A Formal Theory of Relevancy in Problem Solving

Johan Kwisthout, Institute for Computing and Information Sciences, P.O. Box 9010, 6500GL Nijmegen, The Netherlands, johank@cs.ru.nl

TECHNICAL REPORT – DECEMBER 2011

Abstract

When computer scientists discuss the computational complexity of, e.g., finding the shortest path from building A to building B in some town or city, their starting point typically is a formal description of the problem at hand, e.g., a graph with weights on every edge where buildings correspond to vertices, routes between buildings to edges, and route-distances to edge-weights. Given such a formal description, either tractability or intractability of the problem is established, by proving that the problem enjoys a polynomial time algorithm, respectively is NP-hard. However, this problem description is in fact an abstraction of the actual problem of being in A and desiring to go to B: it focuses on the relevant aspects of the problem (e.g., distances between landmarks and crossings) and leaves out a lot of irrelevant details.

This abstraction step is often overlooked, but may well contribute to the overall complexity of solving the problem at hand. For example, it appears that "going from A to B" is rather easy to abstract: it is fairly clear that the distance between A and the next crossing is relevant, and that the color of the roof of B is typically not. However, when the problem to be solved is "make X love me", where the current state is (assumed to be) "X doesn't love me", it is hard to agree on all the relevant aspects of this problem.

In this paper a framework is presented in order to capture the notion of relevance in finding a suitable problem representation. It is shown that it is in itself intractable in general to find a minimal relevant subset of all features that might or might not be relevant to the problem. The paper aims to contribute a formal notion of 'relevancy' in problem solving, in order to be able to separate 'easy to abstract' from 'hard to abstract' problems and discuss individual differences in the abstraction task, e.g., when experts in a particular domain are compared with novice problem solvers.

Keywords: relevancy, abstraction, computational complexity, formal modeling, problem solving.

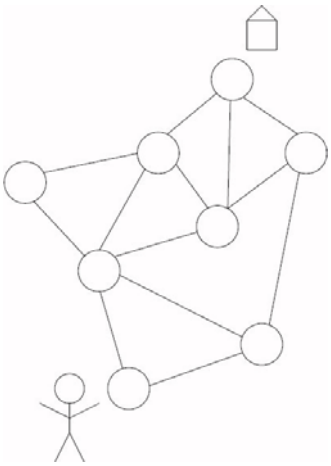
1. Introduction

One of Leonhard Euler's most famous contributions to mathematics was his treatment of the *Seven Bridges of Königsberg-problem*[1]. The city of Königsberg is set on both sides of the Pregel River. Both parts of the city, and two islands in the river, are connected using seven bridges. Euler was asked whether one was able to go for a stroll on a Sunday afternoon, passing each bridge *exactly once* and returning where one started the trip. Euler proved that the answer to this question was 'no' — no such tour exists (Euler, 1741).

Euler's main contribution, arguably, was not in the actual result, but in the way he tackled the problem. He quickly realized that the route *between* the consecutive crossing of two bridges was irrelevant to solving the problem. The only relevant aspects of the problem are to be found in the topology of the bridges: which bridge connects which landmass. While solving the actual problem, Euler laid out the foundations of graph theory: in nowadays terms, each landmass corresponds to a vertex, and each bridge to an edge connecting vertices. The actual problem could be *abstracted* into a graph problem: is there a tour connecting all vertices and traversing each edge exactly once?

However clever and thoughtful, Euler's treatment of the problem is quite typical of the way humans solve problems: by focusing on the relevant aspects of the problem only, dismissing details that are irrelevant. Nevertheless, we do make mistakes. We sometimes include aspects that are not relevant (e.g., we might conclude that the distance between two bridges is relevant), leading to suboptimal representations; we sometimes weed out too many aspects (e.g., we might dismiss multiple bridges between two landmasses, focusing on a connectivity problem instead), leading to representations that may not lead to a correct solution.

[1] http://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg



Input: Graph G = (V,E); weights associated with each edge; designated vertices S and T

Output: sequence of edges (S, X), ..., (Y, T) connecting S to T with minimal total weight

Figure 1: A problem and its formal description. Here the computational problem is to find a shortest path between two designated points S and T in a graph. It is formalized as an input-output mapping.

In general, it appears that the problem of *finding a relevant abstraction* significantly contributes to the overall complexity of solving the actual problem. Despite this observation, when studying a particular problem, a computational complexity analysis typically assumes that a relevant abstraction of the problem is readily available. For example, assume one wants to find ones way in an unknown city; in particular, desires to travel from one's current location to the down-town hotel, using the shortest possible route. This 'real-world problem', which we will denote with Π_R, naturally translates into the computational problem Π_C of finding the shortest path in a graph with weighted edges and designated starting and ending points, as depicted in Figure 1. Such an abstract computational problem, cast into an input-output mapping, can then act as the starting point of a computational complexity analysis. We encode arbitrary instances of Π_C into input strings for a particular computational device (e.g., a Turing machine) and investigate whether the corresponding outputs can be computed efficiently, e.g., in time, polynomial in the input size. In order for the problem to be tractably solvable, we require that the encoding is reasonable (we do not want to artificially increase the input size) and that the computation is feasible (Figure 2).

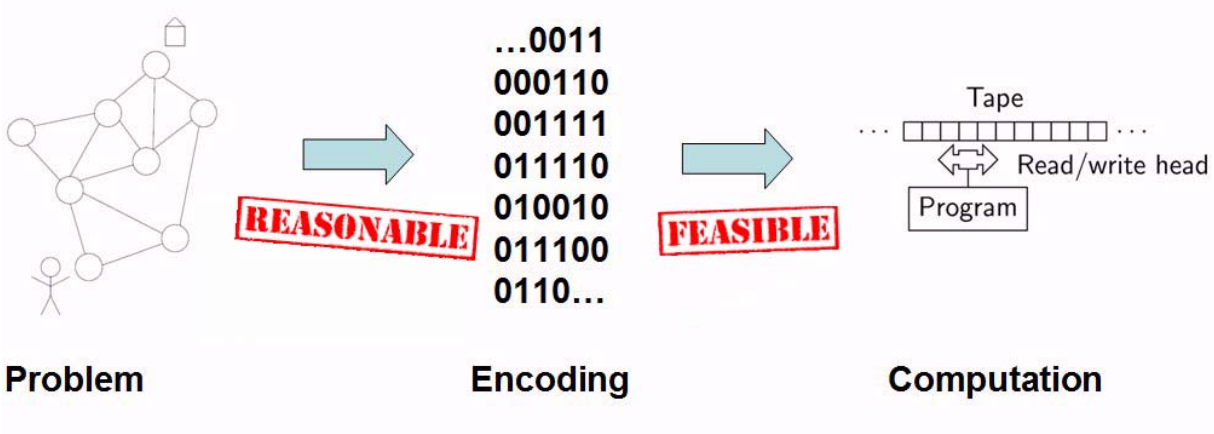


Figure 2: The classical computational complexity view on solving problems. A particular problem instance, expressible as a formal input-output mapping, is to be encoded in some computer-readable encoding; typically, but not necessary, a string of binary numbers. The encoded instance is than fed as input to a computing device like a Turing Machine. In order for the problem to be solved tractably, we demand that the encoding is reasonable*, i.e., that we do not 'blow up' the size of the instance, and that the computation is* feasible*, i.e., that the time needed to compute the output corresponding to a particular input, takes time, polynomial in the input size.*

In this classical notion of computational complexity, however, we deal with *abstract* problems only, i.e., in order to assess the computational complexity of a particular problem, we do not take the abstraction step itself into account. However, there is increasing evidence (see, e.g., Ash, Cushen, & Wiley, 2009 for an overview) from psychology that finding a useful representation of a problem can be as hard as computing a solution to the (representation of the) problem. In order to construct and analyze computational models of cognitive processes like problem solving (in a broad sense, like intention recognition (Baker, Saxe, & Tenenbaum, 2009), visual perception (Cavanagh, 2011), analogy (Keane, 1988) and others), we need to address the issue of representation as well as the issue of computation.

This observation inspires an *enhanced* notion of computational complexity (Figure 3), where the abstraction step is explicit. The starting point of a complexity analysis here no longer is the abstract computational problem Π_C, but the real world problem Π_R: for this problem to be feasibly solvable, we not only require reasonable encodings and tractable computations, but also *relevant abstractions*. As an example, in the find-our-hotel problem, the distance between crossings is typically a relevant characteristic that should be included

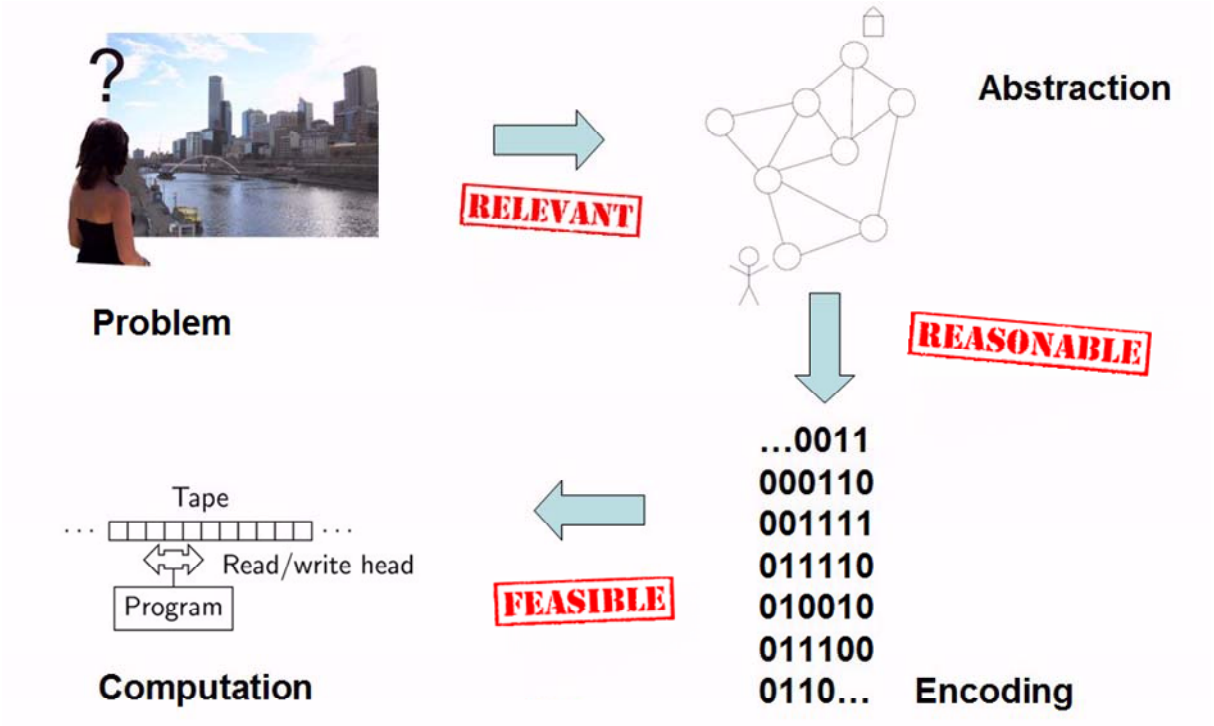


Figure 3: The enhanced *computational complexity view on solving problems. Here we make an explicit distinction between the* actual *problem in the real world, i.e., finding one's way in an unknown city, and the abstract computational problem: finding a shortest path between two points. In addition to the previous demands on the encoding and the computation, we also demand that the abstraction captures the* relevant *aspects of the problem in the real world, e.g., distances between crossings, and abstracts away from many typically irrelevant details.*

in the abstract computational problem. The color of the roof of the hotel is typically *not* relevant and thus should *not* be included in the abstraction. It will be clear that we can 'mess up' with the complexity analysis by inflating the problem instances, in a similar way as we can use unreasonable encodings, such that the resulting running time of the algorithm solving the problem—which is measured as a function on the *input size*—no longer reflects the actual difficulty of solving the problem.

In the remainder of this paper, we introduce a formal theory of relevancy in Section 2. In Section 3 we discuss hard-to-abstract and easy-to-abstract problems, and in Section 4 the theory is put into context by discussing related research on representation, abstraction, and (insight) problem solving. In Section 5 we conclude and propose further research. A formal NP-hardness proof of the abstraction problem is given in the appendix.

2. A Formal Theory of Relevancy

In the previous section we introduced an enhanced view on computational complexity analysis, which makes the abstraction step from the real world problem Π_R to abstract computational model Π_C explicit. Informally, this abstraction step seeks to isolate the *relevant* aspects of Π_R. In this section we introduce a formal theory of this abstraction step.

In the context of this paper, we see solving a problem Π_R simply as a transition from a problem-state to a solution-state, given some function f_Π mapping problems to solutions[2]. For example, the problem state might denote "Standing in front of King's Cross Railway station" and the solution state might denote "Walking on Trafalgar Square". Alternatively, these states might respectively denote "Sitting at the table, frowning and holding an unsolved nine-dot-puzzle" and "Standing up from the table, smiling and waving with a solved nine-dot-puzzle" or the problem and solution state may even refer to "X doesn't love me", respectively "X *does* love me". Note that the descriptions of these states are limited and can be extended *ad infinitum.* When I am actually standing in front of King's Cross, I observe that there is a yellow car passing by, a business man is talking into his cell phone, the sun is shining and someone is just buying a newspaper at the local kiosk. Apart from everything that I observe through my senses, there is another infinite amount of information describing the current state, for example that it is raining in Uganda, that someone in Brooklyn just emptied her glass, and that the amount of radiation from the Pleiades is slightly above yesterday's level.

The bottom line here is that there is a massive amount of properties describing each state, yet hardly any of this information is relevant here. It is difficult to imagine a situation

[2] We are aware of the fact that this is just one notion of *solving a problem*. Other notions may include 'explain why s is a solution to $f_\Pi(p)$' or 'find a function f_Π matching s and p' or 'find a p that matches some given constraints'. In the context of this paper, we will assume that s and p are fully known and observable and leave other notions of *solving a problem* to future work.

where the amount of radiation from the Pleiades influences the way I go from King's Cross to Trafalgar Square. In our problem solving, we abstract from all of these irrelevant properties immediately and focus on what is relevant, e.g., the underground train system connecting nearby stops, or the current time—if it's 3 AM, we will probably need to take a taxi. Furthermore, while we assume that there is exactly *one* problem state (i.e., the state that I am currently in), many states qualify as a solution: me standing on Trafalgar Square with either bus 5, 11 or 15 riding by all are valid solution states.

We formalize "problem solving in the real world" as finding a transition from one point p in a state space (describing the initial problem state) to another point s in a solution region S of the state space, where the dimensions of the state space describe properties of p and s. Some of these dimensions may correspond to properties we typically regard as irrelevant, like "current amount of radiation from the Pleiades". Other dimensions correspond to obviously relevant properties, like "distance to the nearest underground station".

We assume that the number of dimensions N is massive, and that each dimension $d \in N$ can take arbitrary values. Problem solving can then be formally defined as the transition from $p \in \mathbb{N}^N$ to $s \in S \subseteq \mathbb{N}^N$, where $\mathbb{N}$ denotes the set of natural numbers. The transition typically is in the form of some *action plan*, i.e., a set of intermediate states that allows us to move from p to s. Observe that N may—and will—be *very* large. Typically, only a small subset of N corresponds to relevant properties. We denote that subset with M, and assert that the actual transition from p to s is *independent* of values of dimensions outside M: whether the radiation from the Pleiades sums up to γ_1 or γ_2 will not influence my action plan, consisting of me taking the first available tube to Charing Cross. Our abstraction problem now is the problem of finding such a *relevant subset* $M \subset N$. We formalize the above informal notions into computational problems as follows. By the notation $x^{\downarrow M}$, we refer to the vector that is obtained from x by omitting all dimensions of x that are not in M.

PROBLEM SOLVING

Input: A input vector $p \in \mathbb{N}^N$, denoting the problem-state; an output region $S \subseteq \mathbb{N}^N$, denoting the solution-space, such that $f_\Pi(p) = S$ for a particular function f_Π mapping instances of a problem Π to solutions to these instances; an objective function o_Π such that $o_\Pi(s) = 1$ if $s \in S$, and $o_\Pi(s) = 0$ otherwise.

Output: A sequence of vectors $A_1,\ldots,A_k \in \mathbb{N}^N$, defining an action plan to obtain $s \in S$ from p.

ABSTRACTION

Input: A function f_Π mapping instances of a problem Π to solutions to these instances, with corresponding objective function o_Π.

Output: The smallest non-empty subset $M \subseteq N$ such that for every $s \in \mathbb{N}^N$, $o_\Pi(s) = o_\Pi(s^{\downarrow M})$.

Note that in the context of this paper we are not particularly interested in the action plans themselves, nor in how they are to be constructed or executed; our interest lies in the abstraction problem, i.e., obtaining the relevant characteristics of the problem and omitting the non-relevant details. Furthermore, we assume that both p and s are fully observable and that every problem actually has a solution, i.e., S is non-empty.

In the next section we will discuss particular instances of the abstraction problem and discuss why some instances of the abstraction problem appear to be fairly easy, why others appear to be hard. We will introduce the notion of *expected relevancy* to explain which dimensions are typically considered relevant, and others are considered irrelevant (or not considered at all), and how this notion can help in discriminating hard and easy to abstract problems.

3. Hard and Easy Abstraction Problems

For some problems, finding a good abstraction appears to be fairly easy. Most of us will agree that, for solving the problem of finding the shortest path to the down-town hotel, distances between crossings and landmarks are relevant characteristics of the problem that should be preserved in the abstraction, and the structure of the tiles in the pavement are irrelevant details that should be left out. It is not too hard to settle on a subset *M* of relevant dimensions in this problem. The problem generalizes well to, e.g., finding the shortest path to the library.

Other problems are much harder to abstract. One such problem might be: "*Make X love me*" where the (presumed) begin-state *p* is: "*X doesn't love me*" and the desired end state is in the region *S* corresponding with "*X* does *love me.*" Here, there is for many dimensions *d* general disagreement whether $d \in M$ or not; arguably, there are many relevant dimensions. We will have a hard time explaining the relevant dimensions to others; in fact, it will be an educated guess at best. It is not well understood what the relevant dimensions are here. The problem does not well generalize to "*Make Z love me*", as typically the set of relevant dimensions will be different.

Are there problems that are *inherently* hard to abstract? In the context of the proposed formal framework, the answer can be an unambiguous '*yes*'. We show in the appendix, using a computational complexity analysis, that it is NP-hard[3] to decide in general that a particular dimension is relevant, and thus that no polynomial time algorithm can solve the **ABSTRACTION** problem in polynomial time unless P = NP. To discuss *why* the "*Make X love me*" problem is harder to abstract than "*Find shortest path to hotel*" we investigate when a dimension is typically considered to be irrelevant.

It appears that there are two arguments to dismiss a dimension as irrelevant:

[3] We assume that the reader is familiar with the notions P and NP for computational problems that can be decided, respectively for which a candidate solution can be verified, in polynomial time; the notion of NP-hardness, indicating presumed intractability of the computational problem, and the assumed inequality of P and NP. We refer the reader to, e.g., Garey and Johnson (1979) for a thorough discussion of these concepts.

1. Variation along this dimension changes the outcome of the problem solving activity only by a tiny amount (i.e., $f(S^{\downarrow d = di}) \approx f(S^{\downarrow d = dj})$ for designated values d_i and d_j of the dimension d). For example, the exact placement of the hotel's entrance may in- or decrease the distance by a few meters, yet we typically don't regard that as relevant.

2. Variation along this dimension may change the outcome of the problem solving activity considerably, but only for very exceptional values of d. The earlier mentioned *color-of-roof* property is, for example, only relevant in highly exceptional cases, like when the roof is actually made of (brilliantly shining) gold.

Given these observations, we now define the **expected relevancy** δ_d of a dimension as the sum over all values of this dimension of the *product* of the probability that this dimension has a particular value d_i and the amount of deviation from the outcome when compared to a (non-specified) default 'reference' value:

Definition

Given a function f_Π mapping instances of a problem Π to a solution; let d be a dimension of interest in Π, with d_{ref} as its default or stereotype value. The *expected relevancy* δ_d of d is now defined as $\delta_d = \sum_{di} \Pr(d = d_i) \times \text{abs}(f(S^{\downarrow d = dref}) - f(S^{\downarrow d = di}))$.

Note that the expected relevancy δ_d is a *subjective* measure that is typically *estimated* by the problem solver. Given this formal notion of expected relevancy, we suggest that one—as a heuristic—only includes dimensions in a abstraction that have a high expected relevancy. Naturally, making such an abstraction is easy when there are only few dimensions with a high expected relevancy, and hard when there are (many) dimensions with a high expected

relevancy ("*Everything appears to be relevant!*") and when there are (many) dimensions for which the expected relevancy is difficult to estimate ("*I don't know what is relevant*!").

We should emphasize that we do not propose or even suggest that a problem solver actually consciously *computes* expected relevancies. In a similar way as our cognitive processes often behave roughly according to the laws of probability theory, without the brain actually performing probabilistic inference before making a decision (Chater & Oaksford, 2008), abstractions can be made roughly according to the expected relevancy without (consciously) computing them. We *do* suggest that expected relevancy may be a measure to *describe* why some problems are harder to abstract than others; likewise, we suggest that it may be a useful way of *explaining* why experts have an easier task than novices in finding an abstraction.

4. Discussion

The ability to solve problems is often considered to be one of the most complex intellectual abilities of mankind; it is therefore not surprising that studying (computational models of) problem solving is a key topic in cognitive science, perhaps with the seminal work of Newell and Simon (1972) as most impressive example. Even when we restrict ourselves to a narrow definition of problem solving (finding an action plan from a problem state to a solution state) there is a wild variety in the difficulty of problems to be solved. Naturally, some problems are more difficult than others. Funke (1991) contrasts simple from complex problems using criteria like the transparency of the problem and solution definitions on the one hand, and complexity of the problem in terms of number and connectivity of variables on the other hand. Complex problems may have intransparent definitions and large and highly connected problem spaces.

Here, some of the criteria for complexity appear to stem from *representing* the problem, while some others from *solving* the already represented problem. For some

problems, it is both easy to find a suitable formal representation (i.e., a suitable abstraction of the real world problem) *and* solve the problem; finding the shortest path to the downtown hotel is such a problem. Finding a winning strategy in Go is fairly easy to cast into a formal computational problem, but intractable to solve; other problems are easy to solve once we have a suitable representation.

As already noticed by Newell and Simon (1972), some problem solving processes are incremental in nature. They require only fairly routine or analytic steps to solve; the problem solver knows *how* the problem is to be solved, he or she just needs to apply the needed steps. While this may require in itself considerably intellectual effort, this type of problem solving can be characterized by a *steadily increasing Feeling-of-Warmth* (Metcalfe, 1986). Other problem solving processes, in contrast, involve non-incremental, discontinuous, or creative steps to "pass beyond a barrier". These problems are often denoted as *insight problems* (Smith, 1995; Sternberg & Davidson, 1995; Chu & MacGregor, 2011) as they typically require some insight or *aha-erlebnis* to solve. In such problems, Metcalfe (1986) found a *sharp increase* in Feeling-of-Warmth towards the end of the problem solving process. In such insight problems, the problem solver typically starts with an inappropriate representation, and the problem can only be solved when the problem solver realizes that the problem representation should be changed in order to 'break the barrier'. Breaking this barrier can, according to the Representational Change Theory (Ohlsson, 1992), be done by changing the representation, for example by relaxing some constraints.

Insight problems thus are typically problems for which solving them is fairly easy, once the correct representation is found, i.e., if we have an appropriate abstraction that leaves out the irrelevant details but includes all the aspects of the problem that are relevant in solving it. An example is a variant of the so-called *Nine Dots Problem*. Assume you where given a sheet of paper, with nine black dots on it, ordered in a three-by-three square (Figure 4a), and

are asked whether it is possible to connect the nine dots using only *three* consecutive straight lines. While the notion of problem solving here is slightly different as the solution itself is required (rather than a path to a given solution), the act of abstracting the problem is similar. We quickly (and likely, unconsciously) classify the way in which the dots are ordered as relevant for solving the problem, and the thickness of the paper, the color of the dots, the exact placement of the dots on the paper, the current time, weather conditions, and NASDAQ-index as irrelevant. What makes this problem hard to solve is that we typically regard one aspect of the problem as irrelevant why it is in fact *crucial* in solving the problem, namely, the size of the dots. In typical mathematics-like problems, a 'dot' refers to a *point*, a zero-dimensional point in space. We abstract the problem by seeing it as nine *points* that need to be connected. In this particular problem, however, the dots *have* volume, and one can connect them using three straight lines if we don't connect them at their center, but (like in Figure 4b) make use of the volume of the dots. It is, however, fully explainable that we did not (initially) include the volume of the dots as a relevant aspect that needed to be included in the representation, as it has a low expected relevancy.

So, in the context of this paper, we see *relevancy* as an attribute of an aspect of the problem state such that this aspect has high relevancy if deviation of the actual value of the aspect is likely to considerably change the outcome of the problem solving activity; we postulate that problem solvers form representations of the problem state by including the

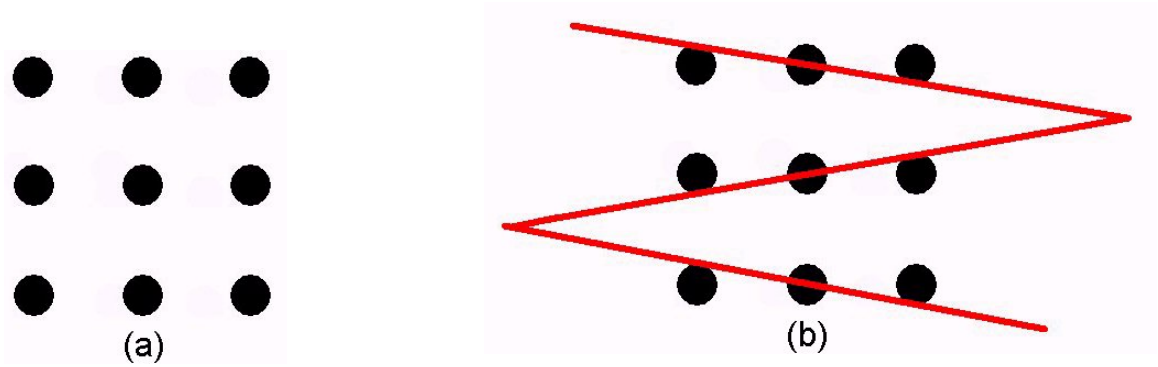


Figure 4: An example of the Nine-Dots-Puzzle (a) and a possible solution (b). Note that the solution depends on the dots having volume, rather than being dimensionless points in space.

relevant aspects and (thus) dismissing the irrelevant aspects of that state. This notion of relevance corresponds to, e.g., Wilson and Sperber's approach where "*an input is relevant to an individual when its processing in a context of available assumptions yields (…) a worthwhile difference to the individual's representation of the world*" (Wilson and Sperber, 2004, p. 608). It is also related to Gorayska and Lindsay's goal-oriented approach, where an aspect is relevant towards a goal if there is some action plan from the current state to the goal where this aspect plays an essential role (Gorayska & Lindsay, 1993). In a somewhat different context, Müller, van Rooij, and Wareham (2009) proposed a related notion of a *relevant set of transformations* in similarity judging.

5. Conclusion

Whether a problem solving situation is trivial or complex depends on many factors; some of which can be classified as representational factors, other as computational factors. The *computational complexity* of (an abstraction of) a problem is a well-studied area, with its own conferences, research groups, and the famous $ 1,000,000 question[4] whether the classes P and NP are equal or different. The *representational complexity* of a problem – how hard is it to

[4] The Clay Mathematics Institute has denoted this open problem as one of the "Millennium Prize" problems and offers a $ 1,000,000 prize for anyone proving either P = NP or (much more likely) P ≠ NP.

find a suitable formal representation of a problem – has received much less attention from computer science. In this paper we proposed one possible framework for studying the representational complexity of a problem and related it to intuitive notions of hard and easy to abstract problems using the *expected relevancy* measure.

While the expected relevancy is a *subjective* measure on a problem aspect, it can be shown that, within this framework, there exist problems that are *inherently* hard to abstract. In future work it would be very interesting if these hard problems can be characterized in some way such that they can be formally separated from the easy problems. Furthermore, while it is NP-hard to find a *minimal* subset of relevant aspects, we do not have formal results yet on approximation (finding a subset that is guaranteed to be *almost* minimal) or randomization (finding a subset that is minimal with a *high probability*, but may be way off in extreme cases) strategies of abstraction.

Furthermore, it is very likely that the expected relevancy is in one way or another *conditioned* on previous experiences, hints, analogous examples[5], or other knowledge related to the problem. We did not touch on these aspects in this paper.

Lastly, we focused on only a specific type of problem solving, where both problem state and solution space are fully observed and we are interested in an *action plan* connecting both. The Nine-Dot-Puzzle example suggests that the theory may be expanded to other types of problem solving; we leave that for future work.

Acknowledgements

This paper is based on the presentation that the author delivered at the Dagstuhl Seminar on *Computer Science & Problem Solving: New Foundations* and greatly benefitted from the discussions there. The author especially wishes to thank Todd Wareham and Iris van Rooij for

[5] For example (Gick and Holyoak, 1980) the analogy between the Attack Dispersion story (an army must be led to a fortress surrounded by mines that go off when a large army passes over them; yet the full army is needed to take the fortress) and Duncker's Radiation Problem (a tumor can only be defied by an amount of radiation that damages also the healthy tissue between the source of the rays and the tumour)

valuable discussions and comments on earlier versions of this paper. He wishes to thank Bas Maes for designing the figures in the paper.

References

Ash, I. K., Cushen, P. J., & Wiley, J. (2009). Obstacles in investigating the role of restructuring in insightful problem solving. *Journal of Problem Solving* 2 (2), 6–41.

Baker, C. L., Saxe, R., & Tenenbaum, J. B. (2009). Action understanding as inverse planning. *Cognition* 113, 329–349.

Cavanagh, P. (2011). Visual cognition. *Vision Research* 51 (3), 2011, 1538–1551.

Chater, N., & Oaksford, M. (2008). *The probabilistic mind: Prospects for Bayesian cognitive science*. Oxford University Press, Cary, NC.

Chu, Y., & MacGregor, J.N. (2011). Human Performance on Insight Problem Solving: A Review. *Journal of Problem Solving* 3 (2), 119–150.

Euler, L. (1741). Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae* 8, 128–140. Retrieved from http://www.math.dartmouth.edu/~euler/pages/E053.html.

Funke, J. (1991). Solving Complex Problems: Exploration and Control of Complex Social Systems. In R.J. Sternberg and P.A. Frensch (Eds.): *Complex Problem Solving: Principles and Mechanisms*. Lawrence Erlbaum Assocates, Hillsday, NJ.

Garey, M.R., & Johnson, D.S. (1979). *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, CA.

Gick, M.L., & Holyoak, K.J. (1980). Analogical problem solving. *Cognitive Psychology* 12, 306–355.

Gorayska, B, & Lindsay, R. (1993). The roots of relevance. *Journal of Pragmatics*, 19 (4), 301–323.

Keane, M.T., (1988). *Analogical problem solving*. Halsted Press, Oxford, UK.

Metcalfe, J. (1986). Premonitions of insight predict impending error. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 12 (4), 623–634.

Müller, M., van Rooij, I., and Wareham, T. (2009). *Similarity as tractable transformation.* Proceedings of the 31st Annual Conference of the Cognitive Science Society pp 50–55.

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

Ohlsson, S. (1992). Information-processing explanations of insight and related phenomena. In M. Keane & K. Gilhooly (Eds.), *Advances in the Psychology of Thinking* (pp 1–44). Harvester-Wheatsheaf, Hemel Hempstead, UK.

Smith, S.M. (1995). Fixation, incubation, and insight in memory, problem solving, and creativity. In S.M. Smith, T.B. Ward, & R.A. Finke (Eds.), *The Creative Cognition Approach* (pp. 135–155). MIT Press, Cambridge, MA.

Sternberg, R.J., & Davidson, J.E. (Eds.) (1995). *The Nature of Insight*. MIT Press, Cambridge, MA.

Wilson, D. & Sperber, D. (2004). Relevance Theory. In Horn, L.R. & Ward, G. (Eds.) *The Handbook of Pragmatics* (pp. 607–632.). Blackwell, Oxford, UK.

Appendix

In the Appendix, we argue that it is in general intractable to solve the **ABSTRACTION** problem defined in Section 3, i.e., to find the smallest relevant subset *M* of the set of all dimensions *N*. In particular, we show that it is NP-hard in general to decide whether a particular dimension is relevant. We start by defining two computational decision problems, and show that they are encapsulated in the **ABSTRACTION** problem mentioned above. Then we proceed to prove NP-completeness, respectively co-NP-completeness, of these decision problems, thus establishing the above mentioned results. We assume that the reader is familiar with the complexity classes P, NP and co-NP, and with intractability proofs in general. We refer to textbooks like Garey and Johnson (1979) for more background.

We call a variable x_i *redundant* in a Boolean formula φ if x_i's truth assignment does not influence the characteristic function[6] $\mathbf{1}_\varphi$ of φ. In other words, x_i is redundant in φ if $\mathbf{1}_\varphi(x_i = 0) = \mathbf{1}_\varphi(x_i = 1)$. In contrast, x_i is *relevant* in φ if $\mathbf{1}_\varphi(x_i = 0) \neq \mathbf{1}_\varphi(x_i = 1)$, i.e., x_i's truth assignment *does* influence the characteristic function. These notions induce the following decision problems.

ISA-RELEVANT VARIABLE

Input: Boolean formula φ with *n* variables, describing the characteristic function $\mathbf{1}_\varphi$: $\{0,1\}^n \rightarrow \{1, 0\}$, designated variable $x_i \in \varphi$.

Question: Is x_i a relevant variable in φ, i.e., is $\mathbf{1}_\varphi(x_i = 1) \neq \mathbf{1}_\varphi(x_i = 0)$?

ISA-REDUNDANT VARIABLE

Input: Boolean formula φ with *n* variables, describing the characteristic function $\mathbf{1}_\varphi$: $\{0,1\}^n \rightarrow \{1, 0\}$, designated variable $x_i \in \varphi$.

Question: Is x_i a redundant variable in φ, i.e., is $\mathbf{1}_\varphi(x_i = 1) = \mathbf{1}_\varphi(x_i = 0)$?

It can be readily shown that these problems are closely related to the **ABSTRACTION** problem: every variable in φ translates to a (binary valued) dimension in $\mathbb{N}^N$ with *d* as designated dimension corresponding to x_i; f_Π outputs a state corresponding to a satisfying truth instantiation; the characteristic function $\mathbf{1}_\varphi$ then corresponds with the objective function

[6] The characteristic function of a Boolean formula φ, denoted by $\mathbf{1}_\varphi$, maps truth assignments to φ to {0, 1}, such that $\mathbf{1}_\varphi(\mathbf{x}) = 1$ iff. **x** denotes a satisfying truth assignment, and 0 otherwise.

o_Π. We denote the problem state p as an arbitrary vector in this space and the solution space S as a region in $\mathbb{N}^N$ corresponding with the satisfying truth instantiations. The smallest subset M of N such that $o_\Pi(s) = o_\Pi(s^{\downarrow M})$ now contains only dimensions that correspond to relevant variables; any other dimension $d' \in N \setminus M$ corresponds to a redundant variable. Naturally, any algorithm that solves **ABSTRACTION** can also be used to decide **ISA-RELEVANT VARIABLE** and **ISA-REDUNDANT VARIABLE**: from φ, construct an **ABSTRACTION** instance as described above and compute M; x_i is relevant if and only if $x_i \in M$, otherwise x_i is redundant. As $M \subseteq N$, this computation can be done in polynomial time. As we will show shortly that deciding **ISA-RELEVANT VARIABLE** and **ISA-REDUNDANT VARIABLE** is NP-complete, respectively co-NP-complete, it follows that computing the relevant subset of dimensions (or deciding whether a particular dimension is relevant) for f_Π is intractable, unless P = NP.

Theorem: **ISA-RELEVANT VARIABLE** and **ISA-REDUNDANT VARIABLE** are NP-complete, respectively co-NP-complete.

Proof. To prove NP-completeness, we reduce **ISA-RELEVANT VARIABLE** from the well-known NP-complete **SATISFIABILITY** problem (given a Boolean formula φ with n variables; is φ satisfiable?). Membership of NP follows as we can verify in polynomial time a certificate, consisting of a tuple of two truth assignments $(\boldsymbol{x}, \boldsymbol{x'})$ such that $\mathbf{1}_\varphi(\boldsymbol{x}) \neq \mathbf{1}_\varphi(\boldsymbol{x'})$. NP-hardness is proven as follows. Let ψ be an instance of **SATISFIABILITY** with n variables and let $\varphi = \psi \wedge x_{n+1}$.

→ If x_{n+1} is a relevant variable, then ψ is satisfiable, as there exists by definition at least one truth assignment to φ such that $\mathbf{1}_\varphi(\psi \wedge x_{n+1} = \text{true}) \neq \mathbf{1}_\varphi(\psi \wedge x_{n+1} = \text{false})$, and given that $\psi \wedge x_{n+1} = \text{false}$ always evaluates to false, we have that ψ is necessarily satisfiable.

← if ψ is satisfiable, then x_{n+1} is a relevant variable as $\mathbf{1}_\varphi(\psi \wedge x_{n+1} = \text{true}) \neq \mathbf{1}_\varphi(\psi \wedge x_{n+1} = \text{false})$ for any satisfiable truth instantiation to ψ, given that $\psi \wedge x_{n+1} = \text{false}$ always evaluates to false.

As we can obviously construct the above reduction in polynomial time, that proves that **ISA-RELEVANT VARIABLE** is NP-complete. The co-NP-completeness proof of **ISA-REDUNDANT VARIABLE** is almost similar, save that we reduce from the co-NP-complete

problem **TAUTOLOGY** (given a Boolean formula φ with *n* variables; is φ a contradiction?) and we prove membership in co-NP by providing the same certificate as above, but now used as a counterexample. *(Q.E.D.)*