision-making for various reasons. Firstly, the approach measures the complexity of problems by studying how efficiently problems can be solved as they increase in size. This is done by considering how computational resource requirements scale, *in the worst case*, given the input size of the problem. Using the example of array sorting, problem complexity is concerned with the growth of computational resource requirements (e.g., number of computational steps, memory), in the worst case, as a function of the size of the initial array. Secondly, it ignores the fact that *instances* of a problem with a fixed input size can vary vastly in terms of computational resource requirements. For example, sorting an array that is already in the desired order will tend to take less time than sorting an array that isn't.

We propose that instance complexity theory (IC), a related framework, is more useful for characterising difficulty of decisions. The aim of instance complexity is the characterisation of the computational complexity of individual instances of a problem, based on an instance's properties. For example, in the case of array sorting, it would be based on properties of the input array. IC theory achieves this aim without reference to a particular algorithm or model of computation [23–25]. Thus, it is considered to characterise the *inherent* computational complexity of instances. Moreover, IC has been shown to be applicable to a wide range of problems including the hamiltonian circuit problem [23], the graph colouring problem [23], the travelling salesman problem [26], the knapsack decision problem [27], and the K-SAT problems (boolean satisfiability problems) [23, 24, 28]. These results suggest that the theory is general.

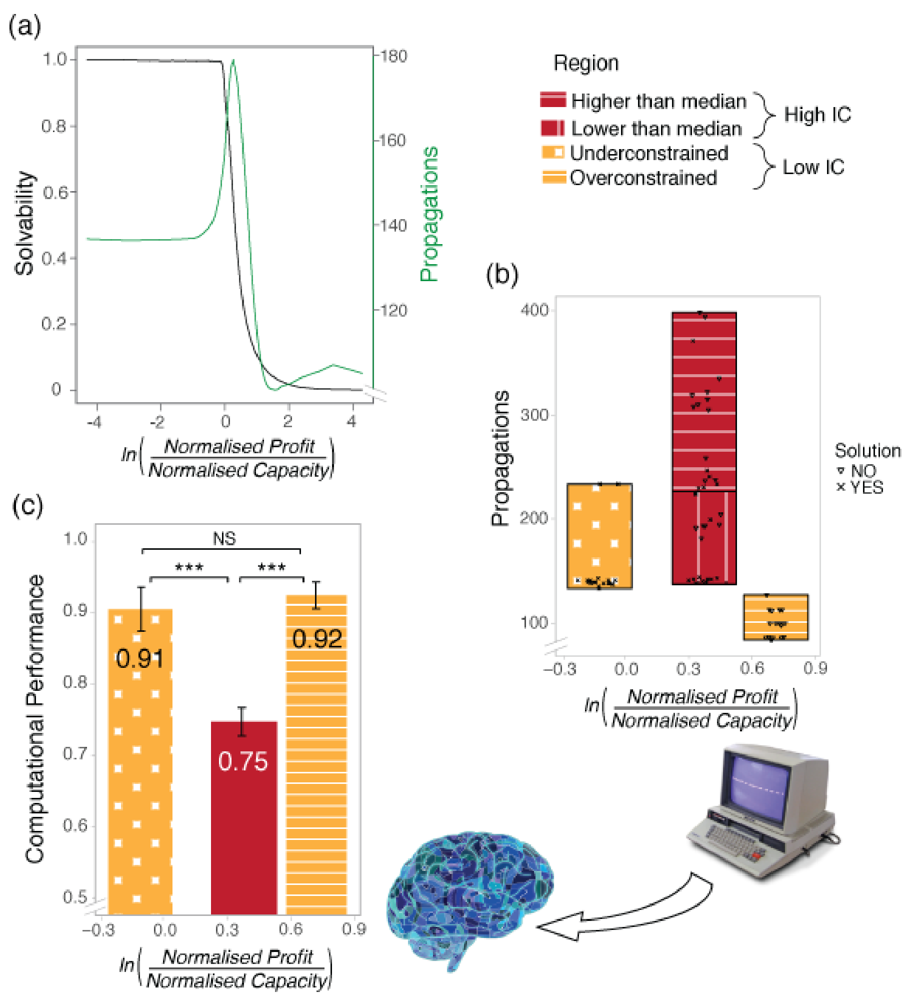
Here, we use IC to characterise the computational complexity of instances of the 0-1 knapsack decision problem. The problem involves selecting a subset from n items with which to fill a knapsack (rucksack) with a specified weight capacity c and a target profit p. Each item has a weight w and a value v. The aim is to decide if there is a subset A of the items for which (1) the sum of weights ($\sum_{i\in A} w_i$) is lower or equal to the capacity c and (2) the sum of values ($\sum_{i\in A} v_i$) yields a target profit p (see S1 Appendix).

The knapsack problem is ubiquitous in everyday life. It is present in problems involving choice of stimuli to attend to, budgeting and time management, portfolio optimisation, intellectual discovery as well as in industrial applications such as the cargo business [29–31]. The problem can also be used to model the symptoms of certain mental disorders such as attention-deficit/hyperactivity disorder [31]. Additionally, the knapsack problem has been widely studied. Not only does there exist a wide range of algorithms to solve the knapsack problem and its extensions. The computational complexity of the problem has been investigated extensively [27, 29].

To apply IC to the knapsack problem, we exploit an important mathematical and statistical property of the problem. When sampling a random instance, the probability that the correct answer to the instance is 'yes' (henceforth *solvable*) can be calculated based on a small set of characteristics of the instance itself [27]. This *solvability* probability exhibits a phase transition, that is, an abrupt shift between 0 and 1 within a narrow range of instance parameters [27]. This boundary separates instances of the problem into two regions: an under-constrained region where the constraints are lenient, and thus many solutions are likely to exist, and an over-constrained region where the constraints are stringent, and thus the existence of a solution is unlikely. Instances in the proximity of this boundary have substantially higher computational complexity than instances further away from it (Fig 1a). This means that there is a mapping from instance characteristics to computational complexity of the instance. We use this mapping as a basis to define IC for the knapsack problem.

In the present study, we tested whether IC thus defined predicts both effort exerted and decision quality in an instance. To this end, we conducted an experiment in which twenty participants each completed two variants of the 0-1 knapsack problem, the decision and the optimisation variant. The optimisation variant differs from the former in that the aim is to maximise the value of the items in the knapsack given a capacity constraint (see S1 Appendix). The two tasks are representatives, respectively, of the two main classes of computational problems, decision problems and optimisation problems.

Fig 1. Instance Complexity and performance in the Knapsack Decision Task. (a) Computer performance and the phase transition. Probability of an instance being *solvable* as a function of the natural logarithm of the normalised profit to normalised capacity ratio (left axis), and compute time proxy (number of propagations using the *Gecode* solver) to solve an instance (right axis). The values correspond to the knapsack decision problem with 6 items. **(b) Instance sampling for the behavioural experiment.** Each point is an instance sampled as a function of the number of propagations and the natural logarithm of the normalised profit to normalised capacity ratio. Equal number of instances were sampled from each of the four regions: (i) overconstrained region, (ii) underconstrained region, and high IC region with a compute time proxy (iii) higher than the median of those instances within the high IC region and (iv) lower than the median of those instances within the high IC region. **(c) Human performance by region in the Knapsack Decision Task.** Mean computational performance and standard errors. *Note:* $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$; *NS: not significant.*



Computer Image by Marcin Wichary (https://commons.wikimedia.org/wiki/File:Tatung-einstein-computer.png), 'Tatung-einstein-computer', Creative Commons Attribution 2.0 Generic license

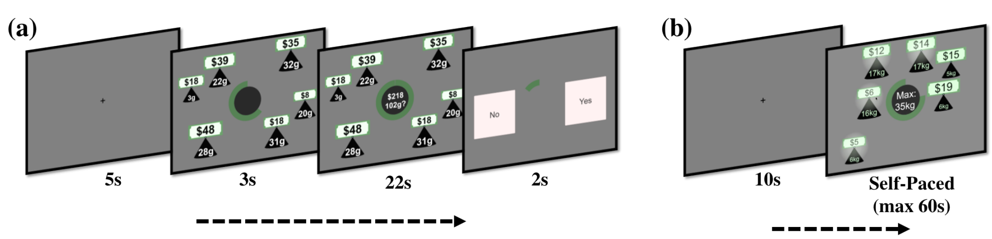
We predicted that performance would be lower in those instances with high IC in both variants. Moreover, we anticipated effort exerted to be positively correlated with IC.

Results

Knapsack Decision Task

Task structure In this task, participants ($n = 20$) were asked to solve a number of instances of the (0-1) knapsack decision problem. In each trial, they were shown a set of items with different values and weights as well as a capacity constraint and a target profit. Participants had to decide whether there exists a subset of those items for which (1) the sum of weights is lower or equal to the capacity constraint and (2) the sum of values yields at least the target profit (Fig 2a; see Methods).

Fig 2. Knapsack Tasks. (a) Knapsack Decision Task. Initially, participants saw a set of items of different values and weights. The green circle at the centre of the screen indicated the time remaining in this stage of the trial. This stage lasted 3 seconds. Then, both capacity constraint and target profit were shown at the centre of the screen. Participants had to decide whether there exists a subset of the items for which (1) the sum of weights is lower or equal to the capacity constraint and (2) the sum of values yields at least the target profit. This stage lasted 22 seconds. Finally, participants had 2 seconds to make either a 'YES' or 'NO' response using the keyboard. A fixation cross was shown during the inter-trial interval (5 seconds). **(b) Knapsack Optimisation Task.** Participants saw a set of items of different values and weights together with a capacity constraint shown at the centre of the screen. The green circle at the centre of the screen indicated the time remaining in this stage of the trial. Participants had to find the subset of items with the highest total value subject to the capacity constraint. This stage lasted 60 seconds. Participants selected items by clicking on them and had the option of submitting their solution before the time limit was reached. After the time limit was reached or they submitted their solution a fixation cross was shown for 10 seconds before the next trial started.



Instances It has been shown that computational complexity of instances in the 0-1 knapsack decision problem can be characterised in terms of a set of instance properties [27].

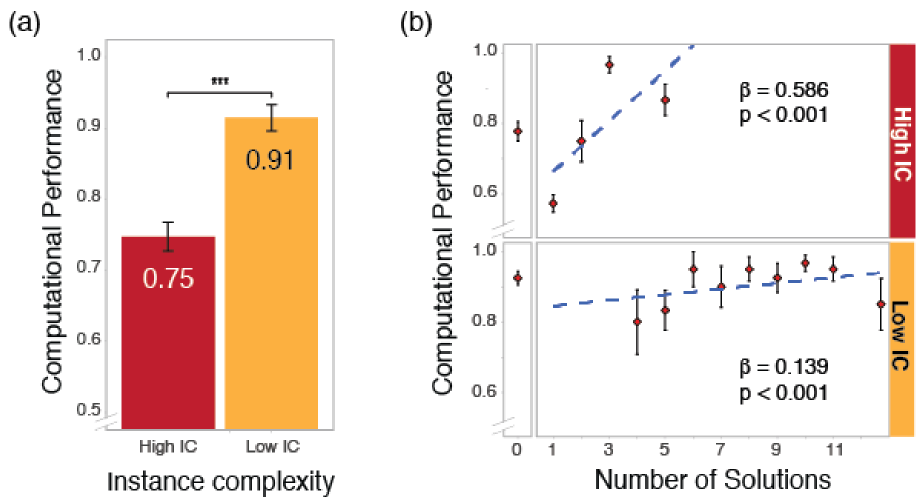
These properties characterise the probability that an instance is solvable, that is, that there exists a subset of items with total weight below the capacity constraint and total value above the target profit. The *solvability* probability exhibits a phase transition [27], which can be characterised in terms of the ratio of the normalised capacity constraint (capacity constraint normalised by sum of all items weights) and the normalised target profit (target profit normalised by sum of all item values). IC is then defined to be higher the closer the instance is to the phase transition (see S1 Appendix for more information). We made use of this property to select instances with high and low IC (see Methods and S3 Appendix for more information). All instances in the experiment had 6 items.

Summary statistics We excluded a total of 13 trials (from 8 participants) in which no response was made. Mean computational performance, measured by the percentage of trials in which a correct response was made, was 83.1% (min $= 0.56$, max $= 0.9$, $SD = 0.08$). On average, participants chose the 'YES' option in 48.1% of trials (min $= 0.32$, max $= 0.60$, $SD = 0.06$). Performance did not vary during the course of the task ($P = 0.196$, main effect of trial number on performance, generalised logistic mixed model (GLMM); S1 Table Model 1), suggesting that neither experience with the task nor mental fatigue affected task performance.

The effect of instance complexity on performance In order to test whether participants' ability to solve an instance was affected by its instance complexity (IC), we compared performance on instances in the phase transition (high IC) with instances outside the phase transition (low IC). Performance was significantly lower on instances in the phase transition ($P < 0.001$, main effect of phase transition proximity on performance, GLMM; Fig 3a; S1 Table Model 2). This suggests that IC affected participants' ability to solve an instance. We further tested this relationship using a continuous parameterisation of IC (see S4 Appendix). We found that this measure captures the negative effect of IC on human computational performance ($P < 0.001$, main effect of continuous measure of IC, GLMM; S4 Appendix).

Effect of solvability and tightness of constraints We hypothesised that performance would be affected by solvability of an instance, that is, whether the answer to the decision problem was 'yes' or 'no'. In order to conclude that an instance is *not solvable*,

Fig 3. Relation between instance complexity and computational performance in the Knapsack Decision Task. (a) Performance on instances of high and low complexity. Mean computational performance of instances grouped by IC. Black lines represent the standard error of the means (SEM). **(b) Relation between performance and number of solutions in the Knapsack Decision Task.** Mean computational performance and standard error by number of solutions. The number of solutions is defined as the number of item combinations that satisfy both capacity and profit constraints. *Note:* $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$; *NS: not significant.*



every possible subset of items needs to be explored in order to determine that none of the subsets satisfies the constraints. Conversely, in case of *solvable* instances, finding a single subset of items is sufficient to determine that the instance is solvable. Such a set may be identified without exploring the full search space and, additionally, there may be more than one such subset. We investigated the effect of solvability and found that the IC was still significant when controlling for solvability ($P < 0.001$, main effect of phase transition on performance, GLMM; S1 Table Model 3), but that there was no significant effect of solvability on performance ($P = 0.355$ main effect of solvability on performance, $P = 0.796$ interaction effect of phase transition and solvability on performance, GLMM; S1 Table Model 3).

For solvable instances, the tightness of the constraints of an instance can be studied further by analysing the number of subsets of items that satisfy the constraints (Fig 3b,1c). We found that for *solvable* instances, the probability of reaching the correct solution increases as the number of subsets that satisfy the constraints increases ($P = 0.001$, main effect of number of subsets on computational performance; GLMM; S1 Table Model 8).

This suggests that participants were more likely to find a solution when there were more possible solutions available. Moreover, this probability increased faster if the instance was in the phase transition ($P < 0.001$, interaction effect of phase transition and number of subsets on computational performance; GLMM; S1 Table Model 8). Furthermore, we found that the mean number of solutions of solvable instances with high IC was lower than for those with a low IC ($P < 0.001$, unpaired t-test).

We also hypothesised that performance would be affected by the tightness of the profit and capacity constraints. We tested whether performance on instances in the over-constrained region was different to performance on instances in the under-constrained region (both of which are outside the phase transition region and thus have low IC). We found no significant difference in performance between the two regions ($P = 0.355$, main effect of region, GLMM; S1 Table Model 7; Fig 1c), but confirmed a significant difference in performance between the phase transition region and each of the other two regions ($P < 0.001$, difference in performance between regions, GLMM; S1 Table Model 6).

Algorithm-specific complexity measures and performance So far, we have used instance complexity measures that are independent of any particular solution algorithm or strategy. That is, we have characterised instance complexity purely in terms of a small set of instance properties. We now investigate whether participants' performance was related to the computational resource requirements of two generic solution algorithms. In particular, we tested whether human performance was related to the number of computational operations these algorithms needed to perform in order to solve an instance.

To perform this test, we considered two widely-used, generic solution algorithms, *Gecode* [32] and *Minisat*$^+$ [33, 34]. Gecode is a constraint-based solver that uses a constraint propagation technique with different search methods, such as branch-and-bound. Minisat$^+$, on the other hand, transforms the problem into a sequence of satisfiability problems that are then solved using constraint propagation and backtracking. For each of these solvers, we chose an output variable that indicates the difficulty for the algorithm to find a solution and whose value is highly correlated with computational time. For Minisat$^+$ we used the number of decisions and for Gecode we used the number of propagations. Both metrics measure the search effort the respective solver had to make to find the solution, which is related to the number of computational steps performed and thus to computational time (see S2 Appendix). We did not use computational time directly because for small size instances, like the ones used in this study, computational time is highly confounded by time spent on reading in the instance, which is not the case for the other variables we considered.

We found that performance in the instances was negatively related to the number of propagations the Gecode algorithm used ($P < 0.001$, main effect of number of propagations, GLMM; S1 Table Model 4). The relation between performance and the Minisat$^+$ decisions measure was not significant ($P = 0.395$, main effect of number of decisions, GLMM; S1 Table Model 5). This finding might provide insights into which approach participants used to solve the instances (see Discussion).

Knapsack Optimisation Task

Task structure After solving the Knapsack Decision Task, participants were asked to solve a number of instances of the (0-1) knapsack optimisation problem. In each trial, they were shown a set of items with different weights and values as well as a capacity constraint. Participants had to find the subset of items that maximises total value subject to the capacity constraint. This means that while in the knapsack decision problem, participants only needed to determine whether a solution exists, in the knapsack optimisation problem, they also needed to determine the nature of the solutions (items in the optimal knapsack; Fig 2b).

Instances To generate instances for the task, a sampling process similar to the one for the Knapsack Decision Task was used (see the Methods section and S3 Appendix for more information). The IC of the optimisation instances was defined according to the IC of the corresponding decision problem at the solution (see S1 Appendix).

Summary statistics We excluded 2 trials (from 2 participant) because solutions were submitted after less than 1 second into the task. In the analysis of submission times, 3 participants were excluded because they never submitted a solution before the time-out, suggesting that they did not understand the submission instructions.

We first analysed participants' ability to find the optimal solution of an instance.

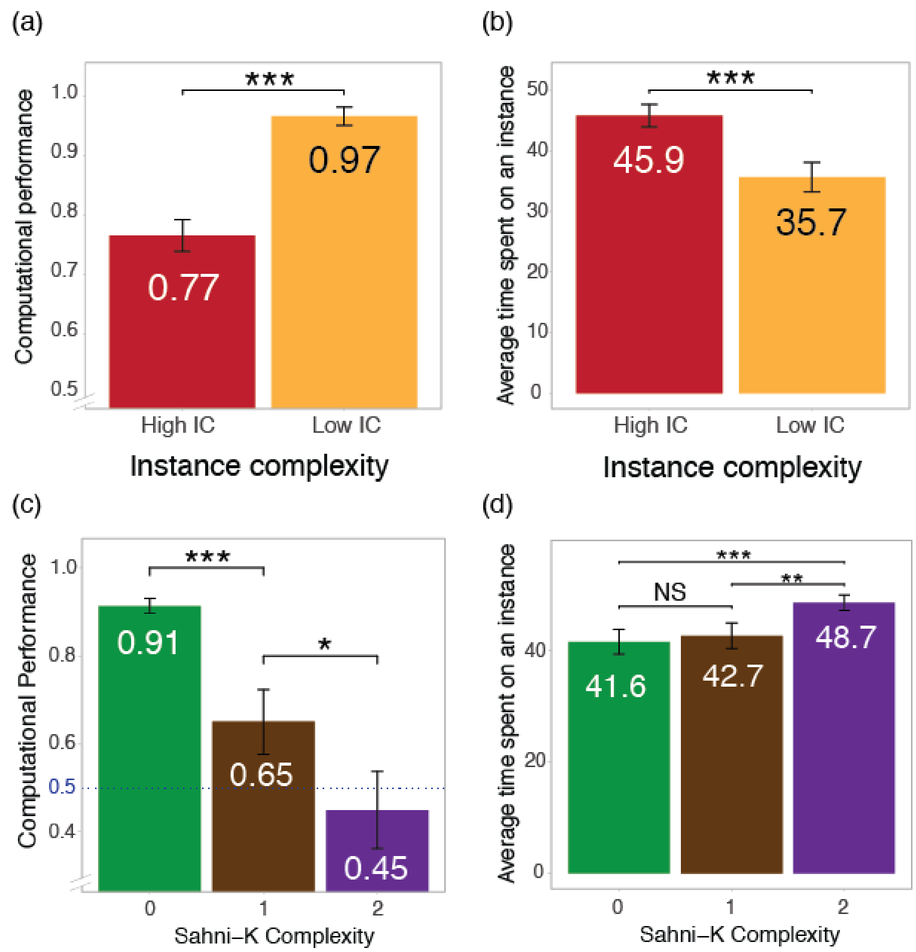
We define *computational performance* as a dichotomous variable that is equal to 1 if the participant obtained a value equal to the maximum value obtainable in the instance, and 0 otherwise. Mean computational performance was 83.2% (min = 0.67, max = 0.94, $SD = 0.08$). Participants spent 43.5 seconds on average on an instance (min = 27.4, max = 60.0, $SD = 8.9$). Participants were allowed to select any set of items, irrespective of the capacity constraint, which implied that they had to ensure that their candidate solution met the capacity constraint. The capacity constraint was only violated in 3% of instances. Performance did not change throughout the task ($P = 0.683$, main effect of trial number on performance, GLMM; S2 Table Model 1), nor did the time spent per instance ($P = 0.483$, main effect of trial number on time, linear mixed model (LMM); S3 Table Model 1), suggesting that neither experience nor mental fatigue affected task performance.

The relation between instance complexity and performance We hypothesised that computational performance in instances in the phase transition would be lower than in instances outside the phase transition. We found that mean computational performance was lower in those instances whose solutions have a corresponding decision problem in the phase transition, relative to those instances whose solutions have a corresponding decision problem outside the phase transition($P < 0.001$, main effect of phase transition proximity, GLMM; Fig 4a; S2 Table Model 2).

So far, we have defined computational performance as a dichotomous variable. We now look at a finer-grained measure. To this end, we define *item performance* as the minimum number of item replacements that are necessary to change a candidate solution to the optimal solution. This includes both the removal of items that are not in the optimal solution and the addition of items that are in the optimal solution (but not part of the candidate solution). The higher the value of this measure, the further away the candidate solution is from the optimum. We found that item performance thus defined was lower, on average, in instances with high IC relative to instances with low IC ($P < 0.001$, main effect of phase transition, LMM; S4 Table Model 2).

Another way of defining performance is in terms of value obtained in an instance. We define *economic performance* as the ratio of the total value of items in the submitted solution to the total value of items in the optimal solution. We found that economic

Fig 4. Relation between computational complexity and performance in the Knapsack Optimisation Task. (a) Relation between instance complexity and computational performance. Mean computational performance and standard error of the means (SEM) in the knapsack optimisation task according to IC of the corresponding knapsack decision instance. **(b) Relation between instance complexity and effort exerted on an instance.** Mean time spent (and SEM) in the Knapsack Optimisation Task according to IC of the corresponding knapsack decision instance. **(c) *Sahni-k* Complexity and Performance.** Mean computational performance and SEM. **(d) *Sahni-k* Complexity and Effort.** Average time spent and SEM. *Note:* **p<0.1;* ***p<0.05;* ****p<0.01;* NS*: not significant.*

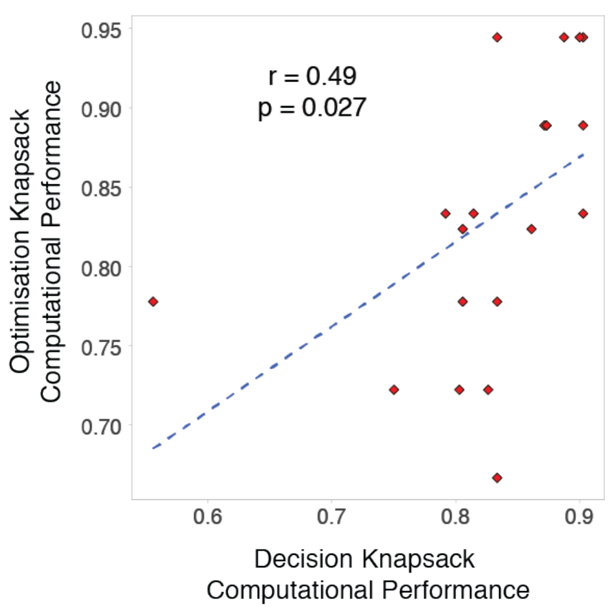


performance was lower in instances with high IC relative to instances with low IC ($P < 0.001$, main effect of phase transition, LMM; S4 Table Model 1).

Relation of performance in Knapsack Decision Task and Knapsack Optimisation Tasks The Knapsack Decision Task and the Knapsack Optimisation Task are based on two fundamentally different types of computational problems. The former is a decision problem with a yes/no answer and a member of the problem complexity class NP-complete. The latter is an optimisation or search problem with the goal to find the maximal value of the value obtainable under the capacity constraint. It is a member of the complexity class NP-hard. The knapsack optimisation problem can be considered as a problem in which the decision-maker has to solve a sequence of knapsack decision problems, starting with the empty set and continuing to the point where there does not exist another admissible subset of items with a higher total value than the current one.

We therefore hypothesised that participants' performance in the two tasks would be related and that participants who performed better in the Knapsack Decision Task would also perform better in the Knapsack Optimisation Task. We found a positive and significant correlation between computational performance in the two tasks (Pearson Correlation = 0.49, $P = 0.027$, d.f. = 18; Fig 5). This result is even stronger if we exclude one participant with performance in the Knapsack Decision Task significantly below the performance of any other participant (Pearson Correlation = 0.67, $P = 0.002$, d.f. = 17). These findings suggests that the two tasks draw on similar cognitive capacities.

Fig 5. Relation of performance in the Knapsack Optimisation and Knapsack Decision Tasks. The overall performance of the decision and optimisation tasks by participant, defined as the mean computational performance.



The relation between instance complexity and effort The Knapsack Optimisation Task also allowed us to investigate effort exerted on an instance. While we could not measure effort directly, we considered the time spent on each instance as a proxy. As we did not incorporate any direct opportunity costs to time in our experimental setting, clock time does not capture this aspect of effort. However, clock time increases in the number of computations performed, as well as the time required for each computation. This justifies using time spent on each instance as a measure of effort. Participants spent more time in instances with high instance complexity relative to those outside of the phase transition ($P < 0.001$, main effect of phase transition proximity, LMM; Fig 4b; S3 Table Model 2). This effect was also present when controlling for computational performance ($P = 0.037$, main effect of phase transition proximity, LMM; S3 Table Model 6).

Next, we analysed the relation between effort exerted in an instance and performance in the instance. We found a negative relation between effort and the probability of finding the solution ($P < 0.001$, main effect of time, GLMM; S2 Table Model 7). However, when we account for instance complexity, the effect of effort on performance is no longer significant ($P = 0.905$, main effect of time; $P = 0.352$, interaction effect of time and phase transition, GLMM; S2 Table Model 3). Taken together with previous results, it appears that the relation between effort and computational performance is moderated by instance complexity. The fact that the probability of finding the optimal solution is lower when participants spend more effort may have been caused by participants spending more effort on instances with a high IC. This, however, suggests that participants are somehow able to adjust their level of effort in response to instance complexity, which we will return to in the Discussion.

In order to further examine the relationship between optimisation instances, effort and IC, we examined the amount of time people spent after each click at each selection of items before doing the next click. After each click participants were faced with the question: "Is there another set of items with a higher profit that still satisfies the weight capacity constraint?" We found that participants spent more time at those stages in which there were fewer options that yielded a more valuable solution, whilst still satisfying the capacity constraint ($P < 0.001$, main effect of the number of more valuable solutions, LMM; S5 Table).

Relation between algorithm-specific complexity measures, effort and performance We next examined a set of alternative complexity measures based on the generic solution algorithms *Gecode* and *Minisat*$^+$. We found qualitatively similar results to those of the knapsack decision problem, with higher instance difficulty, according to *Gecode* propagations associated with lower average performance ($P < 0.001$, main effect of number of propagations, GLMM; S2 Table Model 4). For the *Minisat*$^+$ number of decisions this effect was not significant ($P = 0.157$, main effect of number of decisions, GLMM; S2 Table Model 5).

We also examined whether these complexity measures were related to the time spent on each of the instances. We found that, in line with previous results, instances with higher Gecode propagations were associated with higher levels of effort ($P < 0.001$, main effect of number of propagations, LMM; S3 Table Model 3). We found a similar relation for the Minisat$^+$ decision measure ($P = 0.001$, main effect of number of decisions, LMM; S3 Table Model 4).

We also analysed the relation between computational performance and *Sahni-k*, another measure of instance complexity. *Sahni-k* is proportional to both the number of computations and the amount of memory required to solve an instance of the Knapsack Optimisation Task. This metric has previously been shown to be associated with performance in the Knapsack Optimisation Task [15, 30]. We found a negative relation between *Sahni-k* and computational performance ($P < 0.001$, main effect of *Sahni-k*, GLMM; Fig 4c; S2 Table Model 6) and a positive relation between *Sahni-k* and effort ($P = 0.001$, main effect of *Sahni-k*, LMM; Fig 4d; S3 Table Model 5), confirming the findings of a previous study [15]. However, when controlling for IC, the effect of *Sahni-k* on effort is no longer significant ($P = 0.580$, main effect of *Sahni-k*, LMM; S3 Table Model 7), in line with results reported above.

Relation between performance in knapsack tasks and cognitive function

Finally, we investigated the relation between performance in two knapsack tasks and various aspects of cognitive function. In particular, we used tests aimed at assessing mental arithmetic, working memory, episodic memory, strategy use as well as processing and psychomotor speed. Correlations between performance in these tasks and the knapsack tasks were all non-significant (see Methods and S6 Table for details).

Discussion

Current models of decision-making more often than not ignore the level of difficulty of problems or treat it only informally [1–3]. We propose a generalisable framework to quantify difficulty of a decision task based on the decision's inherent complexity. The framework is based on instance complexity (IC) theory, a branch of computational complexity theory, that relates properties of instances of a computational problem to computational resource requirements. We tested the effect of IC on decision quality in two variants of a canonical task, the decision and optimisation variants of the 0-1 knapsack problem. We also examined effort exerted in the optimisation variant of the 0-1 knapsack problem. We found that IC negatively affects decision quality in both tasks. Moreover, we found that more effort was exerted on instances with higher IC.

The aim of IC theory is to characterise the relation between the number of computational resources (time) required by an algorithm to solve an instance, and properties of the instance. It has been shown for several decision problems (most of them NP-complete) that the probability of an instance having a particular solution (yes/no) can be expressed in terms of an order parameter that is based on a small number of instance properties. Moreover, this probability exhibits a phase transition, that is, there exists a narrow range of values of the order parameter within which the probability of a yes answer changes from close to 0 to close to 1 [23–27, 35]. It has been conjectured that solvability of all NP-complete problems exhibits such a phase transition in terms of an order parameter and that the hard instances, in terms of compute time, of those problems lie in the proximity of the phase transition [23]. It was recently shown that a similar link between hardness of instances and a phase transition in solvability exists for the 0-1 knapsack problem [27]. We exploited this link in the present study.

What makes decisions hard? In the context of decision tasks, it is not entirely understood what makes particular instances hard to solve and why hardness peaks around the phase transition of solvability. One suggestion has been that hardness, that is, compute time, is mainly a function of the tightness of the constraints of an instance [23, 24]. The 0-1 knapsack problem has two constraints, a profit and a weight constraint, that operate in opposite directions. An increase of the weight constraint increases the number of solutions (more subsets of items meet the constraint), ceteris paribus, whereas an increase in the profit constraint decreases the number of solutions, ceteris paribus. For instances with low IC, constraints are either loose or tight. In case the constraints are loose, instances are solvable and many subsets of items satisfy the constraints, making it easy to find one possible solution. If, on the other hand, constraints are tight, there generally does not exist a subset of items that satisfies the constraints, making it easy to conclude that there is no possible solution. Instances with high IC have constraints that are tight enough so that only a few subsets satisfy both constraints, yet they are loose enough to allow, sometimes, a number of possible solutions. We found that solvable instances with high IC had a lower number of solutions than those with low IC. Moreover, we found that as the number of solutions increased, participants' performance in instances with high IC increased more than in those with low IC. These findings suggest that the number of solutions is a key determinant of instance difficulty. Future research should examine more closely the mathematical structure of IC by analysing its relation with the number of solutions.

Complexity and behaviour Our work provides a step towards understanding the effects of computational complexity on behaviour by providing a measure of decision difficulty. We have shown that IC affects behaviour through task performance. Yet, it could also have an impact on behaviour in other ways. For instance, attitudes towards complexity could affect behaviour. Complexity avoidance could lead people to avoid situations that involve solving difficult tasks, whereas complexity seeking could lead to situations in which people seek tasks that require a high amount of effort to be solved [36]. Another way that complexity could be related to behaviour is through its effect on confidence. In the case of the Knapsack Optimisation Task it is still an open question how participants chose when to submit their answer. The IC level could influence the confidence on having found the solution, and in turn this confidence could play a role in the decision of when to submit an answer. We leave it to future work to explore the effects of attitudes towards or preferences for complexity in decision-making, as well as the relation between IC, confidence and behaviour.

Which algorithms did participants use? In addition to analysing IC as a measure of complexity, we investigated other complexity measures that are related more explicitly to the number of computational steps (time) required by an electronic computer to solve an instance. We found that one of the two algorithm-specific complexity measures we considered correlated with both human performance and effort exerted. This is probably related to the main features of each of the algorithms. It is unlikely that humans reformulate the problem as a boolean satisfiability problem in order to reach a solution (MiniSat$^+$). It is more likely that they compute directly on the problem itself as a directed search based on the constraints (Gecode). These results suggest that the computational mechanisms that humans use might be similar in nature to those of particular computer algorithms, a notion that should be explored in more detail by future research.

The relation between decision and optimisation tasks Although the knapsack optimisation and decision problems are two fundamentally different types of computational problems, they are related to each other at a theoretical level. Specifically, the optimisation problem can be solved by the iterative solution of a series of corresponding decision problems. Based on this link, we defined IC for the optimisation problem and found a lower performance on instances with higher IC, thus mirroring the decision problem results. This is further evidence in support of our theoretical framework. We also found that participants who performed better in the decision task tended to perform better in the optimisation task. The latter finding suggests that individual constraints that affected performance were active in both tasks.

The relation between IC and effort exerted One interesting finding is that effort exerted on an instance was adjusted according to IC. This result is perplexing. In order to know which resources a computer algorithm needs to solve an instance, it is necessary for the algorithm to find the solution. That is, a computer algorithm can only compute resource requirements of an instance *ex post*. In contrast, we found that participants adjusted their effort to IC even without being able to find the solution at all. This result is consistent with the findings of a previous study that used a different measure of instance complexity [15].

It is an open question which mechanisms participants used to adjust effort. It has recently been suggested that the brain allocates resources to tasks according to the expected benefits and expected costs, in particular cognitive resource requirements, related to the task [16, 37–39]. These accounts also suggest that decision-makers learn to estimate costs and benefits of a task based on a set of task features [17–19]. These accounts, however, do not specify what these features might be. In fact, selection of these features might be in itself an NP-hard problem. It is conceivable that decision-makers use IC to estimate the expected costs of performing a task. This would require that decision-makers can somehow detect IC [1]. Future research should investigate possible mechanisms of detecting IC.

Performance in the knapsack tasks and basic cognitive abilities Individual differences in performance in the knapsack tasks were independent of individual differences in the set of core cognitive abilities including attention, working memory and mental arithmetic. One possible explanation for the lack of correlation is that these cognitive abilities play only a minor role in solving computationally hard problems and that those problems instead require another cognitive ability that is not captured by any of the tests we administered. Another possible explanation is that we did not measure the active cognitive constraint that drove differences in individual performance. One candidate for such a constraint is memory [40, 41]. It is, of course, also possible that our study did not have sufficient statistical power to detect individual differences. Further research is needed in order to incorporate the full spectrum of cognitive resource limitations and link them to performance and effort in decision tasks [1].

Properties of real-world instances Our results are based on a particular sampling distribution. Specifically we used a uniform distribution to sample the knapsack instances. It is still an open question whether this method is generalisable to other sample distributions and, specially, to those distributions that are important ecologically, that is, that are encountered in everyday life. Characterising the latter distributions of instances is an open research question in computer science [42]. Further research would be required to characterise the probability distribution of knapsack instances found outside of the laboratory setting.

Furthermore, in our study, the task involved finding the optimal solution. However, finding the exact solution might not always be required in the real-world. In many cases finding an approximate solution might suffice. However, for many NP-complete and NP-hard problems, approximating the solution is as hard as finding the optimal solution [20, 43]. It is still an open theoretical question whether IC can be extended to approximation problems. Future research should investigate whether the results found in this study, for both humans and computers, can be extended to approximation instances.

The Church-Turing thesis A core notion in the theory of computation is the Church-Turing thesis. The thesis states that the universal Turing machine is a general model of computation, which implies that any input/output operation that can be performed by a human computer, can also be performed by the universal Turing machine [44–46]. Our findings support a related notion: that an algorithm that requires a larger number of computational resources (time) on a universal Turing machine (here, an electronic computer) also requires relatively more computational resources in the human brain. Thus, our findings strongly suggest that computational tasks have inherent complexity, that is, the amount of computational resources required to solve them is independent of the particular computational model used. The framework we present in this paper is a candidate for the quantification of inherent complexity of decision tasks.

Implications for decision theory and public policy Many theories of decision-making (including meta-decision-making) assume that people optimise [4–7, 9–11, 16, 18, 38, 47]. Our results are consistent with previous results that show that this is often not the case [7, 48]. We show that performance is dependent on task complexity, thus corroborating previous studies that highlight the relevance of incorporating cognitive resource requirements and limitations into decision theory [1, 15, 49]. In addition, our approach allows for a generalisable and formal quantification of those resource requirements in decision and optimisation tasks.

In a broader context, the present study might help to identify the limits of human cognition and decision-making. This is crucial for the design of policies that wish to improve the quality of decisions such as financial investments, selection of insurance contracts, among many others. In those cases where the task is too demanding, mechanisms could be designed to help people improve the quality of their decisions. This could be done, for instance, through software applications that take advantage of the computational power of electronic computers. Finally, our results advocate for closer collaboration between decision scientists and computer scientists. Not only can decision sciences be informed by computation theory, as done in this study, but research on humans could motivate the development of new theories and algorithms.

Methods

Ethics statement

The experimental protocol was approved by the University of Melbourne Human Research Ethics Committee (Ethics ID 1749616). Written informed consent was obtained from all participants prior to commencement of the experimental sessions. Experiments were performed in accordance with all relevant guidelines and regulations.

Participants

Twenty human volunteers recruited from the general population took part in the study (14 female, 6 male; age range = 18-31 years, mean age = 22.0 years). Inclusion criteria were based on age (minimum = 18 years, maximum = 40 years).

Knapsack Decision Task

Task structure In this task, participants were asked to solve a number of instances of the (0-1) knapsack decision problem. In each trial, they were shown a set of items with different values and weights as well as a capacity constraint and a target profit. Participants had to decide whether there exists a subset of those items for which (1) the sum of weights is lower or equal to the capacity constraint and (2) the sum of values yields at least the target profit.

Each trial had four stages. In the first stage (3 s), only the items were presented. Item values, in dollars, were displayed using dollar bills and weights, in grams, were shown inside a black weight symbol. The larger the value of an item, the larger the dollar bill was in size. Similarly, the larger the weight of an item, the larger its weight symbol was in size. At the centre of the screen, a green circle indicated the time remaining in this stage. In the second stage (22 s), target profit and capacity constraint were added to the screen inside the green timer circle. In the third stage (2 s), participants saw a 'YES' or 'NO' buttons on the screen, in addition to the timer circle, and made a response using the keyboard (Fig 2a). A fixation cross was then shown (5 s) before the start of the next trial.

Each participant completed 72 trials (3 blocks of 24 trials with a rest period of 60 s between blocks). Each trial presented a different instance of the knapsack decision problem. The order of instances was randomised for each participant.

Instances All instances in the experiment had 6 items. Instances varied in their computational complexity. It has been shown that computational complexity of instances in the 0-1 knapsack decision problem can be characterised in terms of a set of instance properties [27] (Fig 1a). In particular, IC can be characterised in terms of the ratio of the normalised capacity constraint (capacity constraint normalised by sum of all items weights) and the normalised target profit (target profit normalised by sum of all item values) (see S1 Appendix for more information). We made use of this property to select instances for the task (see S3 Appendix for more information).

We selected the normalised capacity bin of $[0.40-0.45]$ and chose the normalised profit bins that corresponded to the under-constrained (0.35-0.4), phase transition (0.6-0.65) and over-constrained (0.85-0.9) regions. We then randomly selected 18 instances from the under-constrained bin and 18 from the over-constrained bin. Finally, we sampled 18 *solvable* instances and 18 *non-solvable* instances from the phase transition bin (0.4-0.45). Throughout we ensured that no weight/value combinations were sampled twice. In order to also ensure enough variability between instances in the phase transition we added an additional constraint in the sampling from each bin. We forced half of the instances selected in each bin in the phase transition to be easier than the median according to an algorithm specific ex-post complexity measure (*Gecode* propagations parameter) and the other half to be harder than the median (Fig 1b). The order of presentation of instances in the task was randomised for each participant.

Knapsack Optimisation Task

Task structure In this task, participants were asked to solve a number of instances of the (0-1) knapsack optimisation problem. In each trial, they were shown a set of items with different weights and values as well as a capacity constraint. Participants had to find the subset of items that maximises total value subject to the capacity constraint. This means that while in the knapsack decision problem, participants only needed to determine whether a solution exists, in the knapsack optimisation problem, they also needed to determine the nature of the solutions (items in the optimal knapsack).

The task had two stages. In the first stage (60 s), the items were presented together with the capacity constraint and the timing indicator. Items were presented like in the Knapsack Decision Task. During this stage, participants were able to add and remove items to/from the knapsack by clicking on the items. An item added to the knapsack was indicated by a light around it (Fig 2b). Participants submitted their solution by pressing the button 'D' on the keyboard before the time limit was reached. If participants did not submit within the time limit, the items selected at the end of the trial were automatically submitted as the solution. Participants were then shown a fixation cross (10 s) before the start of the next trial.

Each participant completed 18 trials (2 blocks of 9 trials with a rest period of 60 s between blocks). Each trial presented a different instance of the knapsack optimisation problem. The order of the instances was randomised for each participant.

Instances To generate instances for the task, a sampling process similar to the one for the Knapsack Decision Task was used (see S3 Appendix for more information). We selected the same normalised capacity bin as for the Knapsack Decision Task (0.4-0.45) and selected the normalised profit of the solution such that the corresponding decision problem (see S1 Appendix) lied in the phase transition (0.6-0.65) or in the over-constrained region (0.85-0.9). Again, we forced half of the instances selected in each of the bins in the phase transition to be easier than the median, according to the *Gecode* propagations measure, and the other half to be harder than the median. We sampled a total of 18 instances, 12 in the phase transition and 6 out of the phase transition. The order of presentation of instances in the task was randomised for each participant.

Mental arithmetic task

In this task, participants were presented with 33 mental arithmetic problems [50]. The first three trials were considered test trials and thus were not included in the analysis. They were given 13 seconds to solve each problem. The task involved addition and division of numbers, as well as questions in which they were asked to round to the nearest integer the result of an addition or division operation.

Basic cognitive function tasks

In addition, we also tested participants' performance on four aspects of cognitive function that we considered relevant for the knapsack tasks, namely, working memory, episodic memory, strategy use as well as processing and psychomotor speed. To do so, we administered the Reaction Time (RTI), Paired Associates Learning (PAL), Spatial Working Memory (SWM) and Spatial Span (SSP) tasks from the Cambridge Neuropsychological Test Automated Battery (CANTAB) [51].

Procedure

After reading the plain language statement and providing informed consent, participants were instructed in each of the tasks and completed a practice session for each task. Participants first solved the CANTAB RTI task, followed by the Knapsack Decision Task. Then they completed the CANTAB RTI task again, followed by the Knapsack Optimisation Task. Subsequently, they completed the other CANTAB tasks, in the following order: PAL, SWM and SSP. Finally, they performed the mental arithmetic task and completed a set of demographic and debriefing questionnaires. Each experimental session lasted around two hours.

The Knapsack Decision Task, Knapsack Optimisation Task and mental arithmetic task were programmed in Unity3D [52] and administered on a laptop. The CANTAB tasks were administered on a tablet.

Participants received a show-up fee of AUD $10 and additional monetary compensation based on performance. They earned AUD $0.7 for each correct answer in the Knapsack Decision Task and AUD $1 for each correct answer in the Knapsack Optimisation Task.

Statistical Analysis

The R programming language was used to analyse the behavioural data. Python (version 3.6) was used to sample instances and run the simulations.

All of the generalised logistic mixed models (GLMM) and linear mixed models (LMM) included random effects on intercept for participants. Their p-values were calculated using a two-tailed Wald test. All statistical analyses were done in R [53] and mixed models were estimated using the R package lme4 [54].

Data and Code Availability

The raw behavioural data, the data analysis code and the computational simulations are all available from the Open Science Framework.

The Knapsack Decision Task, Knapsack Optimisation Task and mental arithmetic task were programmed in Unity3D [52] and are available as well from the Open Science Framework.

`DOI:10.17605/OSF.IO/T2JV7`

References

1. Bossaerts P, Murawski C. Computational Complexity and Human Decision-Making. Trends in Cognitive Sciences. 2017;21(12):917–929. doi:10.1016/j.tics.2017.09.005.

2. Aaronson S. Why Philosophers Should Care About Computational Complexity. CoRR. 2011;(0844626):58. doi:10.1080/10463280903275375.

3. van Rooij I. The Tractable Cognition Thesis. Cognitive Science: A Multidisciplinary Journal. 2008;32(6):939–984. doi:10.1080/03640210801897856.

4. Samuelson PA. A Note on the Pure Theory of Consumer ' s Behaviour. Economica. 1938;5(17):61–71. doi:10.1111/j.l468-0335.2008.00718.x.

5. Jehle GA, Reny PJ. Advanced microeconomic theory. 3rd ed. Financial Times/Prentice Hall; 2011.

6. Nash JF. Equilibrium points in n-person games. Proceedings of the National Academy of Sciences of the United States of America. 1950;36(1):48–49. doi:10.1073/pnas.36.1.48.

7. Gigerenzer G, Selten R. Bounded rationality : the adaptive toolbox. MIT Press; 2001.

8. Lieder F, Griffiths TL. When to use which heuristic: A rational solution to the strategy selection problem. In: Noelle DRWASYJMTJCD&MPP D C, editor. Proceedings of the 37th Annual Conference in the Cognitive Science Society. Austin, TX: Cognitive Science Society; 2015. p. 1362–1367.

9. Gigerenzer G, Brighton H. Homo Heuristicus: Why Biased Minds Make Better Inferences. Topics in Cognitive Science. 2009;1(1):107–143. doi:10.1111/j.1756-8765.2008.01006.x.

10. Lewis RL, Howes A, Singh S. Computational rationality: Linking mechanism and behavior through bounded utility maximization. Topics in Cognitive Science. 2014;6(2):279–311. doi:10.1111/tops.12086.

11. Gershman SJ, Horvitz EJ, Tenenbaum JB. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. Science. 2015;349(6245):273–278. doi:10.1126/science.aac6076.

12. Kahneman D, Tversky A. Prospect Theory: An Analysis of Decision under Risk. Econometrica. 1979;47(2):263–292. doi:10.2307/1914185.

13. Tversky A, Kahneman D. Advances in prospect theory: Cumulative representation of uncertainty. Journal of Risk and Uncertainty. 1992;5(4):297–323. doi:10.1007/BF00122574.

14. Bourgin D, Lieder F, Reichman D, Talmon N, Griffiths TL. The Structure of Goal Systems Predicts Human Performance. In: Gunzelmann G, Howes A, Tenbrink T, Davelaar A, editors. Proceedigns of the 39th Annual Meeting of the Cognitive Science Society. Austin, TX: Cognitive Science Society; 2017. p. 1660–1665.

15. Murawski C, Bossaerts P. How Humans Solve Complex Problems: The Case of the Knapsack Problem. Nature (Scientific Reports). 2016;6(34851). doi:10.1038/srep34851.

16. Shenhav A, Musslick S, Lieder F, Kool W, Griffiths TL, Cohen JD, et al. Toward a Rational and Mechanistic Account of Mental Effort. Annual Review of Neuroscience. 2017;40(1):99–124. doi:10.1146/annurev-neuro-072116-031526.

17. Lieder F, Plunkett D, Hamrick JB, Russell SJ, Hay NJ, Griffiths TL. Algorithm selection by rational metareasoning as a model of human strategy selection; 2014. Available from: https://dl.acm.org/citation.cfm?id=2969033.2969147.

18. Lieder F, Griffiths TL. Strategy selection as rational metareasoning. Psychological Review. 2017;124(6):762–794. doi:10.1037/rev0000075.

19. Lieder F, Shenhav A, Musslick S, Griffiths TL. Rational metareasoning and the plasticity of cognitive control. PLoS Computational Biology. 2018;14(4):e1006043. doi:10.1371/journal.pcbi.1006043.

20. Arora S, Barak B. Computational complexity : a modern approach. Cambridge University Press; 2009.

21. Pudlák P. Logical foundations of mathematics and computational complexity. 1st ed. Springer International Publishing; 2013.

22. Moore C, Mertens S. The Nature of Computation. 1st ed. Oxford University Press; 2011. Available from: http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780199233212.001.0001/acprof-9780199233212.

23. Cheeseman P, Kanefsky B, Taylor WM. Where the Really Hard Problems Are. The 12nd International Joint Conference on Artificial Intelligence. 1991; p. 331–337. doi:10.1.1.97.3555.

24. Achlioptas D, Naor A, Peres Y. Rigorous Location of Phase Transitions in Hard Optimization Problems. Nature. 2005;435(7043):759–64. doi:10.1038/nature03602.

25. Monasson R, Zecchina R, Kirkpatrick S, Selman B, Troyansky L, Lhomond R, et al. Determining computational complexity from characteristic ‘phase transitions’. Nature. 1999;400(6740):133–137. doi:10.1038/22055.

26. Gent IP, Walsh T. The TSP phase transition. Artificial Intelligence. 1996;88(1-2):349–358. doi:10.1016/S0004-3702(96)00030-6.

27. Yadav N, Murawski C, Sardina S, Bossaerts P. Phase transition in the knapsack problem. ArXiv e-prints. 2018;(arXiv:1806.10244).

28. Selman B, Kirkpatrick S. Critical behavior in the computational cost of satisfiability testing. Artificial Intelligence. 1996;81(1-2):273–295. doi:10.1016/0004-3702(95)00056-9.

29. Kellerer H, Pferschy U, Pisinger D. Knapsack Problems. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004.

30. Meloso D, Copic J, Bossaerts P. Promoting Intellectual Discovery: Patents Versus Markets. Science. 2009;323(5919):1335–1339. doi:10.1126/science.1158624.

31. Torralva T, Gleichgerrcht E, Lischinsky A, Roca M, Manes F. "Ecological" and Highly Demanding Executive Tasks Detect Real-Life Deficits in High-Functioning Adult ADHD Patients. Journal of Attention Disorders. 2013;17(1):11–19. doi:10.1177/1087054710389988.

32. Gecode Team. Gecode: Generic Constraint Development Environment; 2006. Available from: `http://www.gecode.org`.

33. Een N, S orensson N. An extensible SAT-solver. 2003;.

34. Eén N, Sörensson N. Translating Pseudo-Boolean Constraints into SAT. Journal on Satisfiability Boolean Modeling and Computation. 2006;2:1–25.

35. Fu Y, Anderson PW. Application of statistical mechanics to NP-complete problems in combinatorial optimisation. Journal of Physics A: Mathematical and General. 1986;19(9):1605–1620. doi:10.1088/0305-4470/19/9/033.

36. Inzlicht M, Shenhav A, Olivola CY. The Effort Paradox: Effort Is Both Costly and Valued. Trends in cognitive sciences. 2018;22(4):337–349. doi:10.1016/j.tics.2018.01.007.

37. Shenhav A, Botvinick M, Cohen J. The expected value of control: An integrative theory of anterior cingulate cortex function. Neuron. 2013;79(2):217–240. doi:10.1016/j.neuron.2013.07.007.

38. Boureau YL, Sokol-Hessner P, Daw ND. Deciding How To Decide: Self-Control and Meta-Decision Making. Trends in Cognitive Sciences. 2015;19(11):700–710. doi:10.1016/j.tics.2015.08.013.

39. Dayan P. How to set the switches on this thing. Current Opinion in Neurobiology. 2012;22(6):1068–1074. doi:10.1016/j.conb.2012.05.011.

40. Otto AR, Raio CM, Chiang A, Phelps EA, Daw ND. Working-memory capacity protects model-based learning from stress. Proceedings of the National Academy of Sciences. 2013;110(52):20941–20946. doi:10.1073/pnas.1312011110.

41. Schmeichel BJ. Attention control, memory updating, and emotion regulation temporarily reduce the capacity for executive control. Journal of Experimental Psychology: General. 2007;136(2):241–255. doi:10.1037/0096-3445.136.2.241.

42. Bogdanov A, Trevisan L. Average-Case Complexity. CoRR. 2006;.

43. Ausiello GG, Marchetti-Spaccamela A, Crescenzi P, Gambosi G, Protasi M, Kann V. Complexity and Approximation : Combinatorial Optimization Problems and Their Approximability Properties. Springer Berlin Heidelberg; 1999.

44. Church A. An unsolvable problem of elementary number theory. American Journal of Mathematics. 1936;58(2):345–363. doi:10.2307/2371045.

45. Turing AM. On Computable Numbers, With Application to the Entscheidungs Problem. Proceedings of the London Mathematical Society. 1937; p. 230–265. doi:10.1112/plms/s2-42.1.23.

46. Kleene SC. Introduction to Mathematics. New York: Van Nostrand; 1952.

47. Botvinick MM, Cohen JD. The computational and neural basis of cognitive control: Charted territory and new frontiers. Cognitive Science. 2014;38(6):1249–1285. doi:10.1111/cogs.12126.

48. Tversky A, Kahneman D. Judgment under Uncertainty: Heuristics and Biases. Science. 1974;185(4157):1124–1131.

49. Simon HA. A Behavioral Model of Rational Choice. The Quarterly Journal of Economics. 1955;69(1):99–118. doi:10.2307/1884852.

50. Cappelletti M, Butterworth B, Kopelman M. Spared numerical abilities in a case of semantic dementia. Neuropsychologia. 2001;39:1224–1239.

51. CANTAB® [Cognitive assessment software].; 2017. Available from: `www.cantab.com`.

52. Unity 3D; 2017. Available from: `https://unity3d.com/`.

53. R Core Team. R: A Language and Environment for Statistical Computing; 2017. Available from: `https://www.r-project.org/`.

54. Bates D, Mächler M, Bolker B, Walker S. Fitting Linear Mixed-Effects Models Using {lme4}. Journal of Statistical Software. 2015;67(1):1–48. doi:10.18637/jss.v067.i01.

Competing Interests

The authors declare no competing interests.

Supporting Information

S1 Appendix The Knapsack Problem and Instance Complexity

S2 Appendix *Gecode* and *MiniSat*$^+$ complexity measures

S3 Appendix Instance Sampling

S4 Appendix Continuous parameterisation of Instance Complexity in the Knapsack Decision Problem

S5 Appendix CANTAB tasks. Description of the Cambridge Neuropsychological Test Automated Battery (CANTAB®)

S1 Table Mixed effects logistic regressions on computational performance in the Knapsack Decision Task.

S2 Table Mixed effects logistic regressions on computational performance in the Knapsack Optimisation Task.

S3 Table Mixed effects linear regressions on the time spent on an instance in the Knapsack Optimisation Task.

S4 Table Mixed effects linear regressions on the other performance measures in the Knapsack Optimisation Task..

S5 Table Effect of the number of item-subsets that perform better that the current selection of items on time before the next click.

S6 Table Pearson correlation between knapsack task performance and cognitive abilities.