Journal of Problem Solving

SPECIAL ISSUE

The Roles of Internal Representation and Processing in Problem Solving Involving Insight: A Computational Complexity Perspective[1]

Todd Wareham[1]

[1] *Memorial University of Newfoundland*

Correspondence:
Department of Computer Science,
Memorial University of Newfoundland,
St. John's, NL, Canada,
via email to harold@mun.ca.

In human problem solving, there is a wide variation between individuals in problem solution time and success rate, regardless of whether or not this problem solving involves insight. In this paper, we apply computational and parameterized analysis to a plausible formalization of extended representation change theory (eRCT), an integration of problem solving by problem space search and insight as problem restructuring which proposes that this variation may be explainable by individuals having different problem representations and search heuristic choices. Our analyses establish not only the intractability of eRCT in general, but also sets of restrictions under which eRCT-based problem solving can and cannot be done quickly. As such, our analyses both prove that several conjectures about what makes problem solving under eRCT possible in practice are incomplete, in the sense that not all factors in the model whose restriction is responsible for efficient solvability are part of the explanation, and provide several new explanations that are complete.

Keywords:
problem solving, insight, restructuring, computational complexity, ideal observer models

Acknowledgments:
The author would like to thank Iris van Rooij for many invaluable discussions and suggestions throughout the course and write-up of the research reported here (as well as the invitation to the 2011 Dagstuhl meeting that inspired this research in the first place) and the two anonymous reviewers, whose comments greatly improved the presentation and content of this paper. This research was supported by NSERC Discovery Grants 228104-2010 and 228104-2015.

1. INTRODUCTION

Much of human problem solving can be accounted for by Newell and Simon's classic problem space search model (Newell & Simon, 1972). This model underlies many subsequent and current models of human problem solving (Kaplan & Simon, 1990; Knoblich, Ohlsson, Haider, & Rhenius, 1999; Öllinger, Jones, & Knoblich, 2014; Ormerod, MacGregor, & Chronicle, 2002). In this model, a representation of a problem's givens and goals is chosen based on previous experience and search is then performed (possibly aided by heuristics) within the space of problem states associated with this representation until a state is encountered that satisfies the problem's goals, i.e., a goal state. This assumes that the representation initially chosen for the given problem is correct, in that it has goal states that can be reached by search within that representation's problem-state space. If this is not so, i.e., an impasse is encountered, restructurings (also known as insights) are necessary to modify the initial representation such that search can progress and possibly succeed (Ohlsson, 1992).

There are many types of problems that typically require insight to solve them, ranging from simple puzzles to complex word problems (see MacGregor and Cunningham (2009) and references therein). A basic type of such problems is the matchstick arithmetic problems (Knoblich et al., 1999), in which matchsticks are arranged in simple mathematical formulas stated in terms of Roman numerals. The goal is to move one or more matchsticks to convert a given incorrect formula into a correct formula. To solve such problems, previous experience working with mathematical formulas must be overridden in various ways, with each such overriding corresponding to an insight. Several matchstick arithmetic problems and their solutions are shown in Figure 1.

Experimental research on problem solving is typically divided into two camps, depending on whether or not the problems examined do or do not require insight to solve them (the so-called move and insight problems, respectively). There is great variation between individuals in how long it takes them to solve given problems or even if they can solve these problems at all (Newell & Simon, 1972; Ohlsson, 2011), with the most marked variation seen in insight problems

http://dx.doi.org/10.7771/1932-6246.1201

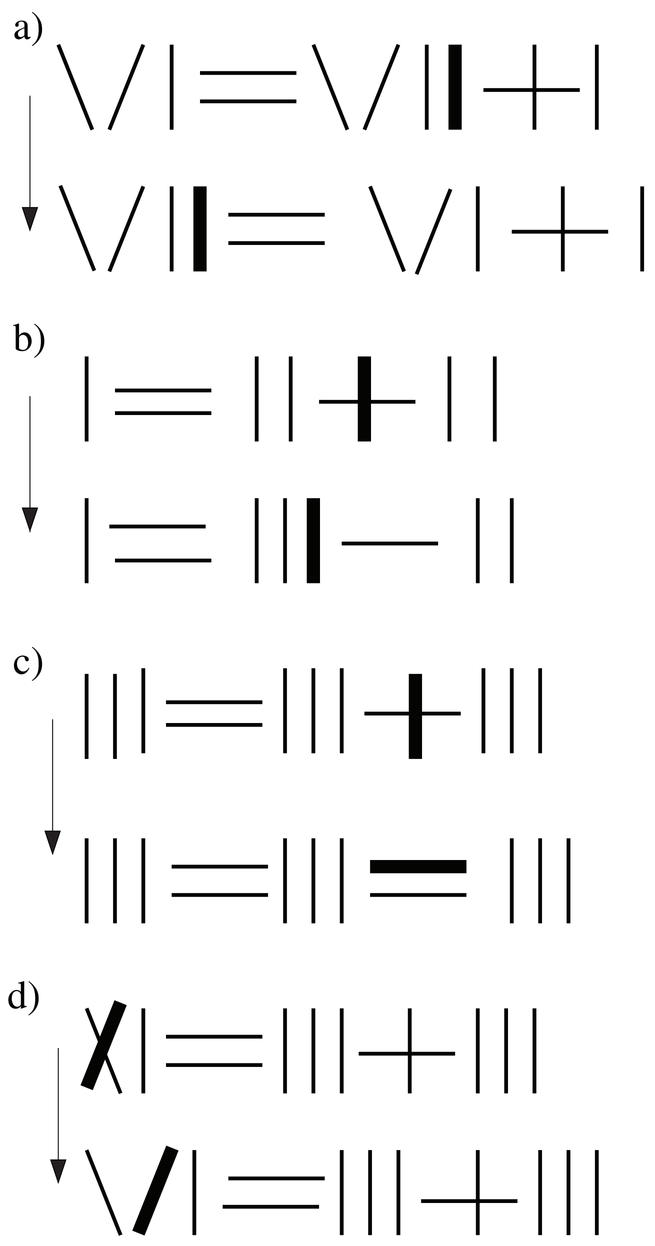

Figure 1.
Matchstick arithmetic problems. Parts (a–d) show the matchstick arithmetic problem types (1–4) given on p. 1538 of Knoblich et al. (1999). The moved matchstick is indicated by increased thickness.

(Chu & MacGregor, 2011; Danek, Wiley, & Ollinger, 2016). Hypothesized causes of this variation in move and insight problems are many and vary from the problem-specific (e.g., Kershaw & Ohlsson. 2004) to the general (Ash & Wiley, 2006; Batchelder & Alexander, 2011; Kaplan & Simon, 1990; Ohlsson, 2011; Öllinger & Knoblich, 2009; Ormerod et al., 2002).

An explanation of particular interest in this paper is the different ways in which individuals represent and/or process problems (Öllinger et al., 2014). These different ways are thought to arise from the differing prior experience of individual problem solvers (Batchelder & Alexander, 2011; Ohlsson, 2011). This explanation, if correct, would have wide-reaching implications, including the abolition of the traditional distinction between move and insight problems:

> …[T]here is no particular class of insight problems that necessarily requires a representational change; each problem can be solved without insight if the initial problem representation is adequate and the appropriate heuristics are available. (Ollinger et al., 2014, p. 267)

Untangling the effects of problem representation and processing on problem solution time and success using human experiments is notoriously difficult, especially in cases involving insight (Ash, Cushen, & Wiley, 2009; Batchelder & Alexander, 2011; Ohlsson, 2011). A potential aid in this endeavor would be ideal observer models—that is, one or more algorithms (possibly based on different restrictions on representations and/or processing) "that can explore the entire problem space and determine sequences of moves that lead from the starting state toward a goal state that minimizes a cost function such as the number of moves" (Batchelder & Alexander, 2011, p. 81). Such models would not only provide optimal-behavior baselines for comparison against human behavior (Batchelder & Alexander, 2011, p. 81) but would also illustrate (1) what factors can and cannot interact in individual human problem solvers to affect problem solution times and rates of success as well as (2) how such interaction might take place within a viable model of human problem solving.

In this paper, we will explore the space of possible ideal observer models for extended representation change theory (eRCT) (Öllinger et al., 2014), a theory that gives an integrated treatment of problem solving by both problem space search and insight. We shall do this in two stages—we will first formalize a plausible computational-level model of eRCT and then analyze this model using techniques from both classical (Garey & Johnson, 1979) and parameterized (Downey & Fellows, 1999) computational complexity to establish not only the intractability of this model in general but also those sets of restrictions under which efficient eRCT-based problem solving is and is not possible (with the former corresponding to sets of restrictions defining distinct classes of viable ideal observer models). As such, our analyses will both prove that several conjectured explanations of what makes problem solving under eRCT possible in practice are incomplete, in the sense that all factors in the model whose restriction are responsible for efficient solvability are not part of the explanation, and provide several new explanations that can be formally proven to be complete.

1.1. OVERVIEW

The remainder of this paper is organized as follows. Section 2 describes a framework for using computational and parameterized complexity analysis to evaluate the tractability and completeness of explanations of the solvability of problems under eRCT. Section 3 presents a computational-level model of problem solving under eRCT. Section 4 contains our computational complexity results, whose implications are discussed in Section 5. All mathematical definitions and proofs of results are given in Appendix A; this is to allow focus in the body of the paper on the model formulations and the implication of results derived for these models rather than the formal particulars of same. Finally, in Section 6, we give our conclusions.

2. A METHODOLOGY FOR VALIDATING EXPLANATIONS OF TRACTABILITY

Recall that our aim in this paper is to explore the space of viable ideal observer models for problem solving under eRCT, as an aid to both deriving algorithms for eRCT and assessing the validity (in terms of completeness) of conjectured explanations of the tractability of problem solving under eRCT. As answering these questions involves issues of the existence of particular types of algorithms implementing a cognitive theory, we will focus our analysis on Marr's computational level (Marr, 1981). That is, our analyses will pertain to the input–output mappings postulated in such a theory to explain the target cognitive processes.

We first explain what we understand as a failure of a theory to explain the observed speed of some cognitive process. A computational-level theory $\psi : I \rightarrow O$ is said to fail to explain the speed (minimally, the tractability) of a given (or any) cognitive (sub)process if there exists no efficient algorithm for tractably computing the function ψ. For example, algorithms whose running times are superpolynomial in the size of the input representations are generally inefficient for all but small input sizes. Such algorithms require an amount of time which cannot be upperbounded by any polynomial function n^c (where n is a measure of the input size and c is some constant). Examples are exponential-time algorithms, which require time that can (at best) be upperbounded by some exponential function c^n. To see that, for instance, exponential-time algorithms are impractical on even medium sized inputs, consider that 2^n is already more than the number of seconds in a year for $n = 25$.

As we will show, the eRCT of problem solving has the property that all algorithms for computing it have running times that are superpolynomial (i.e., exponential or worse) in the input size (for proofs refer to Section A.3 in Appendix A). Note that in problem solving, input size cannot generally be assumed to be small, as our knowledge of different problems can be quite rich and encoding such knowledge may lead to quite large representations overall. This means that an eRCT-based computational-level model of problem solving cannot by itself explain the tractability of the problem solving process; in fact, the very formulations of these models seem to contradict observed instances of efficient human problem solving.

The above does nonetheless allow that instances of efficient human problem solving under eRCT can be explained by enriching eRCT with some constraints. To identify such constraints, we will use techniques to prove fixed-parameter tractability of a function ψ for one or more *parameters* of inputs in I (see also Downey and Fellows, 1999).

Definition 1. Fixed-parameter (fp) tractability. *Let $\psi : I \rightarrow O$ be an input–output function with input parameters $k_1, k_2, \ldots, k_m$. Then ψ is said to be* fixed-parameter tractable *for parameter set $K = \{k_1, k_2, \ldots, k_m\}$ if there exists at least one algorithm that computes ψ for any input of size n in time $f(k_1, k_2, \ldots, k_m)n^c$, where $f(.)$ is an arbitrary computable function and c is a constant. If no such algorithm exists then ψ is said to be* fixed-parameter intractable *for parameter set K.*

In other words, a function is fp-tractable for a parameter set K if all superpolynomial-time complexity inherent in computing the function can be confined to the parameters in K. It then follows from the definition of fp tractability that if an intractable function ψ is fp-tractable for parameter set K then ψ can be efficiently computed even for large inputs, provided that function $f()$ in the runtime bound is well-behaved, e.g., $1.2^{k_1+k_2}$, and all the parameters in K are small. This means that if ψ is postulated as an explanation of the functional form of the input–output mapping computed by a given process, then the speed of that process in certain situations can be explained by postulating that the parameters in K are small exactly in those situations.

Given the above, we can explore the space of viable ideal observer models for eRCT relative to a set of possible constraints (each defined relative to a parameter of eRCT) by establishing the fp status of a computational-level model of eRCT relative to various combinations of these constraints. An fp-intractability result relative to a set K implies that there is no algorithm whose tractability is a function of K. Conversely, an fp-tractability result relative to K implies that there is at least one such algorithm. Note that each such K corresponds to a distinct class of cognitive mechanisms for efficiently implementing eRCT (namely, those cognitive mechanisms that can exploit restricted values of the parameters in K to lower their runtimes). Moreover, the algorithm originally used to establish fp tractability relative to K need not itself be a viable cognitive mechanism (as the algorithm only establishes fp tractability relative to K and thus need not be cognitively plausible or have the best possible runtime relative to K). Hence, in this paper, we focus on fp tractability relative to parameters, and must leave discussions of the effects of particular parameter values on runtimes to subsequent cognitive-algorithm-focused research.

The above also allows us to distinguish between complete and incomplete explanations of the tractability of a function ψ such as a computational-level model of eRCT. This is a consequence of the following.

Observation 2. *There exist functions ψ, and parameter sets K', K, such that $K' \subset K$ and ψ is fp-tractable for K but not for K'.*

This observation implies that there are input constraints (e.g., all parameters in K' are small) that by themselves cannot explain the tractability of a cognitive process modelled by a function ψ but are yet part of such an explanation. In such cases we would say the assumption that all parameters K' are small for the relevant situation is an incomplete explanation of the tractability of the process, whereas replacing K' by K yields a complete explanation. Hence, a conjectured explanation of the tractability of human problem solving under eRCT that invokes a set of constraints K is complete if and only if eRCT is fp-tractable relative to the parameters underlying K.[2]

3. FORMALIZING EXTENDED REPRESENTATION CHANGE THEORY

3.1. EXTENDED REPRESENTATION CHANGE THEORY

The eRCT of Öllinger et al. (2014) is based on the representation change theory (RCT) of Knoblich et al. (1999). Thus, in this subsection, we shall first explain RCT and then describe how eRCT extends RCT.

RCT builds on and extends the problem-restructuring account of insight-based problem solving proposed by Ohlsson (1992). In RCT, a problem representation consists of a structure encoding a problem state, a set of search operators that can transform the problem state, and a set of one or more constraints that encode both restrictions on the search process and the characteristics of those problem states that are goal states.[3] The entities comprising a problem state are grouped into chunks, where each chunk corresponds to a pattern that has proven useful in previous instances of problem solving. Chunks may be nested or intersect, but at any given time, only one set of chunks (whose members are not nested) is considered active. Search operators are restricted to manipulating whole active chunks. RCT envisions two representation restructuring operations, namely the removal of a particular constraint on the search process or the form of a goal state (constraint relaxation) or the replacement of an active chunk by its immediately-nested chunks (chunk decomposition).

A classic experimental paradigm for testing RCT is matchstick arithmetic problems. Here, the problem representation consists of a structure which represents the arrangement of matchsticks at three levels (individual numbers, functional terms, and formulas) such that there are chunks for individual matchsticks and the groupings of matchsticks into Roman numeral components (I, V, X), Roman numerals proper (e.g., III, VI, IX), and mathematical operators (+, −, =). In addition to the constraints defining valid formulas are constraints derived from common mathematical knowledge that restrict the matchstick-movement search process (Knoblich et al., 1999, p. 1537):

1. **Value constraint (VC):** A numerical value cannot be changed except through operations that produce compensating changes in other values, as when the same quantity is added to or subtracted from both sides of an equation.
2. **Operator constraint (OC):** An arithmetic function (e.g., addition or subtraction) cannot be arbitrarily deleted, introduced, or altered, except through operations that make corresponding changes elsewhere in the equation. The same is true of the equals sign.
3. **Tautology constraint (TC):** Arithmetic statements are supposed to have the general form

$$\mathrm{X} = f(Y, Z)$$

where f is addition, subtraction, or some other arithmetic function, because their purpose is to specify a calculation to be performed. Tautological statements of the general form $X = X$ are meaningless. (They have their uses in more advanced mathematics, e.g., as starting points for proofs, but not in elementary arithmetic.)

Each search operator can move a matchstick-chunk from one point in the problem state to another, possibly changing the orientation of that matchstick in the process. In order to invoke such an operator, the appropriate constraints must be relaxed and the appropriate chunks must be decomposed. The various search and restructuring operators that must be applied to give solutions to the example problems given in Figure 1 are shown in Figure 2.

As described above, RCT focuses primarily on the representation change associated with insight. The new aspect added in eRCT is an integration of classical Newell–Simon problem space search, such that "…problem solving is conceptualized as a dynamic search process that might include recursive steps, that is, repeated instances of search, impasse, and representational change" (Danek et al., 2016, p. 2). Note that in eRCT, as in RCT and its underlying ideas in Ohlsson (1992), insights may be false—it is only required that insights make further search possible, not that such search will lead to a solution.

3.2. A COMPUTATIONAL-LEVEL MODEL OF eRCT

A computational-level model of problem solving under eRCT is based on and must therefore provide formal descriptions of the following entities and mechanisms:

1. Problem representations (which encode problem states).
2. Search operators (which alter problem states).

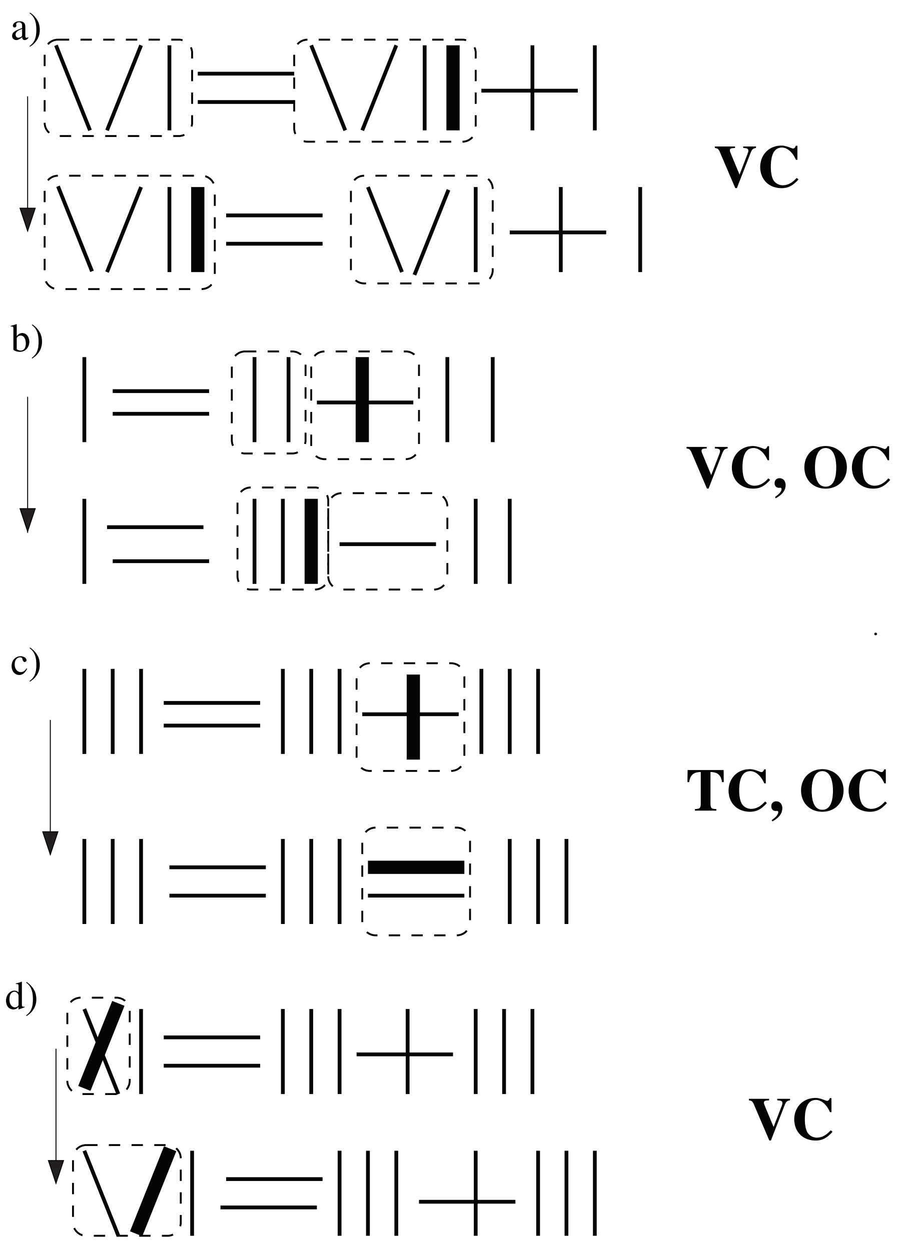

Figure 2.
Solving matchstick arithmetic problems within RCT. Parts (a–d) show how the matchstick arithmetic problems in Figure 1 are solved under RCT. Chunks involved in the solution process are indicated by dashed boxes. To the right of each problem are the constraints that were relaxed (VC: value constraint; OC: operator constraint; TC: tautology constraint).

3. Constraints (which encode both the forms of goal states and restrictions on the applications of search operators).
4. Restructuring operators (which alter problem representations).

In this subsection, we will give an informal description of a plausible (in terms of a basic) formalization of these entities and mechanisms as well as the input–output mapping based on these entities and mechanisms that encodes the process of problem solving under eRCT. Full details of this formalization are given in Section A.1 in Appendix A.

Within eRCT, a problem representation consists of a collection of entities and their relationships, a collection of chunks imposed on this collection, i.e., a chunk-structure, and a subset of those chunks comprising the currently active chunks. A basic model of entities and their relationships is a predicate-structure, a popular type of representation in cognitive science and artificial intelligence. A predicate-structure is composed of objects, e.g., sun, planet, and predicates relating those objects (as well as other predicates), e.g., Attracts (sun, planet), Cause(Gravity(Mass(sun)), Attracts(sun, planet)). Predicate-structures are naturally represented as vertex-labelled directed acyclic graphs in which objects are leaves, predicates are internal vertices, and each predicate is linked to its arguments by arcs that are directed from the predicate to those arguments (see Figure 3(a)).

Modeling chunk-structures is more difficult, as there are no models of chunk-structures in the cognitive science literature (Knoblich et al., 1999, p. 1536). A satisfactory model must have the following properties which are commonly imputed to chunks and chunk-structures (Simon, 1974; see also Ericsson and Kintsch, 1995):

- A chunk is a portion of the problem state which corresponds to a unit recognized as useful in previous problem solving experience.

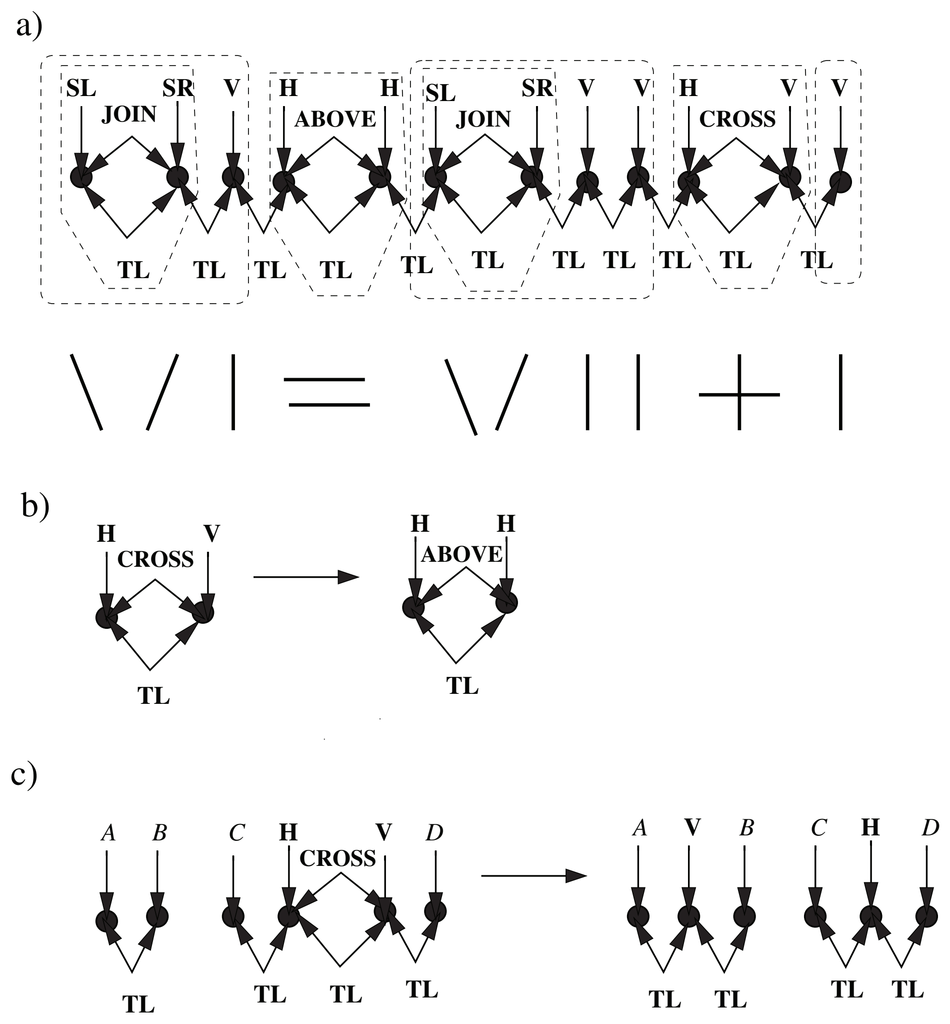

Figure 3.
Representing matchstick arithmetic problems within our formalization of eRCT. (a) A predicate-structure problem representation for the matchstick arithmetic problem in Figure 1(a). (b) A search operator that converts a plus sign to an equals sign. (c) A search operator that converts a plus sign to a minus sign and moves the vertical matchstick in the former plus sign to be part of a full numeral. Objects in the predicate-structures are represented by filled black circles, and each object corresponds to a matchstick. Abbreviated matchstick orientation and adjacency predicate types are HORIZONTAL (H), VERTICAL (V), SLANT-LEFT (SL), SLANT-RIGHT (SR), and TO-LEFT-OF (TL). Chunks are represented by dashed boxes and denote joint numerals (V, X), full numerals (III, VII), and operation signs (+, −, =); there is also a chunk (not shown) around each individual matchstick object.

- Chunks can be of arbitrary size.
- Chunks may overlap or be nested.

At any given time, a problem state is decomposed into and covered by a collection of possibly overlapping but non-nested chunks corresponding to the currently active chunks; moreover, the number of chunks in such a chunk-structure is relatively small, e.g., 7 ± 2.

We will model chunks as sub-predicate-structures, i.e., a subset of the objects in a predicate-structure and all relationships in that structure that are based on the objects in this subset. Such sub-predicate-structures can have arbitrary size, overlap, or be nested inside each other.

Moreover, this gives a natural specification of a chunk-structure as any non-nested collection of chunks that covers all objects (though not necessarily all predicates) in a predicate-structure. An example of such a problem representation for the matchstick arithmetic problem in Figure 1(d) is given in Figure 3(a). Note that there may be more than one instance of the same type of chunk in a problem representation, e.g., the individual matchstick chunks in part Figure 3(a); hence, we will distinguish between the set T of chunk types available for making chunk-structures and the individual chunk-instances making up a chunk-structure D relative to a particular problem representation.

We will model search operators as substructure replacement rules of the form $X \rightarrow Y$ that operate on predicate-structures. An application of such a rule to a predicate-structure p replaces one occurrence of predicate-substructure X in p with predicate-structure Y. As search operators can only manipulate active chunks in the chunk-structure of a predicate-structure, X and Y are phrased in terms of and hence can only move chunks in the chunk-structure of X or add, delete, or modify predicates in the structure linking chunks in, but not contained in, the chunk-structure of X. Example search operators that convert a plus sign into an equals sign and move a vertical matchstick to change a plus sign to a minus sign are given in Figure 3(b,c).

We will model constraints as formulas in a suitably rich form of logic which operate over the objects, predicates, and chunks in a problem representation. We will also recognize these constraints as being in two sets C_G and C_O, with C_G specifying the form of goal states and C_O specifying restrictions in the application of search operators to a problem state. Both of these sets of constraints are effectively AND-ed internally, such that a problem state is a solution if and only if it satisfies all constraints in C_G and the application of a search operator is valid if and only if this application does not violate any constraint, i.e., satisfies all constraints, in C_O.

Finally, we will model the constraint relaxation operator as the deletion of one constraint in $C = C_G \cup C_O$ and the chunk decomposition operator as the replacement of a chunk x in the chunk-structure by one or more non-overlapping chunks that are nested inside and collectively cover all objects in x. The sequence of chunk decomposition and search operators required to solve the matchstick arithmetic problem in Figure 1(b) is shown in Figure 4.

Given the above, we can now specify the input–output mapping that corresponds to the process of problem solving under eRCT. In human problem solving, this process can involve multiple impasses, which in turn require multiple rounds of restructuring followed by further search. This is specified as follows:

Problem Solving under eRCT (Human-behavior Version)

Input: Chunk-type set T, search-operator set O, problem representation p with chunk-structure D, constraint-set $C = (C_G, C_O)$, and integers k_C, k_D, and k_S.
Output: A sequence s consisting of the application of $\leq k_C$ constraint relaxation and $\leq k_D$ chunk decomposition operators and $\leq k_S$ search operators from O in any order that transforms p into a goal state consistent with C_G, if such an s exists, and special symbol $\perp$ otherwise.

Note that we explicitly represent k_C, k_D, and k_S in order to allow complexity analyses of this input–output mapping to assess the effects of these quantities on the runtime of any algorithm that computes this mapping. As we are interested here in ideal observer models which specify the best possible solution behaviors under eRCT, we can eliminate all search operators before the final sequence of restructuring operators (as these served only to illustrate that preceding episodes of restructurings were insufficient to allow a solution).[4] This effectively results in a solution-producing operator sequence in which there is at most one initial sequence of restructuring operator applications followed by a sequence of search operator applications. This is specified as follows:

Problem Solving under eRCT (Ideal-behavior Version)

Input: Chunk-type set T, search-operator set O, problem representation p with chunk-structure D, constraint-set $C = (C_G, C_O)$, and integers k_C, k_D, and k_S.
Output: A sequence s consisting of the application of $\leq k_C$ constraint relaxation and $\leq k_D$ chunk decomposition operators followed by the application of $\leq k_S$ search operators from O that transforms p into a goal state consistent with C_G, if such an s exists, and special symbol $\perp$ otherwise.

It is this second version that we will analyze in the remainder of the paper.

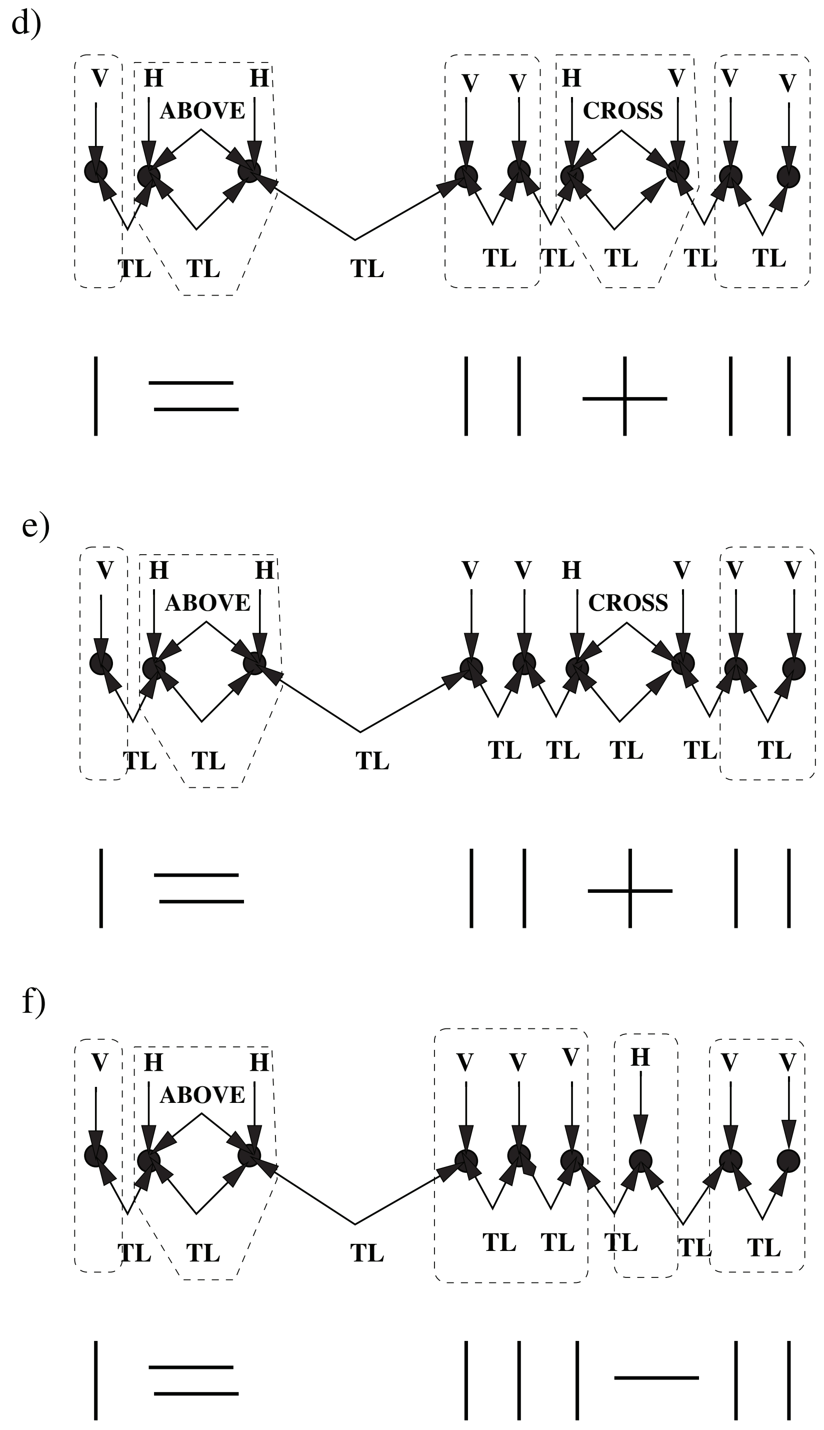

Figure 4.
Representing matchstick arithmetic problems within our formalization of eRCT (continued). This figure shows the sequence of chunk decomposition and search operators needed to solve the matchstick arithmetic problem in Figure 1(b). (d) The initial problem representation. (e) A chunk decomposed version of (d). (f) The problem representation in (e) after application of the matchstick-move search operator in Figure 3(c). Note that the representation in (f) has been "re-chunked" to show the newly created number III and minus operation sign.

4. COMPLEXITY RESULTS

We start by stating a general intractability result for the computational-level model Problem Solving under eRCT as defined in Section 3.2:

Result 1. Problem Solving under eRCT *is intractable even when $k_C = k_D = 0$ in the sense that all algorithms computing this input–output function require super-polynomial time.*[5]

This result establishes that there are no polynomial-time algorithms (and hence efficient ideal observer models) that can correctly compute this input–output mapping for all inputs, regardless of whether or not insight is involved, i.e., whether $k_C + k_D > 0$ or $k_C = k_D = 0$ respectively. Hence, restrictions must be assumed to apply to the input domain of Problem Solving under eRCT for the theory to be able to explain efficient solution of problems (whether or not this solving involves insight) by human beings.

In our parameterized complexity analyses, we consider the following three classes of restrictions (see Table 1):

1. Restrictions on internal problem representation ($|p|$, $|D|$, $|C|$, $|O|$).
2. Restrictions on the problem solution process (k_C, k_D, k_S).
3. Restrictions on interactions between the internal problem representation and the problem solution process ($|O_A|$, $|D_A|$).

Relative to these parameters we obtained the following fp-intractability results.[6]

Result 2. Problem Solving under eRCT *is fp-intractable for parameter set* $\{|C|, |O|, k_C, k_D, k_S\}$ even when $k_C = k_D = 0$.

Table 1.
Parameters considered in our analysis of Problem Solving under eRCT.

Name	Definition
$\|p\|$	Total number of objects and predicates in p
$\|D\|$	Number of chunks in D
$\|C\|$	Total number of constraints in C_G and C_O
$\|O\|$	Number of available search operators
$\|kc\|$	Maximum number of constraint relaxation restructurings
$\|ko\|$	Maximum number of chunk decomposition restructurings
$\|ks\|$	Maximum number of search operator applications
$\|O_A\|$	Maximum number of search operator application opportunities
$\|D_A\|$	Maximum number of active chunks in a chunk-structure

Result 3. Problem Solving under eRCT *is fp-intractable for parameter set* $\{|p|, |D|, |C|, k_C, k_D, k_S, |D_A|\}$ even when $k_C = k_D = 0$.

Result 4. Problem Solving under eRCT *is fp-intractable for parameter set* $\{|C|, |O|, k_C, k_D, k_S, |O_A|\}$ even when $k_C = 0$.

These results show that problem solving under eRCT cannot be done both efficiently and correctly under a number of restrictions. These results are much more powerful than they first appear, as it is known that an input–output mapping that is fp-intractable for a particular parameter set K is also fp-intractable relative to any subset of K (see Lemma 7 in Section A.2 in Appendix A). Hence, *none* of the parameters considered here can be either individually or in many combinations be restricted to yield efficient solvability of problems under eRCT.

Despite this, there are restrictions that do make problem solving under eRCT tractable.

Result 5. Problem Solving under eRCT is fp-tractable for parameter set $\{|p|, |D|, |C|, |O|, k_S\}$.

Result 6. Problem Solving under eRCT is fp-tractable for parameter set $\{|C|, k_D, k_S, |O_A|, |D_A|\}$.

Result 7. Problem Solving under eRCT is fp-tractable for parameter set $\{|C|, |O|, k_D, k_S, |D_A|\}$.

Again, these results are much more powerful than they first appear, as it is known that an input–output mapping that is fp-tractable for a particular parameter set K is also fp-tractable relative to any superset of K (see Lemma 6 in Section A.2 in Appendix A). Hence, any set of parameters including all of the parameters in any of the parameter sets in Results 5–7 above can be restricted to yield efficient solvability of problems under eRCT.

5. DISCUSSION

The general intractability of eRCT even when no insights are invoked, i.e., when $k_C = k_D = 0$ (Result 1), implies that the computational difficulty of eRCT is due to the underlying computational complexity inherent in problem space search. This has long been informally assumed in cognitive science (under the term "combinatorial problem space explosion") and was first shown formally in Sajedinia and Wareham (2014) (courtesy of the intractability of STRIPS, a planning system popular in AI that is based on Newell and Simon's General Problem Solver (Bylander, 1994)). However, it is nonetheless sobering that the simpler case of problem space search embodied in our computational-level model of eRCT is also intractable. This also establishes formally that, contrary to the conjectures of some (Öllinger, Jones, & Knoblich, 2006; Öllinger & Knoblich, 2009), restructuring is not central to the intractability of eRCT.

What does, then, allow eRCT to be tractable and hence explain observed instances of human problem solving with insight? Our fp-intractability results allow a more detailed refutation of the conjectured centrality of restructuring in the computational difficulty of eRCT. In particular Result 3 establishes that restricting the possibilities for constraint relaxation and chunk decomposition (by simultaneously restricting $\{k_C, |C|\}$ or $\{k_D, |D|\}$, respectively) cannot help, even if all four of these parameters are restricted simultaneously. Indeed, as noted in Section 4, our fp-intractability results (Results 2–4) imply that *none* of the parameters considered here (see Table 1) can be either individually or in many combinations be restricted to yield tractability. This renders incomplete additional explanations suggested by the results of experiments on human problem solving, e.g., that problem solving requiring insight becomes easier when either the number of constraint relaxations (k_C) (Knoblich et al., 1999, p. 1535) or the total number of constraints (C) (MacGregor & Cunningham, 2009, p. 133) are restricted.

Complete explanations are nonetheless close at hand. For example, though restricting $\{|D|, |C|, k_C, k_D\}$ does not help, fp tractability and thus a provably complete explanation can be achieved if one in addition limits the opportunities for problem space search by also restricting $\{|p|, |O|, k_S\}$ (Result 5).[7] Once one such complete explanation is in hand, others can readily be derived by examining the mechanisms in the underlying fp algorithm and considering new parameters associated with those mechanisms. For example, the loose upper bounds in the derivation of the runtime of the algorithm underlying Result 5 suggest that restrictions on $|p|$ and $|D|$ may be too broad. Results 6 and 7 thus invoke alternate sets of parameters involving parameters $|D_A|$ and $|O_A|$ (which were not considered in the initial stages of the research reported here) that can be restricted to limit the degree of chunk decomposition and opportunities for problem space search in a tighter fashion. As only active chunks can be decomposed, $|D|$ can be replaced with $|D_A|$ (which is potentially much smaller, e.g., 7 ± 2 (Simon, 1974)); similarly, loose algorithm runtime bounds based on $\{|p|, |O|\}$ can be replaced by tighter bounds based on $|O_A|$ (which is again potentially much smaller). Hence, as noted in Section 2, the parameter sets in Results 5–7 describe distinct but related classes of cognitive mechanisms that can potentially explain efficient problem solving under eRCT.

It must be stressed that such theoretical analyses are but part of the research process. Competing complete explanations should be evaluated against human behavior to see not only if the values of parameters in these explanations are in fact small in problems solved by human beings but also to determine if these parameters interact in the manner proposed by these explanations. This will of course involve the careful interpretation of the results of performed human experiments as well as the design and performance of new experiments, all of which will inform further theoretical analysis. The required integration of theoretical analyses and experiments in such a research cycle is difficult, and a very important direction for future research.

It is tempting to think that the progressively more fine-grained theoretical analyses illustrated above, and in particular the introduction of new and more refined parameters, can be carried out purely in the realm of algorithms. However, such new parameters must always first be evaluated for fp-intractability to assess the extent to which they can be exploited in fp algorithms and hence participate in complete explanations. For example, let $|O_I|$ be the maximum size (in terms of total number of objects and predicates) of any X or Y in any operator $X \rightarrow Y$ in O and $|C_L|$ be the maximum number of symbols of any constraint in C (where predicates count as single symbols). Both are plausible restrictions, as large search operators and/or constraints could conceivably be responsible for computational difficulty. However, it is easy to show that *all* of our intractability results hold under the following conditions:

- $|O_I| \leq 3$ (with each operator being restricted to modifying the type of a single predicate);
- $|C_L| = 1$; and
- $|C| = |C_L| = 1$.

The last two of these are achieved by admittedly gross abuses of formalism (namely, collapsing the computations in a constraint and a set of constraints, respectively, into the internal computation of a single predicate). Such may not be psychologically allowable and should be forbidden in our computational-level model. However, note that such absurdities (and the complex interactions between model aspects underlying these absurdities) only become visible with access to detailed algorithmic models, which are best derived and examined using complexity-based analyses like the ones applied in this paper. This amply demonstrates the utility and necessity of these analyses in deriving complete explanations of the tractability of computational-level models.

A final and important caveat is appropriate at this time: our analyses are geared towards producing ideal observer models and associated complete explanations whose operation is strictly deterministic. This does not allow us to fully address issues associated with probabilistic and/or non-optimal processing in problem solving, such as is introduced by problem space search heuristics like means–ends analysis or hill climbing (Ohlsson, 2011; Öllinger & Knoblich, 2009). However, this does not mean that our results have nothing to say about such matters. For example, the following is a consequence of Result 1.

Result 8. Problem Solving under eRCT *is not solvable by a polynomial-time algorithm that operates correctly with probability* $\geq 2/3$.[8]

Though perhaps too broad in scope, this is to our knowledge the first result to put formally provable limits on the runtimes and achievable success rates of heuristic-based problem solving, regardless of whether or not insight is involved. It is our hope that, with the guidance of computationally minded researchers in the problem solving community, ongoing work on the computational complexity analysis of probabilistic cognitive models (e.g., Blokpoel, Kwisthout, van der Weide, Wareham, & van Rooij, 2013) and probabilistic parameterized complexity (e.g., Kwisthout, 2015; Montoya & Muller, 2013) can be extended to provide additional results that are of more direct relevance.

6. CONCLUSIONS

In this paper, we have illustrated how computational complexity analysis can used to derive ideal observer models for theories of problem solving as well as provably complete explanations for situations in which such models are computationally tractable. In particular, we have (1) shown that several proposed explanations of what makes problem solving possible under eRCT are incomplete and (2) derived several new explanations that are complete. It is our hope that the techniques shown here will aid problem solving researchers in planning and interpreting the results of their experiments as well as formulating new and more accurate computationally based models of problem solving.

NOTES

[1] This paper is a revision and expansion of results originally presented without proof by the author in a 2011 talk given at Dagstuhl Workshop 11351 (Resource-bounded Problem Solving) and a poster abstract presented at ICCM 2012 (Wareham, 2012).

[2] An alternative statement of completeness of an explanation relative to a set of constraints K as defined here is that a complete explanation E of the tractability of some cognitive ability A relative to K implies that (1) an algorithm for A whose tractability is explained by restrictions of K is possible and (2) this algorithm potentially explains how people do A. Conversely, if an explanation E is incomplete then no such algorithm is possible and E cannot explain how people do A.

In this light, our computational complexity analysis approach can be seen as being analogous to Newell and Simon's "sufficiency proofs" approach, in which an explanation E of A is implemented as a program (Newell & Simon, 1972). If that program runs in a manner consistent with human performance, then E is a potential explanation of A; otherwise, it cannot explain how people do A.

[3] Constraint mechanisms have also been invoked to explain other aspects of problem solving, e.g., how people learn from errors during skill practice (Ohlsson, 1996).

[4] It could be argued that the eliminated search operators are necessary because they establish the need for (and may even in some way affect the choice of) the overall set of restructurings needed to achieve solution. However, until some formalizable manner is proposed by which the initial search operators so influence the overall set of restructurings, the best possible sequences of search and restructuring operators to achieve solution in an ideal observer model of eRCT are of the form described here.

[5] This claim follows from our NP-hardness proof for Problem Solving under eRCT (see Section A.3 in Appendix A) and the conjecture, widely believed within computer science (Fortnow, 2009; Garey & Johnson, 1979) and cognitive science (van Rooij, 2008), that $P \neq NP$.

[6] These claims follow from our $W[1]$-hardness proofs for Problem Solving under eRCT (see Section A.3 in Appendix A) and the conjecture, widely believed within computer science (Downey & Fellows, 1999) and cognitive science (van Rooij, 2008), that $FPT \neq W[1]$.

[7] The seven-parameter set in the text and the five-parameter set in Result 5 are equivalent, as $k_C \leq |C|$ and $k_D \leq |D|$ and any restrictions on $|C|$ and $|D|$ thus also imply that k_C and k_D are restricted.

[8] This claim follows the conjecture $P \neq NP$, which is widely believed within computer science (Downey & Fellows, 1999) and cognitive science (van Rooij, 2008), and the conjecture $P = BPP$, which is widely believed within computer science (Wigderson, 2007, Section 5.2).

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.

Ash, I. K., & Wiley, J. (2006). The nature of restructuring in insight: An individual-differences approach. *Psychonomie Bulletin & Review*, *13*(1), 66–73.

Batchelder, W. H., & Alexander, G. E. (2011). Insight problem solving: A critical examination of the possibility of formal theory. *Journal of Problem Solving*, *3*(2), 51–100.

Blokpoel, M., Kwisthout, J., van der Weide, T. P., Wareham, T., & van Rooij, I. (2013). A computational-level explanation of the speed of goal inference. *Journal of Mathematical Psychology*, *57*(3), 117–133.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, *89*(1–2), 165–204.

Chu, Y., & MacGregor, J. N. (2011). Human performance on insight problem solving: A review. *Journal of Problem Solving*, *3*(2), 119–150.

Danek, A. H., Wiley, J., & Öllinger, M. (2016). Solving classical insight problems without Aha! experience: 9 dot, 8 coin, and matchstick arithmetic problems. *Journal of Problem Solving*, *9*(1).

Downey, R., & Fellows, M. (1999). *Parameterized complexity.* Berlin: Springer.

Ericsson, K. A., & Kintsch, W. (1995). Long-term working memory. *Psychological Review*, *102*(2), 211.

Fortnow, L. (2009). The status of the P versus NP problem. *Communications of the ACM*, *52*(9), 78–86.

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

Kaplan, C. A., & Simon, H. A. (1990). In search of insight. *Cognitive Psychology*, *22*, 374–419.

Kershaw, T. C., & Ohlsson, S. (2004). Multiple causes of difficulty in insight: The case of the nine-dot problem. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *30*(1), 3–13.

Knoblich, G., Ohlsson, S., Haider, H., & Rhenius, D. (1999). Constraint relaxation and chunk decomposition in insight problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *25* (6), 1534–1555.

Kwisthout, J. (2015). Tree-width and the computational complexity of MAP approximations in Bayesian networks. *Journal of Artificial Intelligence Research*, *53*, 699–720.

MacGregor, J. N., & Cunningham, J. B. (2009). The effects of number and level of restructuring in insight problem solving. *Journal of Problem Solving*, *3* (2), 130–141.

Marr, D. (1981). *Vision: A computational investigation into the human representation and processing visual information.* San Francisco, CA: W. H. Freeman.

Montoya, J. A., & Muller, M. (2013). Parameterized random complexity. *Theory of Computing Systems*, *52*(2), 221–270.

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

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). London: Harvester-Wheatsheaf.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, *103*(2), 241.

Ohlsson, S. (2011). The problems with problem solving: Reflections on the rise, current status, and possible future of a cognitive research paradigm. *Journal of Problem Solving*, *3*(2), 101–128.

Öllinger, M., Jones, G., & Knoblich, G. (2006). Heuristics and representational change in two-move matchstick arithmetic tasks. *Advances in Cognitive Psychology*, *2*(4), 239–253.

Öllinger, M., Jones, G., & Knoblich, G. (2014). The dynamics of search, impasse, and representational change provide a coherent explanation of difficulty in the nine-dot problem. *Psychological Research*, *78*(2), 266–275.

Öllinger, M., & Knoblich, G. (2009). Psychological research on insight problem solving. In H. Atmanspacher & H. Primas (Eds.), *Recasting reality: Wolgang Pauli's philosophical ideas and contemporary science* (pp. 275–300). Berlin: Springer.

Ormerod, T. C., MacGregor, J. N., & Chronicle, E. P. (2002). Dynamics and constraints in insight problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *28*(4), 791–799.

Sajedinia, Z., & Wareham, T. (2014). Assessing the computational adequacy of the General Problem Solver model (Poster Abstract). In *Proceedings of the 35th Annual Conference of the Cognitive Science Society* (p. 3403). Austin, TX: Cognitive Science Society.

Simon, H. A. (1974). How big is a chunk? *Science*, *183*(4124), 482–488.

van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science*, *32*, 939–984.

van Rooij, I., Evans, P., Muller, M., Gedge, J., & Wareham, T. (2008). Identifying sources of intractability in cognitive models: An illustration using analogical structure mapping. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 915–920). Austin, TX: Cognitive Science Society.

Wareham, T. (1999). *Systematic parameterized complexity analysis in computational phonology* (Ph.D. thesis). Department of Computer Science, University of Victoria.

Wareham, T. (2012). What can (and can't) make problem solving by insight possible? A complexity-theoretic investigation. In N. Rußwinkel, U. Drewitz, & H. van Rijn (Eds.), *Proceedings of ICCM 2012: 11th International Conference on Cognitive Modeling* (pp. 142–143). TU Berlin.

Wareham, T., Robere, R., & van Rooij, I. (2012). A Change for the vetter? Assessing the computational cost of re-representation. In N. Rußwinkel, U. Drewitz, & H. van Rijn (Eds.), *Proceedings of ICCM 2012: 11th International Conference on Cognitive Modeling* (pp. 111–116). TU Berlin.

Wigderson, A. (2007). P, NP and mathematics—A computational complexity perspective. In *Proceedings of ICM 2006: Volume i* (pp. 665–712). Zurich: EMS Publishing House.

APPENDIX A: PROOFS OF RESULTS

In order to prove tractability and intractability results for the model of problem solving given in Section 3, we must fully formalize the various entities mentioned in this model. Hence, in this appendix, we first give the formal details of our computational-level model of eRCT (Section A.1). This is followed by a brief summary of techniques for establishing polynomial-time and fixed-parameter intractability (Section A.2) and the proofs of the various intractability and tractability results presented in Section 4 (Section A.3).

A.1. FORMALIZATION OF REPRESENTATION CHANGE THEORY

In this section, we will give the details of our formalizations for the various entities described in our computational-level model of eRCT given in Section 3. Much of this formalization will build on the formalization of predicate-structures and analogical matching of these structures given in van Rooij, Evans, Muller, Gedge, and Wareham (2008).

A problem representation consists of a collection of entities and their relationships, a collection of chunks imposed on this collection, and a subset of those chunks comprising the currently active chunks. We model entity-relationship collections as predicate-structures. Following van Rooij et al. (2008), predicate-structures are formalized as a restricted type of vertex- and edge-labelled directed acyclic graph.

Definition 3. *A* concept graph *is a quadruple* $(G, \lambda_A, \lambda_B, \lambda_P)$ *for a directed acyclic graph* $G = (V, A)$ *and functions* λ_A, λ_B, *and* λ_P *called* labelings *such that:*

1. $\lambda_A : A \rightarrow N$.
2. λ_B *is 1:1 and onto and defined on the leaves of G.*
3. λ_P *is defined on the internal vertices of G.*
4. *If v is an internal vertex, then* λ_A *either enumerates the arcs leaving v or is constantly* 0 *on this set. In the first case, v is* **ordered**, *and in the second* **unordered**.
5. *For internal vertices u, v with* $\lambda_P(u) \neq \lambda_P(v)$, *the following hold:*
    a) *Either both u and v are ordered or u and v are unordered;*
    b) *u and v both have the same number of children in G; and*
    c) $\{(v', \lambda_A(v, v')) \mid (v, v') \in A\} \neq \{(u\ \lambda_A(u, u')) \mid (u, u') \in A\}$.

A concept graph is ordered if all of its internal vertices are ordered; otherwise, if it contains at least one unordered predicate, it is unordered. Note that in a concept graph, internal vertices correspond to predicates or functions, leaves correspond to objects, and the labelings λ_P, λ_B, and λ_A assign predicate-types, object-names, and predicate argument-order labels, respectively. A chunk-type corresponds to a predicate-structure, a chunk-instance is the portion of a predicate-structure exactly matched by a chunk-type under analogy-morphism as formalized in van Rooij et al. (2008), and a chunk-structure for a predicate-structure *p* is a set of possibly overlapping but non-nested chunk-instances that collectively cover all objects in *p*. Note that the chunk-instances in a chunk-structure of a predicate-structure need not include all predicates in that predicate-structure.

A search operator is a substructure replacement rule of the form $X \rightarrow Y$, where *X* and *Y* are predicate-structures with associated chunk-structures. An application of such an operator to a predicate-structure *p* involves the exact matching via analogy morphism of *X* and its associated chunk-structure to a sub-predicate-structure of *P* and the subsequent replacement of *X* with *Y*. Changes made by rules are restricted such that the transformation from *X* to *Y* can only move chunks in the chunk-structure of *X* or add, delete, or modify predicates in the structure linking chunks in, but not contained in, the chunk-structure of *X*. In the remainder of this appendix, we further restrict search operators such that they cannot move chunks and may only modify the types of predicates attached to but not included in chunks; as such, they correspond to the predicate-structure re-representation rules described in Wareham, Robere, and van Rooij (2012).

A constraint is a formula in first-order logic whose basic elements are Boolean predicates defined over the objects, predicates, and chunks in a problem representation. Each of these predicates is polynomial-time computable and returns *True* if a particular structure exists in a problem representation and *False* otherwise. In the remainder of this appendix, we restrict a constraint to consist of a single Boolean predicate.

Finally, the application of a constraint relaxation operator corresponds to the removal of one of the constraints in $C_G \cup C_O$ and the application of a chunk decomposition operator corresponds to the replacement of a chunk *x* in a chunk-structure *D* by a set of one or more possibly overlapping but non-nested chunks that cover at least those objects covered by *x* that are not covered by the other chunks in *D*.

A.2. PROVING INTRACTABILITY

Given some criterion of tractability like polynomial-time or fixed-parameter solvability, we can define the class *T* of all input–output mappings that are tractable relative to that criterion. For example, *T* could be the class *P* of decision problems (see below) solvable in polynomial-time, or *FPT*, the class of parameterized problems that are fp-tractable. We can show that a particular input–output mapping is not in *T* (and thus that this mapping is intractable) by showing that this mapping is at least as hard as the hardest input–output mapping in any mapping class C that properly includes (or is strongly conjectured to properly include) *T*. For example, *C* could be *NP*, the class of decision problems whose candidate solutions can be verified in polynomial time, or a class of parameterized problems in the *W*-hierarchy = {*W*[1],

$W[2], \ldots, W[P], \ldots, XP\}$ (see Garey and Johnson (1979) and Downey and Fellows (1999) for details).

We will focus here on reducibilities between pairs of decision problems, i.e., input–output mappings whose answers are either "Yes" or "No". The two types of reductions used in this paper are as follows.

Definition 4. *Given a pair* Π, Π' *of decision problems,* Π polynomial-time many-one reduces to Π *if there is a polynomial-time computable function* f *mapping instances* I *of* Π *to instances* $f(I)$ *of* Π' *such that the answer to* I *is "Yes" if and only if the answer to* $f(I)$ *is "Yes".*

Definition 5. *Given a pair* Π, Π' *of parameterized decision problems,* Π fp-reduces to Π *if there is a function* f *mapping instances* $I = (x,p)$ *of* Π *to instances* $I' = (x',p')$ *of* Π' *such that (i)* f *is computable in* $g(p)|x|^a$ *time for some function* $g(\ )$ *and constant a, (ii)* $p' = h(p)$ *for some function* $h(\ )$*, and (iii) the answer to* I *is "Yes" if and only if the answer to* $I' = f(I)$ *is "Yes".*

A reducibility is appropriate for a tractability class T if whenever Π reduces to Π' and $\Pi' \in T$ then $\Pi \in T$. We say that a problem Π is C-hard for a class C if every problem in C reduces to Π. A C-hard problem is essentially as hard as the hardest problem in C.

Reducibilities become particularly useful by following three easily provable properties:

1. If Π reduces to Π' and Π is C-hard then Π' is C-hard.
2. If Π is C-hard and $T \subset C$ then $\Pi \notin T$, i.e., Π is not tractable.
3. If Π is C-hard and $T \subseteq C$ then $\Pi \notin T$ unless $T = C$, i.e., Π is not tractable unless $T = C$.

These properties are easily provable for many commonly used reducibilities, including those given in Definitions 4 and 5 above. The first and third of these properties will be used to show intractability below relative to tractable T-classes P and FPT and enclosing but not provably properly enclosing C-classes NP, $W[1]$, and XP. Note that these intractability results hold relative to the conjectures $P \neq NP$ and $FPT \neq W[1]$ which, though not proved, have strong empirical support and are commonly accepted as true within the computer science community (see Fortnow (2009), Garey & Johnson (1979), and Downey & Fellows (1999) for details).

In the case of parameterized results, one result may actually imply many others by the following.

Lemma 6 *(Wareham, 1999, Lemma 2.1.30). If problem* Π *is fp-tractable relative to parameter set* K *then* Π *is fp-tractable for any parameter set* K' *such that* $K \subset K'$.

Lemma 7 *(Wareham, 1999, Lemma 2.1.31). If problem* Π *is fp-intractable relative to parameter set* K *then* Π *is fp-intractable for any parameter set* K' *such that* $K' \subset K$.

These lemmas follow easily from the definition of fp tractability and parameterized reducibility.

In the remainder of this appendix, as we will be using reducibilities which operate on decision problems, we will actually be showing results relative to the decision problem associated with the input–output mapping Problem Solving under eRCT (see Section A.3). This is acceptable for such suitably defined decision problems because if such a problem is intractable, then so is the associated input–output mapping (otherwise, any tractable algorithm computing the input–output mapping could be used to construct a tractable algorithm for solving the decision problem, which would contradict the intractability of the decision problem).

Note also that in subsequent sections, all intractability results state specifically whether they hold relative to concept graphs that are ordered or unordered. This is done for formal completeness. However, given the fact that any result relative to ordered concept graphs implies the same result for unordered graphs (namely, add an extra unordered predicate of type not previously used in the instance to any concept graph to make the instance unordered), the results cited in the main text do not mention concept graph order.

A.3. RESULTS

These results are derived relative to the following decision problem associated with Problem Solving under eRCT as defined in Section 3:

Problem Solving under eRCT

Input: Concept-graph p with chunk-structure D drawn from chunk-type set T, search-operator set O, constraint-set $C = (C_G, C_O)$, and integers k_C, k_D, and k_S. *Question:* Is there a sequence s consisting of the application of $\leq k_C$ constraint relaxation and $\leq k_D$ chunk decomposition operators followed by the application of $\leq k_S$ search operators from O that transforms p into a concept graph consistent with C_G?

Our intractability results will be derived relative to reductions from the following problem:

Clique

Input: A graph G and an integer k.
Question: Does G contain a clique with k vertices, i.e., is there a subgraph of G that is isomorphic to K_k, the complete graph on k vertices?

These reductions will use the following transformation to create concept graphs:

- Given an unordered graph $G = (V, E)$ and arbitrary orders $v_i, v_2, \ldots, v_{|V|}$ and $e_1, e_2, \ldots, e_{|E|}$ on the vertices and edges of G, let $C'(G)$ be the ordered concept graph consisting of $|V|$ objects with labels $l_1, l_2, \ldots, l_{|V|}$ and $|E|$ ordered binary predicates of type t in which each object corresponds to a unique vertex in G and each predicate $|O_A|$ corresponds to a unique edge $e = (v_i, v_j)$, $i < j$, in E such that the first and second arguments of $|O_A|$ are the objects corresponding to the endpoints v and v_j, respectively, of e in G. In addition, let there be $|V|$ single-argument predicate of type t' such that each object is the argument of one of these predicates.

The predicates of type t and t' will be referred to as edge- and vertex-predicates, respectively. Note that $C'(G)$ is an extension of the transformation $C(G)$ defined in the Supplementary Materials of van Rooij et al. (2008).

Lemma 8. Clique polynomial-time many-one reduces to ordered Problem Solving under eRCT such that in the constructed instance of ordered Problem Solving under eRCT, $k_C = k_D = 0$ and k_S, $|C|$, and $|O|$ are functions of k from the given instance of Clique.

Proof: Given an instance $I = \langle G = (V, E), k\rangle$ of Clique, construct the following instance $I' = \langle p, T, D, O, C = (C_G, C_O), k_c, k_D, k_S\rangle$ of ordered Problem Solving under eRCT:

- $p = C'(G)$;
- T is the set consisting of chunk-type c_1 which encloses individual objects;
- D is the chunk-structure consisting of all $|V|$ chunk-instances of type c_1 in $C'(G)$;
- $O = \{o_1, o_2,\ldots, o_k\}$ where O_i, $1 \leq i \leq k$, is the vertex-selection search operator that takes a c_1-enclosed object and its associated vertex-predicate of type t' and changes the type of that vertex-predicate to t_i;
- C_G is the set of $k + k(k - 1)/2$ predicates $\{select_1(p), select_2(p),\ldots, select_k(p), isedge_{(1,2)}(p), isedge_{(1,3)}(p),\ldots isedge(_{k-1,k})(p)\}$ where $select_i(p)$, $1 \leq i \leq k$, returns True if and only if there is an object with vertex-predicate of type t_i in p and $isedge_{(i,j)}(p)$, $1 \leq i \leq j < k$, returns True if and only if the objects with vertex-predicates of types t and t_j are arguments of an edge-predicate of type t;
- $C_O = \emptyset$;
- $k_C = k_D = 0$; and
- $k_S = k$.

Instance I' can be constructed in time polynomial in the size of I; moreover, in I', $k_C = k_D = 0$, $k_S = k$, $|C| = k + k(k - 1)/2$, and $|O| = k$.

To prove that this construction is a reduction, we must show that the answer to the given instance of Clique is "Yes" if and only if the answer to the constructed instance of Problem Solving under eRCT is "Yes". We will do this by proving both directions of this implication separately:

($\Rightarrow$) If the answer to the given instance of Clique is "Yes", there is a k-clique in G. Let p' be the concept graph created by applying k of the operators in O to objects in p corresponding to the vertices in the k-clique in G such that the resulting objects have vertex-predicates of type t_i that are in the same ascending order as the arbitrary order on V assumed in $C'(G)$. Observe that this p' and D satisfy all constraint-predicates in C_G, which means that the answer to the constructed instance of Problem Solving under eRCT is "Yes".

($\Leftarrow$) Conversely, if the answer to the constructed instance of Problem Solving under eRCT is "Yes", each of the $k = k_S$ search operators in O can be applied to p to create a p' and D such that all of the constraint-predicates in C_G are satisfied. By the specification of O and C_G, this implies that there is a k-clique in G, which means that the answer to the given instance of Clique is "Yes".

This completes the proof.

Lemma 9. Clique polynomial-time many-one reduces to ordered Problem Solving under eRCT such that in the constructed instance of ordered Problem Solving under eRCT, $k_C = 0$, $k_D = k$ and k_S, $|C|$, $|O|$, and $|O_A|$ are functions of k from the given instance of Clique.

Proof: Consider a modified version of the reduction in Lemma 8 in which $k_D = k$ and the chunk-structure D has an additional $|V|$ chunks of type c_2, one enclosing each vertex-object and its associated vertex-predicate. In this reduction, vertex-selection in p now requires both the decomposition of a chunk of type c_2 and the application of the appropriate vertex- selection search operator o_i. The proof of correctness of this reduction is analogous to that for Lemma 8. To complete the proof, note that as at most $k_D = k$ objects can be decomposed at any time and at most $|O| = k$ operators can be applied to each of these decomposed objects, $|O_A| \leq k \times k = k^2$.

Lemma 10. Clique polynomial-time many-one reduces to ordered Problem Solving under eRCT such that in the constructed instance of ordered Problem Solving under eRCT, $k_C = k_D = 0$ and $|p|$, $|D|$, $|C|$, k_S, and $|D_A|$ are functions of k from the given instance of Clique.

Proof: Given an instance $I = \langle G = (V, E), k\rangle$ of Clique, construct the following instance $I' = \langle p, T, D, O, C = (C_G, C_O), k_C, k_D, k_S\rangle$ of ordered Problem Solving under eRCT:

- $p = C'(K_k)$;
- T is the set consisting of chunk-type c_1 which encloses individual objects;
- D is the chunk-structure consisting of all $|V| = k$ chunk-instances of type c_1 in $C'(K_k)$;
- O consists of the following two sets of search operators:
  1. vertex-selection operators $\{vs_1, vs_2,\ldots vs_{|V|}\}$, where vs_i takes a c_1-enclosed object and its associated vertex-predicate of type t' and changes the type of that vertex-predicate to t_i; and
  2. edge-check operators $\{ec_1, ec_2,\ldots ec_{|E|}\}$, where ec_i takes an edge-predicate linking c_1-enclosed objects whose associated vertex-predicates have types corresponding to the endpoints of e_i in G and changes the type of that edge-predicate to t'';

- C_G is the set of $k(k-1)/2$ predicates $\{isValid_{(1,2)}(p), isValid_{(1,3)}(p), \ldots isValid_{(k-1,k)}(p)\}$ where $isValid_{(i,j)}$, $1 \leq i \leq j < k$, returns True if and only if the objects with labels l_i and l_j have an edge-predicate of type t'';
- $C_O = \emptyset$;
- $k_C = k_D = 0$; and
- $k_S = k + k(k-1)/2$.

Note that no edge-check search operator in O can be applied unless the vertex-predicates of the vertex-predicates of the objects corresponding to the endpoints of the edge encoded in that operator have first been modified by the appropriate vertex-selection search operators. Instance I' can be constructed in time polynomial in the size of I; moreover, in I', $k_C = k_D = 0$, $|p| = k_S = k + k(k-1)/2$, $|C| = k(k-1)/2$, and $|D| = |D_A| = k$.

To prove that this construction is a reduction, we must show that the answer to the given instance of Clique is "Yes" if and only if the answer to the constructed instance of Problem Solving under eRCT is "Yes". We will do this by proving both directions of this implication separately:

(⇒) If the answer to the given instance of Clique is "Yes", there is a k-clique in G with k vertices and $k(k-1)/2$ edges. Let p' be the concept graph created by applying first the k vertex-selection search operators in O corresponding to the vertices in the k-clique in G to the objects in p and then the $k(k-1)/2$ edge-check search operators in O corresponding to the edges in the k-clique in G to the edge-predicates in p. Observe that this p' and D satisfy all constraint-predicates in C_G, which means that the answer to the constructed instance of Problem Solving under eRCT is "Yes".

(⇐) Conversely, if the answer to the constructed instance of Problem Solving under eRCT is "Yes", there was a sequence of $k + k(k-1)/2$ search operators from O that could be applied to create a p such that all of the constraint-predicates in C_G are satisfied. By the specification of O and C_G, this implies that there are k vertices and $k(k-1)/2$ edges that form a k-clique in G, which means that the answer to the given instance of Clique is "Yes".

This completes the proof.

Lemma 11. Ordered Problem Solving under eRCT is *NP*-hard when $k_C = k_D = 0$.

Proof: Follows from the *NP*-hardness of Clique (Garey & Johnson, 1979, Problem GT19) and the reduction in Lemma 8.

Result 1. Problem Solving under eRCT *is polynomial-time intractable even when* $k_C = k_D = 0$.

Proof: Follows from Lemma 11.

Result 2. Problem Solving under eRCT *is fp-intractable for parameter set* $\{|C|, |O|, k_C, k_D, k_S\}$ *even when* $k_C = k_D = 0$.

Proof: Follows from the $W[1]$-hardness of Clique for parameter set $\{k\}$ (Downey & Fellows, 1999) and the reduction in Lemma 8.

Result 3. Problem Solving under eRCT *is fp-intractable for parameter set* $\{|p|, |D|, |C|, k_C, k_D, k_S, |D_A|\}$ *even when* $k_C = k_D = 0$.

Proof: Follows from the $W[1]$-hardness of Clique for parameter set $\{k\}$ (Downey & Fellows, 1999) and the reduction in Lemma 10.

Result 4. Problem Solving under eRCT *is fp-intractable for parameter set* $\{|C|, |O|, k_C, k_D, k_S, |O_A|\}$ *even when* $k_C = 0$.

Proof: Follows from the $W[1]$-hardness of Clique for parameter set $\{k\}$ (Downey & Fellows, 1999) and the reduction in Lemma 9.

Result 5. Problem Solving under eRCT *is fp-tractable for parameter set* $\{|p|, |D|, |C|, |O|, k_S\}$.

Proof: Consider the following algorithm: there are at most $|C|^{kC} \leq |C|^{|C|}$ (as $k_C \leq |C|$) ways to remove at most k_C constraints from C to create C', and at most $|D|^{kD} \leq |D|^{|D|}$ (as $k_D \leq |D|$) possible chunk-structures that can be created by applying at most k_D chunk decomposition operations to D. Hence, there are at most $|C|^{|C|} \times |D|^{|D|}$ combinations of p and D' to which search-operators can be applied relative to any C'. As there are at most $2^{|p|}$ subsets of elements of p that can comprise the target of a search operator and at most $|O|$ operators that can apply to each target, there are at most $(2^{|p|}|O|)^{kS}$ ways of applying search operators to each such combination. As each opportunity can be recognized, each application to an opportunity can be done, and the constraint-set C_G evaluated in polynomial time, the exhaustive algorithm implicit in the above runs in time that is exponential purely in $|p|$, $|D|$, $|C|$, $|O|$, and k_S, completing the proof.

Result 6. Problem Solving under eRCT *is fp-tractable for parameter set* $\{k_D, k_S, |C|, |O_A|, |D_A|\}$.

Proof: Consider the following algorithm: There are at most $|C|^{kC} \leq |C|^{|C|}$ (as $k_C \leq |C|$) ways to remove at most k_C constraints from C to create C', and at most $|D_A|^{kD}$ possible chunk-structures that can be created by applying at most k_D chunk decomposition operations to D. Hence, there are at most $|C|^{|C|} \times |D_A|^{kD}$ combinations of p and D' to which search-operators can be applied relative to any C'. As there are at most $|O_A|$ opportunities for applying search operators to any such combination, there are at most $|O_A|^{kS}$ ways of applying search operators to each such combination. As each opportunity can be recognized, each application to an opportunity can be done, and the constraint-set C_G evaluated in polynomial time, the exhaustive algorithm implicit in the above runs in time that is exponential purely in k_C, k_D, k_S, $|C|$, $|D_A|$, and $|O_A|$, completing the proof.

Result 7. Problem Solving under eRCT *is fp-tractable for parameter set* $\{|C|, |O|, k_D, k_S, |D_A|, |O|\}$.

Proof: Observe that each operator in O applies to some set of subsets of the active chunks in p, and it is theoretically possible for all operators in O to apply to each such chunk-subset. As there are at most such chunk-subsets, $|O_A| < |O| \times |D_A|^{|DA|+1}$, and the result follows by the algorithm given in the proof for Result 6.

Result 8. If $P = BPP$ and Problem Solving under eRCT is polynomial-time tractable by an algorithm which operates correctly with probability $\geq 2/3$ then $P = NP$.

Proof: It is widely believed that $P = BPP$ (Wigderson, 2007, Section 5.2) where *BPP* is considered the most inclusive class of problems that can be efficiently solved using probabilistic methods (in particular, methods whose probability of correctness is $\geq 2/3$ and can be efficiently boosted to be arbitrarily close to probability one). Hence, if Problem Solving under eRCT has a polynomial-time algorithm which operates correctly with probability $\geq 2/3$ and hence is in $BPP = P$, then by Result 1 and the definition of *NP*-hardness we know that $P = NP$.