Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Mathematical Psychology 49 (2005) 160–187
Journal of Mathematical Psychology
www.elsevier.com/locate/jmp

Sources of complexity in subset choice

Iris van Rooij[a,*], Ulrike Stege[b], Helena Kadlec[c]

[a]Human-Technology Interaction, Eindhoven University of Technology, P.O. Box 513, 560 MB Eindhoven, The Netherlands
[b]Department of Computer Science, University of Victoria, Victoria, Canada
[c]Department of Psychology, University of Victoria, Victoria, Canada

Received 2 May 2003; received in revised form 14 April 2004

Abstract

Subset choice denotes the task of choosing a subset of items from among a set of available items. Because the number of possible choice options in subset choice grows exponentially with the size of the choice set, subset choice tasks can be computationally challenging. This paper discusses how the computational complexity of subset choice under different models can be utilized in the quest for descriptive models of subset choice. We consider several models of subset choice (including the additive model, the binary-interaction model and the h-ary interaction model) and show how the theory of computational complexity (including the theory of NP-completeness and fixed-parameter tractability) can be used to evaluate the psychological plausibility of such models under different assumptions of processing speed, parallelism and size of problem parameters.

Keywords: Subset choice; Computational complexity; Decision-making; NP-completeness; Intractability; Fixed-parameter tractable; Algorithms

1. Introduction

Subset choice denotes the task of choosing a subset of items from among a set of available items. Subset choice arises in many different settings, ranging from mundane choice situations (e.g., when selecting toppings on a pizza, buying groceries, or inviting friends to a party) to highly specialized choice situations (e.g., when selecting members on a committee, hiring applicants for a set of positions, or deciding on a set of drugs to prescribe to a patient; see also Farquhar & Rao, 1976; Falmagne & Regenwetter, 1996; Fishburn & LaValle, 1993, 1996; Regenwetter, Marley, & Joe, 1998, for more examples). Two common assumptions in the literature on subset choice are that (i) each item or subset has an associated value and (ii) the subset is chosen according to some value criterion. For example, a decision maker may associate a value with each item and choose all those items whose value is at or above some threshold (Falmagne & Regenwetter, 1996; Regenwetter et al., 1998). Alternatively, a decision maker may associate a value with each subset and choose a subset whose value is maximum (Fishburn & LaValle, 1996).

As noted by Fishburn and LaValle (1996), subset choice problems can be notoriously challenging, because the decision maker potentially faces the problem of combinatorial explosion. Consider, for example, the task of choosing k pizza toppings from a set of n available toppings. For $n = 20$ and k even as small as 5 the number of possible subsets of toppings exceeds 15,000. If one is free to choose a subset of any size, the number of feasible solutions even exceeds 1,000,000. The fact that a person cannot possibly consider each and every one of these subsets places a strong constraint on the type of subset choice tasks that a person can perform. To illustrate, let us consider the model that assumes that (i) each subset has an associated value and (ii) a subset with maximum value is chosen. Table 1 presents an example with available pizza toppings $V = \{pepperoni, salami, ham, mushroom, pineapple\}$. The table lists for each possible subset $A \subseteq V$ the value $u(A)$

*Corresponding author. Fax: +31 40 2449875.
E-mail address: I.v.rooij@tm.tue.nl (I. van Rooij).

0022-2496/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmp.2005.01.002

Table 1
Each possible subset, $A \subseteq V$, for the set of pizza toppings $V = \{pepperoni, salami, ham, mushroom, pineapple\}$ and its value, $u(A)$, for a hypothetical decision-maker

Subset A	$u(A)$	Subset A	$u(A)$
$\varnothing$	0	{*pepperoni*, *salami*, *ham*}	10
{*pepperoni*}	8	{*pepperoni*, *salami*, *mushroom*}	14
{*salami*}	9	{*pepperoni*, *salami*, *pineapple*}	12
{*ham*}	7	{*pepperoni*, *ham*, *mushroom*}	15
{*mushroom*}	4	{*pepperoni*, *ham*, *pineapple*}	18
{*pineapple*}	2	{*pepperoni*, *mushroom*, *pineapple*}	9
{*pepperoni*, *salami*}	10	{*salami*, *ham*, *mushroom*}	17
{*pepperoni*, *ham*}	11	{*salami*, *ham*, *pineapple*}	20
{*pepperoni*, *mushroom*}	12	{*salami*, *mushroom*, *pineapple*}	10
{*pepperoni*, *pineapple*}	10	{*ham*, *mushroom*, *pineapple*}	13
{*salami*, *ham*}	13	{*pepperoni*, *salami*, *ham*, *mushroom*}	14
{*salami*, *mushroom*}	13	{*pepperoni*, *salami*, *ham*, *pineapple*}	17
{*salami*, *pineapple*}	11	{*pepperoni*, *salami*, *mushroom*, *pineapple*}	11
{*ham*, *mushroom*}	11	{*pepperoni*, *ham*, *mushroom*, *pineapple*}	17
{*ham*, *pineapple*}	14	{*salami*, *ham*, *mushroom*, *pineapple*}	19
{*mushroom*, *pineapple*}	1	{*pepperoni*, *salami*, *ham*, *mushroom*, *pineapple*}	16

assigned to A, reflecting the preferences of a hypothetical decision maker. If the decision maker's subset choice task is defined by assumption (ii) above, then the subset {*salami*, *ham*, *pineapple*} is the chosen subset. More formally, we can describe this model of subset choice by the following input–output specification, called Generalized Subset Choice:

Generalized Subset Choice

Input: A set $V = \{x_1, x_2, \ldots, x_n\}$ of n available items and a value function $u : 2^V \to \mathbb{Z}$ assigning an integer[1] value to every subset of V. (Here 2^V denotes the power set of V.)
Output: A subset $A \subseteq V$ such that $u(A)$ is maximum (i.e., for all $B \subseteq V$, $u(B) \leqslant u(A)$.)

If the decision maker is given no further information as part of the input, and if we assume that the decision maker performs a subset choice task by following an algorithm[2] (which we assume throughout this paper), then performing the task of Generalized Subset Choice requires him/her to consider *all* possible subsets; there simply is no other way to guarantee the output of a maximum valued subset. From this we conclude that Generalized Subset Choice is a psychologically implausible model of subset choice for all but very small n.

The situation may be quite different, however, if the value function can be decomposed, say, into a function of the values of individual items. As an example of this case, consider Fishburn's additive model of subset choice (e.g., Fishburn, 1992; Fishburn & LaValle, 1993). This model assumes that the value of a subset is equal to the sum of the values of its individual elements; i.e. for every subset $A \subseteq V$ we have

$$u(A) = \sum_{x \in A} u(x). \tag{1}$$

For example, in Table 1 we have $u(\{ham, mushroom, pineapple\}) = 13 = 7+4+2 = u(\{ham\}) + u(\{mushroom\}) + u(\{pineapple\})$.

The additive model of subset choice is formally captured by the following input–output specification, called Additive Subset Choice:

Additive Subset Choice

Input: A set $V = \{x_1, x_2, \ldots, x_n\}$ of n available items. For every item $x \in V$ there is an associated integer value $u(x)$.
Output: A subset $A \subseteq V$ such that $u(A) = \sum_{x \in A} u(x)$ is maximum.

Performing the task of Additive Subset Choice clearly does not require one to consider all possible subsets. Instead, the task can be performed by simply considering each item one by one, including an item in A if its value is positive and rejecting it otherwise. Additive Subset Choice is in this sense a much easier task than Generalized Subset Choice. The algorithm for computing Additive Subset Choice appears below in pseudo-code:

Additive Subset Choice Algorithm

Input: A set V and an integer value $u(x)$ for all $x \in V$
Output: A subset $A \subseteq V$ such that $u(A) = \sum_{x \in A} u(x)$

[1] We assume that decision makers can evaluate subset values with only finite precision. For simplicity we work with integer values. All results in this article can be generalized to any values of fixed precision, simply by scaling by the precision factor.

[2] We assume that the decision-maker does not have "oracle powers" (Garey & Johnson, 1979; Tsotsos, 1990).

is maximum
1. Let $A = \varnothing$
2. **while** $V \neq \varnothing$ **do**
3. pick some $x \in V$
4. **if** $u(x) \geqslant 0$ **then**
5. $A := A \cup \{x\}$
6. **end if**
7. $V := V \backslash \{x\}$ [The symbol "\"denotes the *set difference* operation; i.e., for two sets V and W, $V \backslash W = \{x \in V : x \notin W\}$]
8. **end** while
9. **return** A

Note that we may substitute "**if** $u(x) \geqslant 0$ **then**" in line 4 by "**if** $u(x) \geqslant \lambda$ **then**," where λ represents a preset value threshold. In this more general form, the algorithm can be seen as computing a subset containing all items whose value is at or above some threshold—i.e., the subset choice task described by Falmagne and Regenwetter (1996) and Regenwetter et al. (1998). In the special case that $\lambda = 0$ the algorithm solves Additive Subset Choice.

While Generalized Subset Choice is psychologically implausible for complexity reasons, Additive Subset Choice is psychologically implausible for a quite different reason. Namely, in many subset choice situations of psychological interest Eq. (1) simply does not hold. Table 1 serves to illustrate this point. Here the decision maker prefers a pizza with pepperoni alone over a pizza with pineapple alone ($u(pepperoni) > u(pineapple)$), prefers a pizza with salami alone over a pizza with ham alone ($u(salami) > u(ham)$) and, at the same time, prefers a pizza with both ham and pineapple over a pizza with both pepperoni and salami ($u(\{ham, pineapple\}) > u(\{pepperoni, salami\})$). This preference ordering of subsets of pizza toppings is entirely conceivable from a psychological point of view but is at odds with the additive model. Namely, the additive model would prescribe that if $u(pepperoni) > u(pineapple)$ and $u(salami) > u(ham)$ then $u(\{pepperoni, salami\}) = u(pepperoni) + u(salami) > u(pineapple) + u(ham) = u(\{ham, pineapple\})$.

Violations of Eq. (1) may be due to value interdependencies between items in the choice set. For example, the decision maker may consider pepperoni and salami so similar in taste that the value of their combination is *smaller* than the sum of their respective values. Alternatively, ham and pineapple may enhance each other's flavor such that the value of their combination is *larger* than the sum of their respective values. To accommodate the possibility of such value interdependencies, Fishburn and LaValle (1996; see also Fishburn, 1972) proposed the binary-interaction model of subset choice. This model assumes that each pair of items in a choice set has an associated interaction value denoting the value loss or value gain associated with the combination of the two items. That is, for every $x, y \in V$ there is a binary interaction term $\Delta(x, y)$ such that

$$u(\{x, y\}) = u(x) + u(y) + \Delta(x, y). \tag{2}$$

Furthermore, the binary-interaction model assumes that the value of any given subset $A \subseteq V$ is given by

$$u(A) = \sum_{x \in A} u(x) + \sum_{(y,z) \in A^2} \Delta(y, z), \tag{3}$$

where A^2 denotes the set of all unordered pairs of items in A.

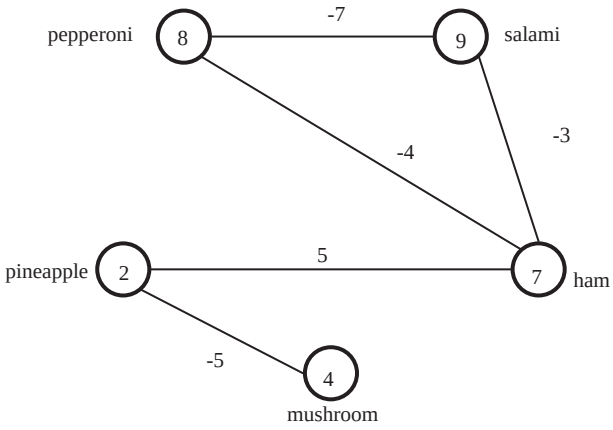

Fig. 1. The value-structure of the hypothetical decision-maker whose subset values appear in Table 1 represented by a weighted graph.

Fig. 1 illustrates how the binary-interaction model captures the subset values listed in Table 1. The figure presents a graph $G = (V, E)$ with vertex set $V = \{pepperoni, salami, ham, mushroom, pineapple\}$ and edge set $E = \{(pepperoni, salami), (salami, ham), (pepperoni, ham), (ham, pineapple), (mushroom, pineapple)\} \subseteq V^2$. Each vertex $x \in V$ has an associated value $u(x)$ and each edge $(y, z) \in E$ has an associated value $\Delta(y, z)$. Furthermore, for every $(y, z) \in V^2$, $(y, z) \in E$ if and only if $\Delta(y, z) \neq 0$. Let $E_G(A) = \{(x, y) \in E : x, y \in A\}$ be the edges in G that connect pairs of items in A. Then careful inspection of Table 1 and Fig. 1 reveals that $u(A) = \sum_{x \in A} u(x) + \sum_{(y,z) \in E_G(A)} \Delta(y, z)$ for each subset $A \subseteq V$. Hence, the values in Table 1 can be seen as arising from a value structure that satisfies the binary-interaction model.

In general we can represent the binary-interaction model using a weighted graph $G = (V, E)$ as done in Fig. 1. If the task is to choose a maximum valued subset, the binary-interaction model of subset choice is defined by the following input–output specification, called Graph Subset Choice:

Graph Subset Choice

Input: A graph $G = (V, E)$ with vertex set $V = \{x_1, x_2, \ldots, x_n\}$and edge set $E \subseteq V^2$. For every

$x \in V$ there is an associated integer value $u(x)$ and for every edge $(y,z) \in E$ there is an associated non-zero integer value $\Delta(y,z)$.

Output: A subset $A \subseteq V$ such that $u(A) = \sum_{x \in A} u(x) + \sum_{(y,z) \in \mathrm{E}_G(A)} \Delta(y,z)$ is maximum.

If we assume that the decision maker whose preference values appear in Table 1 knows that his/her preference values can be decomposed as done in Fig. 1, then the decision maker's task is better modeled by Graph Subset Choice than by Generalized Subset Choice. Notice that now it is no longer obvious that the only way to guarantee a maximum valued subset as output is to consider *all* possible subsets.

To illustrate, consider again Fig. 1. Note, for example, that $u(\mathit{mushroom}) + \Delta(\mathit{mushroom}, \mathit{pineapple}) < 0$ and $\Delta(\mathit{mushroom}, x) \leqslant 0$ for all $x \in V$. This means that for any two subsets A_1, $A_2 \subseteq V$ with $A_1 \subseteq A_2$, $\mathit{pineapple} \in A_1$, and $\mathit{mushroom} \in A_2$, we have $u(A_1) > u(A_2)$. In other words, the decision maker need not consider any subset that contains both *pineapple* and *mushroom*. This observation reduces the number of feasible subsets from 2^n to $2^n - 2^{n-2} = 3(2^{n-2})$. Also note that $u(\mathit{pineapple}) \geqslant 0$ and $\Delta(\mathit{pineapple}, x) \geqslant 0$ for all $x \in A$, with $x \neq \mathit{mushroom}$. This means that for any subset $A \subseteq V$, if $\mathit{mushroom} \notin A$, then $u(A \cup \{\mathit{pineapple}\}) \geqslant u(A)$. In other words, the decision maker can ignore all subsets that contain neither *pineapple* nor *mushroom*. This leaves him/her with $3(2^{n-2}) - 2^{n-2} = 2(2^{n-2}) = 2^{n-1}$ subsets to consider. Using more observations like these may shrink the search space for Graph Subset Choice even more. Note, however, that each of the illustrated reductions in search space was minor compared to the number of possible subsets; even after reduction the number of to be considered subsets is exponential in n. Can the search space be reduced such that it is bounded by a *polynomial* function of n? In the next section, we explain why this question is important for purposes of evaluating the psychological plausibility of Graph Subset Choice as a model of subset choice.

1.1. Tractability as a theoretical constraint on cognition

Philosophy of science distinguishes between empiricist and rationalist approaches to scientific understanding (Robinson, 1999). Much of the research in the psychological community on subset choice falls within the empiricist tradition, for example, by aiming at deriving empirically testable predictions from different subset choice models (e.g., Barberá & Pattanaik, 1986; Fishburn, 1972; Fishburn & LaValle, 1996; Regenwetter et al., 1998). In contrast, our approach is best seen as part of the rationalist tradition. Specifically, we aim at using mathematical theories of computational complexity to evaluate the a priori plausibility of subset choice models. To be clear, we do not mean to suggest that rationalist approaches should replace empiricist approaches. On the contrary, we see each approach as making valuable but incomplete contributions to psychology. A psychological model that is theoretically plausible need not be veridical, and this may be revealed by a test of empirical fit. Conversely, a model that fits empirical data sets may still fail as a psychological theory, for example, if the model implies properties of cognitive systems that are theoretically implausible or incoherent. In sum, we believe that both empirical and theoretical investigations can help us converge on veridical psychological theories.

Cognitive psychologists often overlook the potential contribution of systematic theoretical analysis (Kukla, 1989; Green, 1994). In fact, theoretical analysis can contribute in several different ways. For example, theoretical analyses can directly suggest cognitive structures and mechanisms (cf. *rational analysis* as pursued by, among others, Anderson (1990) and Oaksford and Chater (1998)). Also, mathematical theory plays a crucial role in the definition and assessment of "parsimony" of a model (see e.g. model selection approaches described by Grünwald (2000) and Pitt and Myung (2002)). Third, mathematical theory can contribute by specifying theoretical limits to serve as constraints on psychological models (e.g. Frixione, 2001; Oaksford & Chater, 1998; Parberry, 1994, 1997; Tsotsos, 1990; van Rooij, 2003). It is in the latter way that our assessment of the computational complexity of different subset choice models contributes to psychological science. We next explain our goal in more detail.

As mentioned before, Additive Subset Choice fails as a descriptive model whenever inter-item value dependencies play a role (Fishburn, 1972; Fishburn & LaValle, 1996). Hence, in these cases, more intricate models of subset valuation are needed. However, more intricate models of subset choice (e.g., Generalized Subset Choice and Graph Subset Choice) may require more computational power on the part of the decision maker. Since humans are limited in their computational (i.e., information processing) speed, only those subset choice models that stay within reasonable bounds of computational complexity should be deemed psychologically realistic. To capitalize on this theoretical constraint one needs working definitions of "computational complexity" and "reasonable bound." We adopt definitions that were developed in the mathematical theory of computational complexity (Downey & Fellows, 1999; Garey & Johnson, 1979).

Computational complexity theory, or complexity theory for short, provides a way of evaluating the amount of computational resources required for computation. Here we consider the resource *time*. Complexity theory defines the *time-complexity* of a task $\Pi : I \to R$, with input $i \in I$ and output $\Pi(i) \in R$, in terms of a function $O(f(n))$, where $n = |i|$ denotes the

Table 2
Polynomial, exponential and fpt running times as a function of the size of choice set, n, assuming 100 ms per basic operation (κ is fixed at 8)

n	$T(n)$	$T(n^2)$	$T(n^3)$	$T(2^n)$	$T(2^\kappa n)$	$T(2^\kappa + n)$	$T(n^\kappa)$
5	0.5 s	2.5 s	12.5 s	3.2 s	2.1 min	26 s	11 h
10	1.0 s	10 s	1.7 min	1.7 min	4.3 min	27 s	115 days
15	1.5 s	23 s	5.6 min	55 min	6.4 min	27 s	8.1 years
20	2.0 s	40 s	13 min	1,2 days	8,5 min	27 s	81 years
25	2.5 s	1 min	26 min	39 days	11 min	28 s	4.8×10^2 years
30	3.0 s	1.5 min	45 min	3.4 years	13 min	28 s	2.1×10^3 years
50	5.0 s	4.2 min	3.5 h	3.5×10^6 years	21 min	31 s	1.2×10^5 years
100	10 s	17 min	28 h	4.0×10^{21} years	43 min	35 s	3.2×10^8 years

size of the input. Here $O(f(n))$ (read as: *Big-Oh of* $f(n)$) expresses an asymptotic upper-bound on the number of basic operations required to compute $\Pi(i)$ for any i. A function $g(x)$ is said to be $O(f(x))$ if there are constants $c \geqslant 0$ and $x_o \geqslant 1$ such that $g(x) \leqslant cf(x)$, for all $x \geqslant x_o$. The definition of $O(.)$ can be straightforwardly extended to functions with two or more arguments. For example, a function $f(x, y)$ is $O(g(x, y))$ if there are constants $c \geqslant 0$ and x_o, $y_o \geqslant 1$ such that $f(x, y) \leqslant cg(x, y)$, for all $x \geqslant x_o$ and $y \geqslant y_o$. Note that the $O(.)$ notation serves to ignore constants and lower-order polynomials in the description of a function.[3] This loss of specificity has the added benefit that the classification of time-complexity becomes invariant (up to a polynomial amount of precision) over different encoding schemes and different models of computation (see e.g., Frixione, 2001; Garey & Johnson, 1979; Parberry, 1994, 1997; Tsotsos, 1990, 1991; van Rooij, 2003).

Traditionally, complexity theory considers a problem tractably computable only if it is of *polynomial* time-complexity, in other words, if the problem can be computed in time $O(n^\alpha)$, where α is a constant. A running time that is not bounded by some polynomial function of n, such as an exponential running time $O(\alpha^n)$, is then considered computationally *intractable*. To see that this classification has merit, let us assume that, say, 100 ms is a lower bound on the time it takes to perform a basic cognitive operation in subset choice (such as compare two values, retrieve the value of an item, sum two values). If we further assume that only one such basic operation can be performed at a time, then a human can perform a maximum of 10 basic cognitive operations per second. Table 2 illustrates how subset choice models of polynomial-time complexity and exponential-time complexity fare under this assumption. Here the function $T(f(n))$ expresses the time it takes to perform $f(n)$ operations, assuming one operation takes 100 ms, and the function $f(n)$ denotes the number of basic operations required for choosing a subset from a choice set of n items. For now we will only consider columns 1 through 5 (columns 6–8 of Table 2 are discussed later in the context of parameterized complexity theory).

As can be seen in Table 2, if the function $f(n)$ is *exponential* in n (e.g., $f(n) = 2^n$) then $T(f(n))$ grows ever so much faster than when $f(n)$ is *polynomial* in n (e.g., $T(n)$, $T(n^2)$, or $T(n^3)$). It is for this reason that we say a model that assumes on the order of 2^n basic operations is psychologically unrealistic for all but small n. Of course, which n is small enough depends not only on the speed of a single operation, but also on the time available for making the decision. Assume, for example, that a person takes at most, say, 5 min to choose a subset of pizza toppings from a set of 12 available toppings. Then a subset choice model that bases that decision on 2^n sequential 100 ms-operations can be rejected simply because it cannot explain how the decision is made in that time frame.

The reader perhaps wonders how much of our illustration hinges on the assumption of sequentiality. After all, it is possible that some cognitive operations can be performed in parallel, and this may allow for considerable speed-up in processing. Indeed, all else being equal, the range of feasible n is larger under a parallel model than under a serial model. Still the qualitative difference between polynomial-time and exponential-time computation remains important even if parallel processing is assumed (Frixione, 2001; Parberry, 1994, 1997; Tsotsos, 1990; van Rooij, 2003). Consider, for example, a decision maker who can perform any set of S operations in parallel. Then the time it takes him/her to perform $f(n)$ operations is given by $T(\frac{f(n)}{S})$. The speed-up due to S can be significant if $f(n)$ is a polynomial function, but its contribution is relatively small if $f(n)$ is a exponential function. To illustrate, assume again that $T(\frac{f(n)}{S})$ is to be at most 5 min and let n_c denote the *largest feasible* n obeying this time constraint (i.e., the maximum n such that $T(\frac{f(n)}{S}) < 5$ min). Then a change from the sequential model (i.e., with $S = 1$) to a parallel model with, say,

[3]For example: $1 + 2 + \cdots + x = x(x + 1)/2$ is $O(x^2)$; $x^4 + x^3 + x^2 + x + 1$ is $O(x^4)$; $x^4 + 2^x$ is $O(2^x)$; and 2^{x-1} is $O(2^x)$.

$S = 1000$,[4] leads to a significant increase in n_c from 14 to 144 for the *polynomial* function $f(n) = n^3$, but a relatively minor increase in n_c from 11 to 21 for the *exponential* function $f(n) = 2^n$. The take-home message is this: Even if one assumes that humans can perform (possibly many) basic operations in parallel, exponential-time models are generally unrealistic for all but relatively small n.

Adopting the convention that polynomial-time computation is considered tractable, while exponential-time computation is considered intractable (cf. Garey & Johnson, 1979), Generalized Subset Choice and Additive Subset Choice can be seen as two ends of the tractability continuum for subset choice. At the one extreme we have Additive Subset Choice, a prototypical example of computational tractability, requiring on the order of n operations to be performed. At the other extreme we have Generalized Subset Choice, a prototypical example of computational intractability, requiring on the order of 2^n operations to be performed. But where does Graph Subset Choice fit in? Fishburn and LaValle (1996) presented a result that implies that Graph Subset Choice is not computable in polynomial-time (unless P = NP, a condition that will be explained in Section 3). Does this mean that Graph Subset Choice is an unrealistic model of subset choice? The answer is "yes" and "no"; namely, "yes" because it means that Graph Subset Choice is computationally intractable *in its general form* (for reasons discussed above and illustrated in Table 2), but also "no" because some *restricted versions* of Graph Subset Choice may still be computationally tractable.

To explain the importance of the last qualification we consider a restricted version of Graph Subset Choice: Let the input graph $G = (V, E)$ be such that each vertex x takes only the value $+1$ (i.e., $u(x) = +1$, for all $x \in V$), and each edge takes only the value -1 (i.e., $\Delta(e) = -1$, for all $e \in E$). Then we call G a *unit-weighted conflict graph*, and the task of finding a maximum valued subset for G we call UCG Subset Choice:

UCG Subset Choice

Input: A unit-weighted conflict graph $G = (V, E)$ with vertex set $V = \{x_1, x_2, \ldots, x_n\}$ and edge set $E \subseteq V^2$. For every $x \in V$ there is an associated value $u(x) = +1$ and for every edge $(y, z) \in E$ there is an associated value $\Delta(y, z) = -1$.

Output: A subset $A \subseteq V$ such that $u(A) = \sum_{x \in A} u(x) + \sum_{(y,z) \in \mathrm{E}_G(A)} \Delta(y, z)$ is maximum.

Because the set of possible inputs for UCG Subset Choice is a proper subset of the set of possible inputs for Graph Subset Choice, and both problems have the same output, UCG Subset Choice is *a special case* of Graph Subset Choice, and we write UCG Subset Choice $\subseteq$ Graph Subset Choice. Note, however, that the finding by Fishburn and LaValle (1996)—that Graph Subset Choice is of non-polynomial time complexity (unless P = NP)—does not imply that the special case UCG Subset Choice is of non-polynomial time-complexity. If UCG Subset Choice would happen to be polynomial-time computable we should conclude as follows: Graph Subset Choice is a psychologically viable model of subset choice in situations where the decision-maker's value structure can be modeled by a unit-weighted conflict graph, but it is not necessarily a psychologically viable model in other situations.

As we prove in Section 4 below, UCG Subset Choice turns out to be of *non-polynomial* time-complexity (unless P = NP) just like Graph Subset Choice. This means that UCG Subset Choice is, in its general form, not essentially easier than Graph Subset Choice in its general form. Again we may ask if there exist, perhaps, restricted versions of UCG Subset Choice that are tractably computable. For illustrative purposes, let us consider the following restriction of Graph Subset Choice. Let $\deg_G(x)$ denote the number of edges in the graph $G = (V, E)$ that are incident to x (in Fig. 1, for example, $\deg_G(ham) = 3$ and $\deg_G(mushroom) = 1$). We call $\deg_G(x)$ the *degree* of x in G. Further, let $\delta_{\max}$ denote the largest degree of vertices in G (i.e., $\deg_G(x) \leqslant \delta_{\max}$ for all $x \in V$), and let the input graph $G = (V, E)$ be a *unit-weighted conflict graph* such that $\delta_{\max}$ is generally small and independent from the size of V. Now it makes sense to analyze the time-complexity of UCG Subset Choice as a function of *both* $n = |V|$ and $\delta_{\max}$, instead of only n. In this case, we say we analyze the *parameterized* complexity (Downey & Fellows, 1999) of the *parameterized* problem $\{\delta_{\max}\}$-UCG Subset Choice, defined as follows:

$\{\delta_{\max}\}$-UCG Subset Choice

Input: A unit-weighted conflict graph $G = (V, E)$ with vertex set $V = \{x_1, x_2, \ldots, x_n\}$ and edge set $E \subseteq V^2$. For every $x \in V$ there is an associated value

[4]We think that $S = 1000$ greatly overestimates the capacity of humans to perform in parallel the basic operations underlying the type of subset choice tasks we consider here. However, in other cognitive domains massive parallel processing may be more likely. For example, in the domain of vision, Tsotsos (1990) models visual search as the task of finding a subset of "pixels" in the visual field that sufficiently matches a visual template. Although significant parallel processing may underlie this task, Tsotsos nevertheless concludes that exponential-time models of visual search are generally unrealistic. To appreciate this conclusion one needs to take into account that n in visual search (i.e. the number of pixels in the visual field) is orders of magnitude larger than n in subset choice tasks such as choosing pizza toppings or selecting political representatives.

$u(x) = +1$ and for every edge $(y, z) \in E$ there is an associated value $\Delta(y, z) = -1$.

Parameter: The positive integer $\delta_{\max}$ (here $\delta_{\max}$ is the smallest integer such that $\deg_G(x) \leqslant \delta_{\max}$ for all $x \in V$).

Output: A subset $A \subseteq V$ such that $u(A) = \sum_{x \in A} u(x) + \sum_{(y,z) \in \mathrm{E}_G(A)} \Delta(y, z)$ is maximum.

Note that $\{\delta_{\max}\}$-UCG Subset Choice is not a special case of UCG Subset Choice—the two tasks take the same input and give exactly the same output. The only difference is the explicit statement of the parameter $\delta_{\max}$ in the formulation of $\{\delta_{\max}\}$-UCG Subset Choice, while this parameter is left implicit in the formulation of UCG Subset Choice. The explicit statement of the parameter in $\{\delta_{\max}\}$-UCG Subset Choice signals that we are interested in the number of basic operations required to compute UCG Subset Choice as a function of both n and $\delta_{\max}$.

Now two possibilities arise. We can either compute UCG Subset Choice in a number of steps that is $O(f(\delta_{\max})n^{\alpha})$, where α is a constant and f is any function depending only on $\delta_{\max}$, or we cannot. If it turns out that we can, then $\{\delta_{\max}\}$-UCG Subset Choice is said to be *fixed-parameter tractable* (fpt), and $\{\delta_{\max}\}$-UCG Subset Choice is said to be computable in *fpt-time*. If indeed $\{\delta_{\max}\}$-UCG Subset Choice is fixed-parameter tractable, then we conclude as follows: Graph Subset Choice is a psychologically viable model of subset choice in situations where the decision-maker's value structure can be modeled by a unit-weighted conflict graph and the maximum vertex degree is small (but, again, Graph Subset Choice is not necessarily a psychologically viable model in other situations). To illustrate, consider columns 6–8 in Table 2. Here n is again the size of the choice set and κ denotes a problem parameter (e.g., $\delta_{\max}$, as in our example). The illustration assumes that κ is small ($= 8$) and, again, processing speed is 100 ms per basic operation. Columns 6 and 7 show that the fpt-time functions $T(2^{\kappa}n)$ and $T(2^{\kappa} + n)$ grow much slower than the exponential-time function $T(2^n)$. This illustrates how subset choice problems of exponential-time complexity may still be psychologically feasible, even for *large* n, provided only that it is computable in fpt-time for a parameter that is relatively small. We remark that the requirement of fpt-time is not a luxury, not even for small parameters. As can be seen in Column 8 of Table 2, a running time that is *not* fpt-time, such as $O(n^{\kappa})$, is generally impractical even if κ is relatively small.

1.2. Overview

This research sets out to gain insight into the fundamental complexity of (some classes of) subset choice models. Additionally it aims to illustrate the use of the analytic tools of computational complexity theory for analyzing the psychological plausibility of subset choice models (or other cognitive models) in general. To make this paper as much self-contained as possible, we include brief reviews of the concepts and tools of both classical complexity theory (Garey & Johnson, 1979) and parameterized complexity theory (Downey & Fellows, 1999).

The remainder of this paper is organized as follows. We start, in Section 2, by generalizing the binary interaction model of subset choice proposed by Fishburn and LaValle (here called Graph Subset Choice) to an h-ary interaction model (called Hypergraph Subset Choice). Section 3 defines basic concepts and terminology of classical complexity theory and explains how this theory can be used for analyzing the complexity of subset choice models. Then, in Section 4, we present new classical complexity results for two special cases of Graph Subset Choice. Section 5 lays out the framework of parameterized complexity theory and explains how parametric techniques can be used to identify sources of non-polynomial time complexity in subset choice models. Subsequently, Sections 6–8 put these tools into practice for special cases of Hypergraph Subset Choice. Section 9 briefly comments on how our complexity results do not automatically generalize to subset choice problems that assume a bound on the size of the chosen set. We conclude with a discussion of our main results in Section 10.

2. Value-structures as hypergraphs

Fishburn and LaValle (1996) proposed a binary interaction model of subset choice in which a decision maker's value-structure is modeled by a weighted graph (Fig. 1). Their model improves upon the additive model by allowing interactions between pairs of items to play a role in subset valuation. Still, the model assumes that the value of a subset can be understood solely in terms of the value of its elements and the pairwise interactions (see Eq. (3)). That is, all interaction terms of order 3 and higher are assumed to be zero in the binary model.

In some situations, higher-order interactions appear essential for subset valuation. Consider, for example, the situation in which a consumer has to decide on a subset of computer parts to buy from a given set of alternatives (e.g., computer, monitor, keyboard, mouse, printer, scanner). None of the computer parts is very useful by itself, thus, reasonably $u(x) \leqslant 0$ for all $x \in \{computer, monitor, keyboard, printer, scanner\}$. Further note that a computer is not very useful without a monitor and a keyboard, a monitor is not very useful without a computer and a keyboard, and a keyboard is not very useful without a computer and a monitor. Therefore probably no added value is associated with any *pair* in the set $\{computer, monitor, keyboard\}$; i.e.,

$\Delta(x,y)=0$ for all $x,y \in \{computer, monitor, keyboard\}$. Despite all this it may still be the case that the subset consisting of computer, monitor, and keyboard is of considerable value; i.e., $u(\{computer, monitor, keyboard\})>0$. The relationship between subset valuations in this illustration is entirely conceivable from a psychological perspective, but the binary interaction model cannot capture it. Namely, the binary interaction model prescribes that if $u(x)\leqslant 0$ for all $x \in \{computer, monitor, keyboard\}$ and $\Delta(x,y)=0$ for all $x,y \in \{computer, monitor, keyboard\}$, then $u(\{computer, monitor, keyboard\}) = u(computer) + u(monitor) + u(keyboard) + \Delta(computer, monitor) + \Delta(computer, keyboard) + \Delta(monitor, keyboard) \leqslant 0$.

To incorporate the possibility of higher-order interaction between items in subset choice, we generalize Fishburn and LaValle's binary interaction model to an h-ary interaction model. In the h-ary interaction model a decision maker's value-structure is modeled by a weighted *hyper*graph. A hypergraph is a generalization of the concept of a graph. Whereas in a graph $G=(V,E)$, the set E consists of unordered *pairs* of distinct vertices (i.e., $E \subseteq V \times V$), in a hypergraph $H=(V,E)$, the set E consists of unordered h-tuples of distinct vertices, $2 \leqslant h \leqslant |V|$ (i.e., $E \subseteq \bigcup_{2\leqslant h\leqslant |V|} V^h$, where V^h denotes the h-fold product of V). In other words, a graph is a special type of hypergraph—viz., one in which E contains only 2-tuples of vertices. In a hypergaph, we call an element in E a *hyperedge*.

In the h-ary interaction model of subset choice, each vertex $x \in V$ has an associated *vertex weight*, $u(x)$, and each hyperedge $e=(x_1,x_2,\ldots,x_h)$, $e \in E$, has associated *hyperedge weight*, $\Delta(e)$. For simplicity, we assume that $u(x) \in \mathbb{Z}$, and $\Delta(e) \in \mathbb{Z}\backslash\{0\}$ (here $\mathbb{Z}$ denotes the set of integers, and $\mathbb{Z}\backslash\{0\}$ denotes the set of non-zero integers). A vertex weight $u(x)$ represents the value of choice alternative x when evaluated in isolation. A hyperedge weight $\Delta(e)$, with $e=(x_1,x_2,\ldots,x_h)$, represents the added value of the combination of vertices $x_1,x_2,\ldots,x_h$ over and above the value of the singular elements $x_1,x_2,\ldots,$ and x_h and over and above the value of all hyperedges that are combinations of at most $h-1$ vertices in $\{x_1,x_2,\ldots,x_h\}$. In other words, if $A=\{x_1,x_2,\ldots,x_h\}$ is a set of h vertices with value $u(A)$, then the interaction weight associated with the hyperedge $(x_1,x_2,\ldots,x_h)$ is given by

$$\Delta(x_1,x_2,\ldots,x_h) = u(A) - \sum_{x\in A} u(x) - \sum_{e\in A^2\cup A^3\cup\cdots\cup A^{h-1}, e\in E} \Delta(e).$$

Having defined the hyperedge weight this way, the value $u(A)$ associated with choosing subset $A \subseteq V$ in the h-ary model is given by

$$u(A) = \sum_{x\in A} u(x) + \sum_{e\in \mathrm{E}_H(A)} \Delta(e), \tag{4}$$

where $\mathrm{E}_H(A) = \{(x_1,x_2,\ldots,x_h) \in E | x_1,x_2,\ldots,x_h \in A\}$ denotes the set of all hyperedges whose endpoints are contained in A. Using Eq. (4) we formulate the problem Hypergraph Subset Choice as follows:

Hypergraph Subset Choice

Input: A weighted hypergraph $H=(V,E)$, $E \subseteq \bigcup_{2\leqslant h\leqslant |V|} V^h$. For every $x \in V$ there is a weight $u(x) \in \mathbb{Z}$ and for every $e \in E$ there is a weight $\Delta(e) \in \mathbb{Z}\backslash\{0\}$.
Output: A subset $A \subseteq V$ such that $u(A) = \sum_{x\in A} u(x) + \sum_{e\in \mathrm{E}_H(A)} \Delta(e)$ is maximum.

Note that Graph Subset Choice is a special case of Hypergraph Subset Choice; viz., the special case in which H in the input is a weighted graph.

We define graph-theoretic notation and terminology that we use in the remainder of the paper: Let $H=(V,E)$ be a hypergraph. We say a vertex $x \in V$ is *incident* to hyperedge $e=(x_1,x_2,\ldots,x_h)$ and, conversely, e is incident to x, if $e \in E$ and $x \in \{x_1,x_2,\ldots,x_h\}$. The *degree* of a vertex $x \in V$ is the total number of hyperedges in H that are incident to x, denoted $\deg_H(x)$. A vertex $x \in V$ is called a *singleton* if $\deg_H(x)=0$ and is called *pendant* if $\deg_H(x)=1$. The *span* of an hyperedge $e=(x_1,x_2,\ldots,x_h)$ is denoted by $\mathrm{span}(e)=h$. Two vertices $x,y \in V$ are *neighbors* in H if there exists an hyperedge $(x_1,x_2,\ldots,x_h) \in E$ with $x,y \in \{x_1,x_2,\ldots,x_h\}$. The (open) *neighborhood* $\mathrm{N}_H(x)$ is the set of neighbors of x in H, and the *closed neighborhood* $\mathrm{N}_H[x] = \mathrm{N}_H(x) \cup \{x\}$. The *set difference* of two sets V and W is denoted by $V\backslash W = \{x \in V | x \notin W\}$.

Note that for any specific hypergraph $H=(V,E)$ there exists a positive integer $\varepsilon \leqslant |V|$ such that $\mathrm{span}_H(e) \leqslant \varepsilon$ for all $e \in E$. In the special case that $\varepsilon = 2$ (i.e., $E \subseteq V^2$), we will write $G=(V,E)$ instead of $H=(V,E)$, and call G a *graph* and $e \in E$ an *edge*. Note that $\deg_G(x) = |\mathrm{N}_G(x)|$. We define the following terms specifically for graphs. A sequence $\langle x_1,x_2,\ldots,x_k\rangle$ of pairwise distinct vertices with $(x_1,x_2),(x_2,x_3),\ldots,(x_{k-1},x_k) \in E$ is called a *path* in G. If $x_1=x_k$ and $k \geqslant 3$ then $\langle x_1,x_2,\ldots,x_k\rangle$ is called a *cycle* in G. A graph is *connected* if for every pair of vertices $x,y \in V$ there is a path in G from x to y. An acyclic graph is called a *forest* and a connected forest is called a *tree*. A *rooted tree* is a tree with a designated vertex called the *root*. Let $T=(V_T,E_T)$ be a rooted tree, with root $r \in V_T$. A pendant vertex in a tree is called a *leaf*. For two vertices $x,y \in V_T$, with $(x,y) \in E_T$, we say x is *parent* of y, and y is *child* of x, if $\langle r,\ldots,x,y\rangle$ is a path in T. A graph $G'=(V',E')$ is a *subgraph* of $G=(V,E)$ if $V' \subseteq V$ and $E' \subseteq E$. For any $V' \subseteq V$, $G'=(V',\mathrm{E}_H(V'))$

Table 3
Overview of the special value-structures consider in this article

Value-structure	$u(x)$	$\Delta(e)$	span(e)
Unit-weighted conflict graph (UCG)	$+1$	-1	2
Unit-weighted surplus graph (USG)	-1	$+1$	2
Edge-weighted conflict graph (ECG)	$+1$	$\leqslant -1$	2
Vertex-weighted conflict graph (VCG)	$\geqslant 1$	-1	2
Conflict graph (CG)	$\geqslant 1$	$\leqslant -1$	2
Conflict hypergraph (CH)	$\geqslant 1$	$\leqslant -1$	$\leqslant \lvert V\rvert$

Note: $u(x)$ denotes the weight of a vertex x, $\Delta(e)$ denotes the weight on a hyperedge e, and span(e) denotes the span of a hyperedge e.

is called the subgraph of G *induced by* V'. Let $G^* = (V^*, E^*)$ be a subgraph of G. We say G^* is a *component* of G if (1) G^* is connected, and (2) there does not exist a subgraph G' of G, $G^* \neq G'$, such that G' is connected and G^* is a subgraph of G'.

In our analyses we consider value-structures that can be represented by hypergraphs with special weighting functions. Let $H = (V, E)$ be a weighted hypergraph, then we define the following classes for hypergraphs. (1) We say H is a *unit-weighted* hypergraph if $u(x) \in \{-1, +1\}$ for all $x \in V$ and $\Delta(e) \in \{-1, +1\}$ for all $e \in E$. (2) We say H is an *edge-weighted* hypergraph if $u(x) \in \{-1, +1\}$ for all $x \in V$ and $\Delta(e) \in \mathbb{Z}\backslash\{0\}$ for all $e \in E$. (3) We say H is a *vertex-weighted* hypergraph if $u(x) \in \mathbb{Z}$ and $\Delta(e) \in \{-1, +1\}$ for all $e \in E$. (4) We say H is a *conflict* hypergraph if $u(x) \geqslant 0$ for all $x \in V$ and $\Delta(e) \leqslant -1$ for all $e \in E$. (5) We say H is a *surplus* hypergraph if $u(x) \leqslant 0$ for all $x \in V$ and $\Delta(e) \geqslant +1$ for all $e \in E$. Conjunctions of these classes provide further restrictions on the value-structure (e.g., edge-weighted conflict graphs, unit-weighted surplus hypergraphs, etc.). Table 3 gives an overview of the classes considered in our analyses.

3. Classical complexity theory

This section introduces the basic concepts and terminology of classical complexity theory.[5] For more details the reader is referred to Garey and Johnson (1979), Karp (1975), and Papadimitriou and Steiglitz (1988). Cognitive psychologists may find treatments by Frixione (2001), Parberry (1997), and Tsotsos (1990) particularly illustrative.

[5]The reader is advised that what we call classical complexity theory is typically referred to as (computational) complexity theory in both the computer science and cognitive science literature. Because we wish to contrast this theory with a younger branch of complexity theory, called parameterized complexity theory and discussed in Section 5, we refer to the earlier theory as "classical."

3.1. Optimization and decision problems

Complexity theory distinguishes between *optimization problems* and *decision problems*. In an optimization problem, the required output is one that is optimized (either maximized or minimized) on some dimension. We already encountered examples of optimization problems in the form of Additive Subset Choice, Generalized Subset Choice, Graph Subset Choice, and Hypergraph Subset Choice. In each case, the output is a subset that is maximized with respect to subset value. A decision problem, on the other hand, states a Yes/No-question and its required output is the answer to that question.

For any given optimization problem we can formulate a decision version as follows: Let Π be an optimization problem that, given a choice set V, asks for a subset $A \subseteq V$ such that $u(A)$ is maximized (respectively, minimized). Its decision version, Π_D, then asks if there exists a subset $A \subseteq V$ such that $u(A) \geqslant p$ (respectively, $u(A) \leqslant p$), for some preset threshold p. Below we state the decision versions of Hypergraph Subset Choice and its special case Graph Subset Choice. (The decision versions of Additive Subset Choice and Generalized Subset Choice can be analogously derived.)

Hypergraph Subset Choice (*decision version*)

Input: A weighted hypergraph $H = (V, E)$, $E \subseteq \bigcup_{2 \leqslant h \leqslant |V|} V^h$. For every $x \in V$ there is a weight $u(x) \in \mathbb{Z}$ and for every $e \in E$ there is a weight $\Delta(e) \in \mathbb{Z}\backslash\{0\}$. A positive integer p.
Question: Does there exist a subset $A \subseteq V$ such that $u(A) = \sum_{x \in A} u(x) + \sum_{e \in \mathrm{E}_H(A)} \Delta(e) \geqslant p$?

Graph Subset Choice (*decision version*)

Input: A graph $G = (V, E)$ with vertex set $V = \{x_1, x_2, \ldots, x_n\}$and edge set $E \subseteq V^2$. For every $x \in V$ there is an associated integer value $u(x)$ and for every edge $(y, z) \in E$ there is an associated non-zero integer value $\Delta(y, z)$. A positive integer p.
Question: Does there exist a subset $A \subseteq V$ such that $u(A) = \sum_{x \in A} u(x) + \sum_{(y,z) \in \mathrm{E}_G(A)} \Delta(y, z) \geqslant p$?

Solving a decision problem consists of correctly responding either "yes" or "no." An input i for a decision problem Π_D is called a *yes-instance* for Π_D if the answer to the question posed by Π_D is "yes" for i. Otherwise, i is called a *no-instance*. If an algorithm for a decision problem returns a possible solution (e.g., in the case of Hypergraph Subset Choice this would be a subset A with $u(A) \geqslant p$) whenever the answer to the question is "yes," then we say the algorithm is *constructive*. All algorithms that we consider are constructive.

Note that there is a strong relationship between the time-complexity of an optimization problem and its

decision version. Clearly, solving a decision problem cannot be more difficult than solving its corresponding optimization version: viz., if we know a solution to the optimization version of Hypergraph Subset Choice (i.e., we know a subset $A \subseteq V$ such that $u(A)$ is maximized), then we also know if there exists a subset with $u(A) \geqslant p$, for any p. The converse, that solving an optimization problem is computationally not much harder than solving its decision version, is perhaps less obvious. Note, however, that we can easily determine a maximum valued subset by solving the decision version of Hypergraph Subset Choice for different p: viz., if we find that there exists a subset $A \subseteq V$ with $u(A) \geqslant p$ for $p = c$ but not for $p = c + 1$, then we can conclude that $A \subseteq V$ with $u(A) = c$ is a maximum valued subset, and hence A is a solution to the optimization version of Hypergraph Subset Choice. This means that the optimization version of Hypergraph Subset Choice is solvable in polynomial time, if and only if the decision version is solvable in polynomial time. This observation enables us to use the theory of NP-completeness and its accompanying notion of polynomial-time reducibility to assess the computational complexity of Hypergraph Subset Choice.

3.2. NP-completeness and polynomial-time reduction

In the theory of NP-completeness, a distinction is made between (1) decision problems that are solvable in (*deterministic*) *polynomial time* (class P) and (2) decision problems that are solvable in *non-deterministic polynomial time* (class NP), with $\mathrm{P} \subseteq \mathrm{NP}$. A decision problem Π is a member of the class P, $\Pi \in \mathrm{P}$, if there exists an algorithm that solves Π for any possible input i in time $O(n^{\alpha})$ (where α is a constant and $|i| = n$). A decision problem Π is a member of the class NP, $\Pi \in \mathrm{NP}$, if, given a yes-instance i for Π and a candidate solution for i, we can *verify* in polynomial time whether the candidate solution is indeed a solution for i. For example, given a yes-instance for Graph Subset Choice and a candidate solution $A \subseteq V$, we can compute $u(A) = \sum_{x \in A} u(x) + \sum_{(y,z) \in \mathrm{E}_G(A)} \Delta(y, z)$, and compare $u(A)$ to p, in time $O(|V|^2)$.

It is widely believed that there exist problems in NP that are not in P, and thus that $\mathrm{P} \neq \mathrm{NP}$. This conjecture is motivated, among other things, by the existence of so-called NP-hard problems. To explain NP-hardness we define the notion of polynomial-time reducibility: For decision problems Π_1 and Π_2 we say that Π_1 *reduces* to Π_2 if there exists an algorithm that transforms any input i_1 for Π_1 into an input i_2 for Π_2 such that input i_1 is a yes-instance for Π_1 if and only if input i_2 is a yes-instance for Π_2. We say the reduction is a *polynomial-time reduction* if the algorithm performing this transformation runs in polynomial time. A decision problem Π is NP-*hard* if every problem in NP can be polynomial-time reduced to Π. Thus, NP-hard problems are not in P unless $\mathrm{P} = \mathrm{NP}$. Problems that are NP-hard *and* members of NP are called NP-*complete*.

The technique of polynomial-time reduction is very useful. Once a problem is known to be NP-hard we can prove other problems to be NP-hard by polynomial-time reducing it to these other problems. Today hundreds of problems are known to be NP-hard (including Graph Subset Choice, and thus also Hypergraph Subset Choice). Despite great efforts from many computer scientists, nobody to date has succeeded in finding a polynomial-time algorithm that solves an NP-hard problem (hence, the belief that $\mathrm{P} \neq \mathrm{NP}$). Therefore, the finding that a decision problem Π is NP-hard is seen as very strong evidence that all algorithms solving Π run at best in exponential-time; e.g., in time $O(\alpha^n)$, where α is a constant and n is the input size.

4. Classical complexity results

Fishburn and LaValle (1996) already considered the time-complexity of Graph Subset Choice. In their paper they sketched a polynomial-time reduction from the problem Independent Set to Graph Subset Choice. A subset $A \subseteq V$ is called an *independent set* for a graph $G = (V, E)$ if no two vertices in A are adjacent (i.e., $\forall x, y \in A, (x, y) \notin E$). The problem Independent Set is then defined as follows.

Independent Set (*decision version*)

Input: A graph $G = (V, E)$ and a positive integer k.

Question: Does there exist an independent set $A \subseteq V$ for G with $|A| \geqslant k$?

Independent Set is known to be NP-complete (Garey & Johnson, 1979). As a consequence the polynomial-time reduction described by Fishburn and LaValle (1996) establishes the following theorem.

Theorem 1. (Fishburn & LaValle, 1996). *Graph Subset Choice is NP-hard.*

With Lemma 1 we also present a reduction from Independent Set to Graph Subset Choice, but a different one than described by Fishburn and LaValle (1996). From Lemma 1 we conclude an even stronger result than Theorem 1 (see Corollary 1).

Lemma 1. *Let the graph $G = (V, E)$ and the positive integer k form an instance for Independent Set. Then we define an instance for Graph Subset Choice, consisting of a weighted graph G^* and a positive integer p, as follows. Let $G^* = (V^*, E^*)$ with $V^* = V$ and $E^* = E$. Further, for every $x \in V^*$ let $u(x) = +1$ and for every $e \in E^*$ let $\Delta(e) = -1$. Let $p = k$. Then G and k form a yes-instance*

for Independent Set if and only if G^* and p form a yes-instance for Graph Subset Choice.

Proof. ($\Rightarrow$) Let (G, k) be a yes-instance for Independent Set. Then there exists an independent set $A \subseteq V$ for G with $|A| \geqslant k$. Since $G^* = G$, A is also an independent set for G^*. This means that $E_{G^*}(A) = \{(x, y) \in E^* : x, y \in A\} = \varnothing$ and thus $u(A) = \sum_{x \in A} u(x) + \sum_{e \in E_{G^*}(A)} \Delta(e) = |A| - |E_{G^*}(A)| = |A| \geqslant k = p$. We conclude (G^*, p) is a yes-instance for Graph Subset Choice. ($\Leftarrow$) Let (G^*, p) be a yes-instance for Graph Subset Choice. Then there exists a subset $A \subseteq V^*$ with $u(A) \geqslant p$. We distinguish two cases: (1) If $E_{G^*}(A) = \varnothing$ then $E_G(A) = \varnothing$ and thus A is an independent set for G, with $|A| \geqslant p = k$. We conclude that (G, k) is a yes-instance of Independent Set. (2) If $E_{G^*}(A) = \varnothing$ then $E_G(A) \neq \varnothing$. We transform A into an independent set A' for G using the following algorithm:

1. $A' \leftarrow A$
2. **while** $E_G(A') \neq \varnothing$ **do**
3. pick an edge $(x, y) \in E_G(A')$
4. $A' \leftarrow A' \backslash \{x\}$
5. **end while**
6. **return** A'

The algorithm considers each edge in G at most once and thus runs in time $O(|E|)$ or $O(|V|^2)$. Note that every call of line 4 results in the removal of at least one edge from $E_G(A')$. Hence, $u(A') \geqslant u(A) \geqslant p$. Furthermore, when the algorithm halts then $E_G(A') = \varnothing$ and thus A' is an independent set of size at least $p = k$ for G. We conclude that (G, k) is a yes-instance for Independent Set. □

Note that the reduction in Lemma 1 is a polynomial-time reduction. Namely, we can copy every element in V to V^* and E to E^* in time $O(|V|^2)$, we can set $p = k$ in time $O(1)$, and we can assign each vertex in V the weight '1' and assign each edge in E the weight '-1' in time $O(|V|^2)$. Further, the algorithm that, given a subset with value at least p, computes an independent set of size at least $k = p$, runs in time $O(|V|^2)$. Also note that G^* in Lemma 1 is a very special type of weighted graph, viz., one in which $u(x) = 1$ for all $x \in V$ and $\Delta(e) = -1$ for all $e \in E$. In other words, G^* is a *unit-weighted conflict graph* as discussed in the Introduction (see also Table 3). The polynomial-time reduction in Lemma 1 thus shows that Graph Subset Choice is NP-hard even in the restricted case where the input can be represented by a unit-weighted conflict graph:[6]

[6]Note that any value-structure $G = (V, E)$, $E \subseteq V^2$, with integer weight $u(x) = \alpha$, for constant $\alpha \geqslant 1$, and $\Delta(e) = -u(x)$, for all $x \in V$ and all $e \in E$, can be represented by a unit-weighted conflict graph $G^* = (V^*, E^*)$. In general, any value structure $H = (V, E)$ with $u(x) \in \mathbb{Z}$ for all $x \in V$ and $\Delta(e) \in \mathbb{Z} \backslash \{0\}$ for all $e \in E$, can be modeled by a hypergraph $H^* = (V^*, E^*)$, with $u^*(x) = \frac{u(x)}{d}$ for all $x \in V^*$ and $\Delta^*(e) = \frac{\Delta(e)}{d}$ for all $e \in E^*$, where d is a *common divisor* of all vertex and hyperedge weight values in H. There exist a subset $A \subseteq V$ with $u(A) = p$ for H if and only if $u^*(A) = \frac{p}{d}$ for H^*.

Corollary 1. *UCG Subset Choice is* NP*-hard.*

As a comparison, we also consider the restricted version of Graph Subset Choice that only takes *unit-weighted surplus graphs* as inputs (see Table 3). We call this special case USG Subset Choice. Note that the only difference between unit-weighted conflict graphs and unit-weighted surplus graphs is a reversal in sign of vertex- and edge-weights. Interestingly, this reversal in sign leads to a version of subset choice that is polynomial-time solvable.

Theorem 2. *USG Subset Choice* $\in$ P.

Theorem 2 follows directly from the following lemma:

Lemma 2. *Let $G = (V, E)$ be an instance USG Subset Choice. Let $C_i = (V_i, E_i)$, $1 \leqslant i \leqslant l$, be the components of G. Let U denote the set of components in G that contain at least one cycle. Then $A = \{x \in V | x \in V_i$ and $C_i \in U$, $1 \leqslant i \leqslant l\}$ has maximum value $u(A)$.*

Proof. Let $A = \{x \in V | x \in V_i$ and $C_i \in U, 1 \leqslant i \leqslant l\}$. We show in two steps that $u(A)$ is maximum. We show that (1) we cannot improve the value of any vertex set $A^* \subseteq V$ by including vertices from $V \backslash A$ in A^*. We conclude that there is a vertex set with maximum value that does not contain any vertices in $V \backslash A$. Then we show that (2) there does not exist a set $A^* \subset A$ with $u(A^*) > u(A)$.

(1) Let $A^* \subseteq V$. Consider the subgraph F of G induced by vertex set $V \backslash A$. Then by the definition of A, $F = (V \backslash A, E_G(V \backslash A))$ is a forest. In any forest the number of edges is smaller than the number of vertices (for a proof of this basic property see e.g. Gross and Yellen (1999)). Since every subgraph of a forest is also a forest, we know that for any subset $A' \subseteq V \backslash A$, $|E_G(A')| < |A'|$. Therefore, for any subset $A' \subseteq V \backslash A$, $\text{value}_G(A') = |E_G(A')| - |A'| < 0$. Consider $A^* \cup A'$. Then $\text{value}_G(A^* \cup A') < \text{value}_G(A^*)$. In other words, we can never improve the value of a vertex set $A^* \subseteq V$ by including vertices in $V \backslash A$.

(2) From the above we know that *if* the value of A is *not* maximum, then there must exist a set $A^* \subset A$ with $u(A^*) > u(A)$. We show by contradiction that such a set cannot exist. Let $A^* \subset A$ be a largest vertex set with $u(A^*) > u(A)$. Then at least one of the following two situations is true. (a) There exists a vertex $x \in A \backslash A^*$ such that x has at least one neighbor $y \in A^*$. But then $u(A^* \cup \{x\}) \geqslant u(A^*) + |E_G(\{x, y\})| - |\{x\}| = u(A^*) + |\{(x, y)\}| - |\{x\}| = u(A^*) + (1-1) = u(A^*)$, contradicting the claim that A^* is a largest vertex set with $u(A^*) > u(A)$. (b) There exists a cycle $\langle v_1, v_2, \ldots, v_k, v_1 \rangle$ with $v_1, v_2, \ldots, v_k \in A \backslash A^*$. But then $u(A^* \cup \{x_1, x_2, \ldots, x_k\}) \geqslant u(A^*) + |E_G(\{x_1, x_2, \ldots, x_k\})| - |\{x_1, x_2, \ldots, x_k\}| \geqslant u(A^*) + k - k \geqslant u(A^*)$, again contradicting the claim that A^* is a largest vertex set with $u(A^*) > u(A)$. □

The lemma implies that we can solve USG Subset Choice by simply choosing all the vertices in every component of G that contains at least one cycle; we then check whether this set of vertices has at least p value in G. Since the components and cycles in a graph can be found in polynomial time (Brandstädt, Le, & Spinrad, 1999; Gross & Yellen, 1999), Theorem 2 follows.

We have shown that, although Graph Subset Choice is computationally tractable for some very restricted inputs (Theorem 2), in its general form it is of non-polynomial time-complexity (unless P = NP; Theorem 1). Furthermore, even the special case UCG Subset Choice requires non-polynomial time to be computed (unless P = NP; Corollary 1), meaning that the same holds for Hypergraph Subset Choice on any generalization of the unit-weighted conflict graph (i.e., for all value-structures in Table 3, except for unit-weighted surplus graphs).

5. Parameterized complexity theory

This section introduces the basic concepts and terminology of parameterized complexity theory. For more details the reader is referred to Downey and Fellows (1999), Downey, Fellows, and Stege (1999a, b), and Fellows (2001, 2002). Cognitive psychologists may find treatments by van Rooij (2003) and Wareham (1998) particularly illustrative.

5.1. Fixed-parameter (in)tractability

We denote a parameter set by $\kappa = \{k_1, k_2, \ldots, k_m\}$, where each k_i, $i = 1, 2, \ldots, m$, denotes an aspect of the input. We denote a problem Π parameterized on κ by κ-Π, and call κ-Π a *parameterized problem*. An instance for κ-Π, with input i and parameter κ, is denoted by a tuple (i, κ).

When a problem is shown to be NP-hard (like Hypergraph Subset Choice) it is important to understand which aspects of the input are actually responsible for the non-polynomial time behavior of the problem. The theory of parameterized complexity is motivated by the following observation. Some NP-hard problems can be solved by algorithms whose running time is non-polynomial in some parameter κ but polynomial in the input size $|i|$. In other words, the main part of the input contributes to the overall complexity in a "good" way, while only κ contributes to the overall complexity in a "bad" way. In these cases, we say κ *confines* the non-polynomial time complexity in the problem, and the problem is said to be fixed-parameter tractable for parameter κ.

More formally, a parameterized problem κ-Π is said to be *fixed-parameter tractable* if any instance (i, κ) for κ-Π can be decided in time $O(f(\kappa)n^{\alpha})$, where $f(\kappa)$ is a function depending only on κ. An algorithm that solves a parameterized problem κ-Π in time $O(f(\kappa)n^{\alpha})$ is called a fixed-parameter tractable (fpt-) algorithm. Parameterized decision problems that are not fixed-parameter tractable are called *fixed-parameter intractable*. Analogous to the classical complexity classes P and NP, parameterized complexity theory introduces the parameterized complexity classes FPT and W[1], with FPT $\subseteq$ W[1].[7] Fixed-parameter tractable parameterized problems are said to be in the class FPT. It is widely believed that there exist parameterized problems in W[1] that are fixed-parameter intractable, and thus that FPT $\neq$ W[1]. This conjecture is, among other things, motivated by the observation that there exist W[1]-hard problems.

To explain W[1]-hardness we define the notion of parametric reduction: For parameterized problems κ_1-Π_1 and κ_2-Π_2 and functions f and g, we say a *parametric reduction* from κ_1-Π_1 to κ_2-Π_2 is a reduction that transforms any instance i_1 for κ_1-Π_1 into an instance i_2 for κ_2-Π_2 with $\kappa_2 = g(\kappa_1)$. Further, the reduction runs in time $f(\kappa)|i_1|^{\alpha}$, where α is a constant. A parameterized problem κ-Π is said to be W[1]-*hard* if any parameterized problem κ'-$\Pi' \in$ W[1] can be transformed to κ-Π via a parametric reduction. Problems that are W[1]-hard *and* in W[1] are called W[1]-*complete*. Since membership of a W[1]-hard problem in FPT would imply that FPT = W[1], the finding that a problem is W[1]-hard is seen as very strong evidence that the problem is not in FPT.

It is important to realize that in classical complexity theory we analyze and classify the complexity of *problems*, while in parameterized complexity theory we analyze and classify the complexity of a given problem with respect to different *parameters*, i.e., parameterized problems. Thus, one and the same problem may be in FPT for one parameter but not in FPT for another.

5.2. Bounded search tree technique

There exist several techniques for constructing fpt-algorithms (see e.g., Downey & Fellows, 1999; Fellows, 2001, 2002; Fellows, McCartin, Rosamond, & Stege, 2000; Niedermeier, 2002). In this paper we use the *bounded search tree* technique. A *search tree* T is created as follows (see Fig. 2 for an illustration): First, the root of the tree, denoted s, is labeled by the parameterized instance (i, κ). Then we recursively apply a *branching rule* that creates a set children of s, denotes $s_1, s_2, \ldots, s_d$, and labels them by new instances, $(i_1, \kappa_1), (i_2, \kappa_2), \ldots, (i_d, \kappa_d)$, such that i is a yes-instance

[7] See, for example, Downey and Fellows (1999) for a definition of the class W[1].

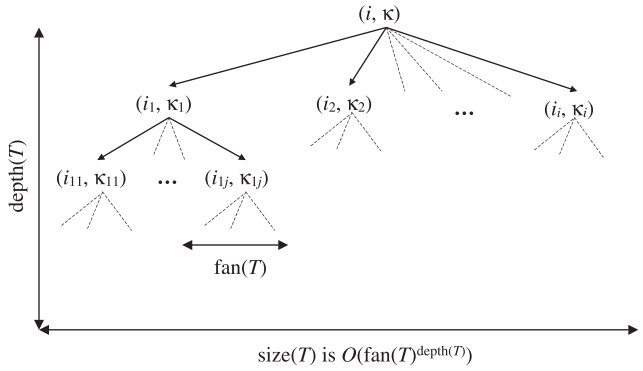

Fig. 2. A graphical illustration of a bounded search tree T for a parameterized problem κ-Π. The root of T is labeled by the to-be-decided instance (i, κ) for κ-Π. Each node s in T, labeled by an instance (i_s, κ_s), has children $s_1, s_2, \ldots, s_k$, labeled by $(i_{s_1}, \kappa_{s_1}), (i_{s_2}, \kappa_{s_2}), \ldots, (i_{s_k}, \kappa_{s_k})$, such that (i_s, κ_s) is a yes-instance for κ-Π if and only if at least one of $(i_{s_1}, \kappa_{s_1}), (i_{s_2}, \kappa_{s_2}), \ldots, (i_{s_k}, \kappa_{s_k})$ is a yes-instance for κ-Π. If the length of the longest path from the root to a leaf in T is depth(T) and the maximum number of children per node in T is fan(T), then the size of the whole tree, size(T), is $O(\text{fan}(T)^{\text{depth}(T)})$.

Table 4
Overview of the input parameters considered in this article

Parameter	Definition
ε	Smallest positive integer such that, for all $e \in E$, $\text{span}_H(e) \leqslant \varepsilon$
$u_{\max}$	Smallest positive integer such that, for all $x \in V$, $u(x) \leqslant u_{\max}$
$u_{\min}$	Smallest positive integer such that, for all $x \in V$, $u(x) \geqslant -u_{\min}$
$\Delta_{\max}$	Smallest positive integer such that, for all $e \in E$, $\Delta(e) \leqslant \Delta_{\max}$
$\Delta_{\min}$	Smallest positive integer such that, for all $e \in E$, $\Delta(e) \leqslant -\Delta_{\min}$
$\delta_{\max}$	Smallest positive integer such that, for all $x \in V$, $\deg_H(x) \leqslant \delta_{\max}$
$\text{N}_{\max}$	Smallest positive integer such that, for all $x \in V$, $\text{N}_H(x) \leqslant \text{N}_{\max}$
P	A positive integer
k	A positive integer
q	$q = p - u(V)$

for κ-Π if and only if i_1 or i_2 or ... or i_d is a yes-instance for κ-Π. For each newly created node in the search tree, the branching rule is applied again, until a node s_j is encountered that is labeled by either an identifiable yes-instance or an identifiable no-instance. As soon as a yes-instance is encountered the algorithm halts and returns the answer "yes." If the algorithm halts without returning the answer "yes" (i.e., if all leaves of the search tree are labeled by identifiable no-instances) then the algorithm returns the answer "no."

We denote the maximum number of children created per node in a search tree by fan(T) and the maximum length of the path from root to any leaf in T by depth(T). The total number of nodes in the search tree is denoted by size(T). Note that size(T) is bounded by $2\text{fan}(T)^{\text{depth}(T)}-1$ which is $O(\text{fan}(T)^{\text{depth}(T)})$ .The goal of the bounded search tree technique is to construct the search tree such that for every instance i for κ-Π, the following two conditions are met: (1) each node in the search tree can be created and labeled in fpt-time, $O(g(\kappa)n^{\alpha})$, and (2) the size of the search tree is bounded by some function $h(\kappa)$. These two conditions ensure that the bounded search tree algorithm runs in fpt-time. Namely, let $O(g(\kappa)n^{\alpha})$ be the time required to create a node in T and let size(T) $\leqslant h(\kappa)$, then the bounded search tree algorithm runs in time $O(h(\kappa)g(\kappa)n^{\alpha}) = O(f(\kappa)n^{\alpha})$.

5.3. Identifying sources of non-polynomial time-complexity

Consider a problem Π with parameter κ such that $\Pi \notin \text{P}$ and κ-$\Pi \in \text{FPT}$. Although clearly κ is *sufficient* for confining the non-polynomial complexity inherent in Π, it may contain one or more parameters that are 'redundant,' in the sense that the fpt-classification of κ-Π does not depend on those parameters. Namely, for a problem Π and parameter set κ, if κ-$\Pi \in \text{FPT}$ then κ'-$\Pi \in \text{FPT}$ for all $\kappa' \supseteq \kappa$. To distinguish between parameter sets that exactly circumscribe a source of non-polynomial time complexity in Π and parameter sets that merely contain such a source, we define the notion of a minimal parameter set (cf. Wareham's (1998) notion of *intractability map*).

A parameter set κ is said to be *minimal* if and only if κ-$\Pi \in \text{FPT}$ and there does *not* exist a proper subset $\kappa' \subset \kappa$ such that κ'-$\Pi \in \text{FPT}$. If κ is a minimal parameter set for a classically intractable problem Π we also call κ a *crucial source of* (*non-polynomial time*) *complexity* in Π, because then Π is "easy" for small values of parameters in κ *irrespective* the size of other input parameters. Note that a crucial source of complexity need not be unique. That is, there may exist different parameter sets κ and κ', with $\kappa' \not\subset \kappa$, such that κ-$\Pi \in \text{FPT}$ and κ'-$\Pi \in \text{FPT}$. Also note that every problem has $\kappa = \{|i|\}$ as a crucial source of complexity. Thus the label 'crucial source of complexity' should be read as denoting a *minimal* set of parameters that is *sufficient*, but not necessary, for confining the non-polynomial time behavior in a problem.

In the following Sections (6–8) we set out to identify some crucial sources of complexity in Hypergraph Subset Choice, using the techniques of parametric reduction and bounded search tree. In our analyses we consider several parameters (see Table 4 for an overview).

6. Subset choice on unit-weighted conflict graphs

In Section 3 we saw that Hypergraph Subset Choice is NP-hard even for the special case where the value-

structure can be modeled by a unit-weighted conflict graph, called UCG Subset Choice (Corollary 1). In this section we investigate the parameterized complexity of UCG Subset Choice for two different parameters. The first parameter is the positive integer p explicitly stated as part of the input to UCG Subset Choice. The second parameter is the implicit parameter q, where q is defined as follows: Let (H, p), with $H = (V, E)$, be an instance for Hypergaph Subset Choice; then $q = p - u(V)$. As we can rewrite $p = u(V) + q$, the criterion q is naturally interpreted as the requirement that the value of the chosen subset A should exceed the value of V by amount q.

First, in Section 6.1, we show that p-UCG Subset Choice is not in FPT (unless FPT = W[1]). In Section 6.2, we explain in more detail how the relational parameter q for Hypergraph Subset Choice relates to the parameter p, by introducing a problem called Hypergraph Subset Rejection. Then, in Section 6.3, we show that q-UCG Subset Choice is in FPT.

6.1. p-UCG subset choice is W[1]-hard

The following theorem shows that p-UCG Subset Choice is not in FPT (unless FPT = W[1]). The proof involves a reduction from the known W[1]-complete problem, k-Independent Set (Downey & Fellows, 1999).

Theorem 3. *p-UCG Subset Choice $\notin$ FPT (unless FPT = W[1]).*

Proof. Reconsider the proof of Lemma 1. Lemma 1 presents a polynomial-time reduction in which we transform any instance (G, k) for Independent Set to an instance (G^*, p) for Graph Subset Choice, with G^* a unit weighted conflict graph and $p = k$. In other words, Lemma 1 presents a parametric reduction from k-Independent Set to p-UCG Subset Choice that runs in polynomial time (and thus also fpt-time). Since, the problem k-Independent Set is known to be W[1]-complete, we conclude that p-UCG Subset Choice is W[1]-hard. □

Since UCG Subset Choice is a special case of Hypergaph Subset Choice we conclude:

Corollary 2. *p-Hypergraph Subset Choice $\notin$ FPT (unless FPT = W[1]).*

Corollary 2 shows that the desire to obtain a satisfactorily large subset value (i.e., we want a value of at least p) is not in itself a crucial source of complexity in Hypergraph Subset Choice.

6.2. Subset rejection and parameter q

This section explains in more detail the parameterization of Hypergraph Subset Choice by $q = p - u(V)$. To facilitate thinking in terms of the parameter q (instead of p), we define a new value function on subsets of vertices in a weighted hypergraph: Let $H = (V, E)$ be a hypergraph and let $B \subseteq V$ be a subset. Then the improvement in value of $A = V \backslash B$, relative to the value of V, is called *rejection value* of B and is defined as

$$\begin{aligned} u_r(B) &= u(V/B) - u(V) \\ &= \left(\sum_{x \in V \backslash B} u(x) + \sum_{e \in \mathrm{E}_H(V \backslash B)} \Delta(e) \right) \\ &\quad - \left(\sum_{x \in V} u(x) + \sum_{e \in \mathrm{E}_H(V)} \Delta(e) \right) \\ &= -1 \cdot \left(\sum_{x \in B} u(x) + \sum_{e \in \mathrm{R}_H(B)} \Delta(e) \right), \end{aligned}$$

where $\mathrm{R}_H(B) = \{(x_1, x_2, \ldots, x_h) \in E | x_1 \text{ or } x_2 \text{ or} \ldots \text{or } x_h \in B\}$ denotes the set of hyperedges incident to at least one vertex in B.

Note that a vertex x has positive rejection-value if its weight *plus* the sum of the weights of its incident hyperedges is negative. In other words, a choice alternative has positive rejection-value if it strongly clashes with other choice alternatives in the set of available alternatives. More generally, a subset $B \subseteq V$ has positive rejection-value if the sum of the values of its elements *plus* the sum of the weights of all hyperedges incident to a vertex $x \in V$ is negative. Thus, removing a rejection set B with positive rejection-value from V entails the removal of negative value from H.

We can now state a new problem, called Hypergraph Subset Rejection:

Hypergraph Subset Rejection (*decision version*)

Input: A *weighted* hypergraph $H = (V, E)$, $E \subseteq \bigcup_{2 \leqslant h \leqslant |V|} V^h$, for every $x \in V$ a weight $u(x) \in \mathbb{Z}$, for every $e \in E$ a weight $\Delta(e) \in \mathbb{Z} \backslash \{0\}$, and a positive integer q.

Question: Does there exist a subset $B \subseteq V$ such that $u_r(B) = -1 \cdot \left(\sum_{x \in B} u(x) + \sum_{e \in \mathrm{R}_H(B)} \Delta(e) \right) \geqslant q$?

While the problem Hypergraph Subset Choice asks for the subset A that we want to *choose*, the problem Hypergraph Subset Rejection asks for the subset $B = V \backslash A$ that we want to *reject* (in the latter case $A = V \backslash B$ is the subset that we want to choose). Note that there exists a subset $B \subseteq V$ with $u_r(B) \geqslant q = p - u(V)$ if and only if there exists a subset $A = V \backslash B$ with $u(A) \geqslant p = q + u(V)$. Thus,

solving Hypergraph Subset Rejection is equivalent to solving Subset Choice.[8]

6.3. q-UCG subset choice is in FPT

We consider the parameterized complexity of q-Hypergraph Subset Choice on unit-weighted conflict graphs, i.e., q-UCG Subset Choice and prove the following theorem.

Theorem 4. *q-UCG Subset Choice* $\in$ FPT.

For simplicity, we work with the version UCG Subset Rejection instead of UCG Subset Choice. That is, we consider the problem as one of deciding whether or not a subset B with $u_r(B) \geqslant p - u(V) = q$ exists; instead of deciding whether or not a subset $A = V \backslash B$ with $u(A) \geqslant p$ exists. Keep in mind, though, that the two conceptualizations are equivalent; i.e., the answer is "yes" for the one if and only if the answer is "yes" for the other.

The proof of Theorem 4 is organized as follows. We start with two observations: The first is a general observation that holds for conflict hypergraphs (Observation 1), the second applies specifically to unit-weighted conflict graphs (Observation 2). Using Observations 1 and 2, we define a branching rule, (B1), that can be used to construct a bounded search tree for q-UCG Subset Rejection, and thus also for q-UCG Subset Choice. Finally, we conclude an fpt-algorithm that solves q-UCG Subset Choice in time $O(2^q|V|)$.

Observation 1 applies to general conflict hypergraphs. It shows that a vertex x with non-positive rejection-value (i.e., its weight plus the weights of its incident edges is positive), in a conflict hypergraph, never needs to be rejected (i.e., always can be chosen). Namely, if $u_r(x) \leqslant 0$, then there always exists a subset B with maximum rejection-value such that $x \notin B$.

Observation 1. *Let conflict hypergraph $H = (V, E)$ and positive integer q form an instance for Conflict Hypergraph (CH) Subset Rejection. Further, let $x \in V$ be a vertex such that $u_r(x) \leqslant 0$. Then (H, q) is a yes-instance for CH Subset Rejection if and only if there exist a subset B with $u_r(B) \geqslant q$ and $x \notin B$.*

Proof. ($\Rightarrow$) Let (H, q) be a yes-instance for CH Subset Rejection. Then there exists a subset $C \subseteq V$ with $u_r(C) \geqslant q$. We show that there exist a subset $B \subseteq C$ with $u_r(B) \geqslant u_r(C) \geqslant q$ and $x \notin B$. We distinguish two cases: (1) Let $x \notin C$. Then $B = C$ proves the claim. (2) Let $x \in C$. Then consider the subset $C \backslash \{x\}$. Since H is a conflict hypergraph (i.e., it has only negative edges and positive vertices) and $u_r(x) \leqslant 0$, we know that $u_r(C \backslash \{x\}) \geqslant u_r(C) \geqslant q$. Then $B = C \backslash \{x\}$ proves the claim. ($\Leftarrow$) Let (H, q) be an instance for CH Subset Rejection and let $B \subseteq V$ with $u_r(B) \geqslant q$ and $x \notin B$. Then (H, q) is a yes-instance for CH Subset Rejection. □

We remark that Observation 1 implies that if a conflict hypergraph $H = (V, E)$ does not contain any vertices of positive rejection-value, then the subset $A \subseteq V$ with $A = \varnothing$ has maximum value $u(A) = 0$, and thus H is a no-instance for any $q > 0$.

Observation 2 applies to unit-weighted conflict graphs. It shows that for every edge (x, y) in a unit-weighted conflict graph we may reject x or y. Namely, in that case, there always exists a subset B with maximum rejection-value with at least one of x or y in B.

Observation 2. *Let $G = (V, E)$ and q form an instance for UCG Subset Rejection and let $(x, y) \in E$. Then (G, q) is a yes-instance for UCG Subset Rejection if and only if there exists $B \subseteq V$ with $u_r(B) \geqslant q$ and $x \in B$ or $y \in B$.*

Proof. ($\Rightarrow$) Let (G, q) be a yes-instance for UCG Subset Rejection and let $(x, y) \in E$. Then there exists a subset $C \subseteq V$ with $u_r(C) \geqslant q$. We show that there exists a subset $B \supseteq C$ with $u_r(B) \geqslant u_r(C) \geqslant q$ such that $x \in B$. We distinguish two cases: (1) Let $x \in C$ or $y \in C$. Then the $B = C$ proves the claim. (2) Let $x, y \notin C$. Then $(x, y) \notin \mathrm{R}_G(C)$. Since $\Delta(x) = 1$ and $\Delta(x, y) = -1$ we know $u_r(C \cup \{x\}) \geqslant u_r(C) + (-\Delta(x) - \Delta(x, y)) = u_r(C) \geqslant q$. Then $B = C \cup \{x\}$ proves the claim. ($\Leftarrow$) Let (G, q) be an instance for UCG Subset Rejection and let $B \subseteq V$ with $u_r(B) \geqslant q$ and $x \in B$ or $y \in B$. Then (G, q) is a yes-instance for UCG Subset Rejection. □

From Observations 1 and 2 we derive the following branching rule (refer to Fig. 3 for an illustration):

(B1) The Positive Endpoint Edge-Branching Rule. *Let s be a search tree node labeled by an instance (G, q), $G = (V, E)$, for UCG Subset Rejection and let $(x_1, x_2) \in E$ with $u_r(x_i) > 0$ for at least one vertex $x_i \in \{x_1, x_2\}$. Then for each $x_i \in \{x_1, x_2\}$ with $u_r(x_i) > 0$ we create a child s_i of s and label it by (G_i, q_i), where $G_i = (V \backslash \{x_i\}, E \backslash \mathrm{R}_G(\{x_i\}))$, $q_i = q - u_r(x_i)$.*

Note that (B1) only applies if there exists an edge (x_1, x_2) in G with $u_r(x_1) > 0$ or $u_r(x_2) > 0$. Thus application of (B1) to a node in the search tree always leads to the

[8] Even though Hypergraph Subset Choice and Hypergraph Subset Rejection ask for a different solution subset (i.e., if $A \subseteq V$ is a solution for the one problem, then $V \backslash A$ is a solution for the other), this difference is insubstantial for present purposes. Namely, the transformation from a subset A to its complement $V \backslash A$ can be done in polynomial-time and thus does not affect the (classical and parameterized) complexity classification of Hypergraph Subset Choice/Hypergraph Subset Rejection. Further, since V is given as part of the input, one can build the sets A and $V \backslash A$ simultaneously by deleting a vertex x from V as soon as x is included in A; the remaining vertices in V together form $V \backslash A$.

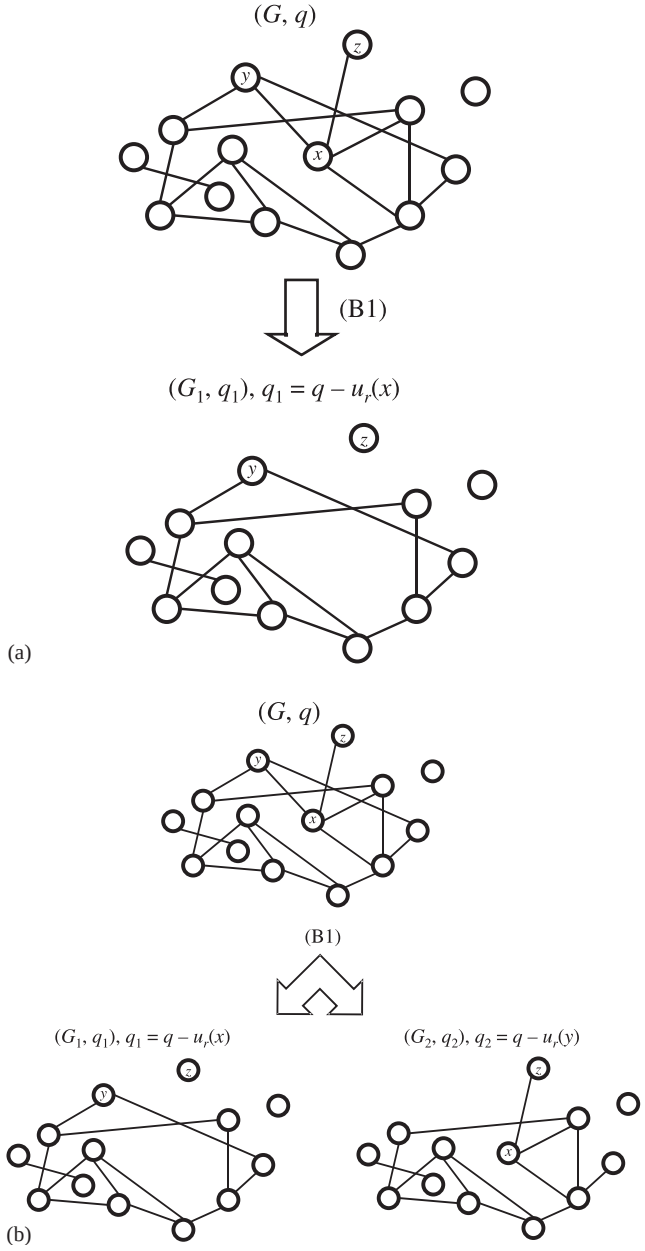

Fig. 3. Illustration of the application of branching rule (B1) to an instance (G, q) for UCG Subset Rejection. Part (a) of the figure illustrates branching on an edge (x, z), where only one endpoint, x, has positive rejection-value, and part (b) of the figure illustrates branching on an edge (x, y) in G where both endpoints x and u have positive rejection-value. In the first case one new instance (G_1, q_1) is created, while in the second case two new instances (G_1, q_1) and (G_2, q_2) are created.

creation of at least one child of that node. If only one of x_1 and x_2 has positive rejection-value then exactly one child is created, and if both x_1 and x_2 have positive rejection-value then exactly two children are created.

To prove that (B1) is a valid branching rule for UCG Subset Rejection, we need to show that (G, q) is a yes-instance for UCG Subset Rejection if and only if at least one of the children of s is labeled by a yes-instance for UCG Subset Rejection.

Proof of (B1). Let (G, q) be a yes-instance for UCG Subset Rejection and let $(x_1, x_2) \in E$ with $u_r(x_i) > 0$ for at least one vertex $x_i \in \{x_1, x_2\}$. We distinguish two cases: (1) Let both x_1 and x_2 have positive rejection-value. Then application of rule (B1) to edge (x_1, x_2) leads to the creation of two instances (G_1, q_1) and (G_2, q_2), where (G_1, q_1) represents the possibility that $x_1 \in B$ and (G_2, q_2) represents the possibility that $x_2 \in B$. From Observation 2 we know that there exists a subset $B \subseteq V$ with maximum rejection-value such that $x_1 \in B$ or $x_2 \in B$. We conclude that (G, q) is a yes-instance for UCG Subset Rejection if and only if (G_1, q_1) or (G_2, q_2) is a yes-instance for UCG Subset Rejection. (2) Let only one of x_1 and x_2 have positive rejection-value. W.l.o.g. let $u_r(x_1) > 0$ and $u_r(x_2) \leqslant 0$. Then application of rule (B1) leads to the creation of only one instance (G_1, q_1) representing the assumption that $x_1 \in B$. From Observation 2 we know that there exists a subset $B \subseteq V$ with maximum rejection-value and $x_1 \in B$ or $x_2 \in B$. Further, from Observation 1 we know that there exists a subset $B \subseteq V$ with maximum rejection-value such that $x_2 \notin B$. We conclude that (G, q) is a yes-instance for UCG Subset Rejection if and only if (G_1, q_1) or (G_2, q_2) is a yes-instance for UCG Subset Rejection. □

With the following lemma we conclude an fpt-algorithm for q-UCG Subset Rejection that runs in time $O(2^q|V|)$.

Lemma 3. q-UCG *Subset Rejection can be solved in time* $O(2^q|V|)$.

Proof. We describe an fpt-algorithm for q-UCG Subset Rejection. The algorithm takes as input an instance (G, q) and creates a search tree T by recursively applying (B1) to (G, q) until an instance (G_i, q_i) in encountered such that either (1) $q_i \leqslant 0$ (in which case the algorithm returns the answer "yes") or G_i does not contain any vertices with positive-rejection value anymore (in which case we know, from Observation 1, that G_i is a no-instance). If the algorithm halts without returning the answer "yes" then we know that all leaves of the search tree T are labeled by no-instances and we conclude (G, q) is a no-instance. We now prove that the algorithm halts in time $O(2^q|V|)$.

First, to apply (B1) to an instance (G, q), $G = (V, E)$, we need to find a vertex $x \in V$ with $u_r(x) \geqslant 1$. To find such a vertex we need to consider at most $|V|$ vertices. Further, whenever we consider a vertex that has non-positive rejection-value we spend no more than $O(1)$ time to compute its rejection-value. Hence, we can find a vertex with positive rejection-value (or know that none exists) in time $O(|V|)$. If we find a vertex x with $u_r(x) \geqslant 1$,

then we branch on any edge (x, y) incident to x. For each new search tree node s_i that we create we spend at most $O(|V|)$ time to label it by (G_i, q_i). Namely, we spend time $O(\deg_G(x))$ to compute the value $u_r(x)$, and we need time $O(\deg_G(x))$ to delete x and its adjacent edges from G. Thus each node in the search tree can be labeled in time $O(\deg_G(x))$. Since $\deg_G(x) \leqslant |V| - 1$, we conclude that $O(\deg_G(x)) \leqslant O(|V|)$.

Second, observe that each application of (B1) leads to the creation of at most 2 new branches in the search tree, and thus $\text{fan}(T) \leqslant 2$. Further, whenever (B1) creates a node labeled by (G_i, q_i) for a parent labeled by (G, q) then $q_i \leqslant q - 1$. Thus, we have $\text{depth}(T) \leqslant q$. We conclude that $\text{size}(T) \leqslant O(2^q)$. Combined with the time spent per node of the search tree we conclude the algorithm runs in time $O(2^q|V|)$. □

Since, the parameterized problem q-UCG Subset Rejection is equivalent to q-UCG Subset Choice, we also have:

Corollary 3. q-UCG *Subset Choice can be solved in time* $O(2^q|V|)$.

Since, $O(2^q|V|)$ is fpt-time for parameter q, Corollary 3 proves Theorem 4.

We have shown how we can solve q-UCG Subset Choice in fpt-time $O(2^q|V|)$. The arguments we used to derive this result are intended to provide an easy to follow illustration. We remark that, with the use of different techniques and a better running-time analysis, it is possible to derive a much faster fpt-algorithm for q-UCG Subset Choice that runs in time $O(1.151^q + q|V|)$. This improved result follows from an analysis by Stege, van Rooij, Hertel, and Hertel (2002) and is discussed by van Rooij (2003).

7. Generalizing q-UCG subset choice

Theorem 4 shows that if a decision-maker has a value-structure that can be represented by a unit-weighted conflict graph, and s/he aims to choose a subset with a value that is at least q more than $u(V)$, then the task is practically feasible for large $|V|$ as long as q is not too large. In this section, we study to what extent this result generalizes to value-structures that form generalizations of the unit-weighted conflict graph. Specifically, we consider Subset Choice on edge-weighted conflict graphs (ECG Subset Choice), vertex-weighted conflict graphs (VCG Subset Choice), conflict graphs (CG Subset Choice), and conflict hypergraphs (CH Subset Choice) (see Table 3). For problems Π and Π', let $\Pi' \subseteq \Pi$ denote that Π' is a special case of Π. Then we have UCG Subset Choice $\subseteq$ ECG Subset Choice $\subseteq$ CG Subset Choice $\subseteq$ CH Subset Choice; and also UCG Subset Choice $\subseteq$ VCG Subset Choice $\subseteq$ CG Subset Choice $\subseteq$ CH Subset Choice.

The investigation in Sections 7.1–7.4 takes the following form. Each subsection considers one of the aforementioned problems. For each considered problem Π we ask: Is q sufficient to capture the non-polynomial complexity inherent in Π? If the answer is "no," we attempt to find a superset $\kappa \supseteq q$, such that κ-$\Pi \in$ FPT. In Section 8, we will review to what extent the analyses have led to the identification of crucial sources of complexity as defined in Section 5.3.

7.1. q-ECG subset choice is in FPT

Here we show that q-ECG Subset Choice is in FPT. First note that Observation 1 above also applies to ECG Subset Choice (viz., edge-weighted conflict graphs are a special type of conflict hypergraphs). Further, Observation 2 for unit-weighted conflict graphs directly generalizes for edge-weighted conflict graphs, as shown in the next observation.

Observation 3. *Let $G = (V, E)$ and q form an instance for ECG Subset Rejection and let $(x, y) \in E$. Then (G, q) is a yes-instance for ECG Subset Rejection if and only if there exists $B \subseteq V$ with $u_r(B) \geqslant q$ and $x \in B$ or $y \in B$.*

Proof. Analogous to the proof of Observation 2, with G being an edge-weighted conflict graphs instead of a unit-weighted conflict graph. □

From Observations 1 and 3 we conclude that the algorithm described for q-UCG Subset Choice also solves q-ECG Subset Choice in time $O(2^q|V|)$.

Corollary 4. q-ECG *Subset Choice* $\in$ FPT.

Corollary 4 shows that the presence of edge-weights in a conflict graph, in itself, does not add non-polynomial time complexity to subset choice on conflict-graphs over and above the non-polynomial time complexity already captured by q.

7.2. q-VCG subset choice is W[1]-hard

We next show that, although q is sufficient for capturing the non-polynomial time complexity in ECG Subset Choice, the same is not true for VCG Subset Choice (unless FPT = W[1]).

Theorem 5. q-VCG *Subset Choice* $\notin$ FPT (*unless* FPT = W[1]).

To prove Theorem 5, we present a parametric reduction from k-Independent Set to q-VCG Subset Choice in Lemma 4 (see Fig. 4 for an illustration). For simplicity, in this reduction we assume that the input graph for k-Independent Set is of minimum degree 1

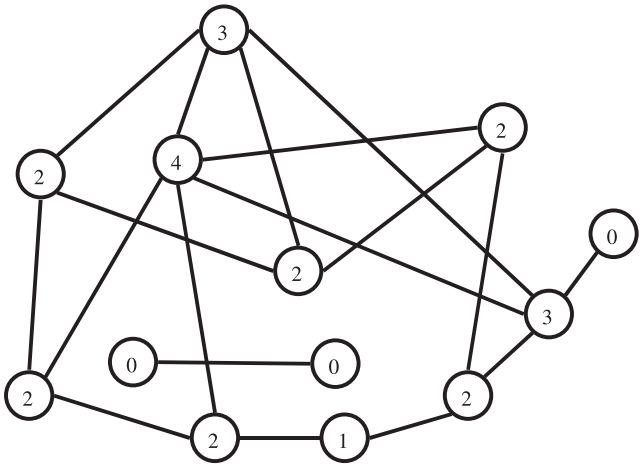

Fig. 4. Illustration of the reduction in Lemma 4. The reduction transforms G into G^*. The unweighted version of the figure represents graph G, and the weighted version of the figure represents the graph G^*. Only the vertex weights are shown for G^*; since G^* is a vertex-weighted conflict graph, all edge weights are set '−1.' Note that $u(x) = \deg_G(x) - 1$ for each vertex x in G^*. This property ensures that G has an independent set of size k if and only if there exists a subset $B \subseteq V$, with $u_r(B) \geqslant q = k$.

(i.e., G contains no singletons). Since k-Independent Set (with or without singletons) is known to be W[1]-complete (Downey & Fellows, 1999) the reduction shows that q-VCG Subset Choice is W[1]-hard.

Lemma 4. *Let (G, k), $G = (V, E)$, be an instance for k-Independent Set, where G contains no singletons. We build an instance (G^*, q), with $G^* = (V^*, E^*)$, for q-VCG Subset Choice as follows. G^* has the same edges and vertices as G but is a vertex-weighted conflict graph. Therefore let $V^* = V$ and $E^* = E$. We define the weights for G^* as follows: for every $x \in V^*$ let $u_r(x) = \deg_{G^*}(x) - 1$ and for every $e \in E^*$ let $\Delta(e) = -1$. Note that $u_r(x) = 1$ for all $x \in V^*$. Furthermore let $q = k$. Then G has an independent set of size at least k if and only if there exists $B \subseteq V^*$ with $u_r(B) \geqslant k$.*

Proof. ($\Rightarrow$) Let $B \subseteq V$, $B = \{x_1, x_2, \ldots, x_k\}$, be an independent set for G. This means that no two vertices $x_i, x_j \in \{x_1, x_2, \ldots, x_k\}$ share an edge in G and since $E = E^*$ also no two vertices $x_i, x_j \in \{x_1, x_2, \ldots, x_k\}$ share an edge in G^*. Thus $u_r(B) = u_r(x_1) + u_r(x_2) + \cdots + u_r(x_k) = k$. ($\Leftarrow$) Let $B \subseteq V^*$ with $u_r(B) \geqslant q$. We show G has an independent set of size at least q. We distinguish two cases: (1) If B is an independent set for G^* then B is an independent set for G. Assume $B = \{x_1, x_2, \ldots, x_k\}$. Then $u_r(B) = u_r(x_1) + u_r(x_2) + \cdots + u_r(x_k) \geqslant q$, and thus $k \geqslant q$. (2) If B is not an independent set for G^*, we transform B into an independent set B' for G^* with the algorithm described in the proof of Lemma 1. Note that line 4 of the algorithm always results in the removal of at most $\deg^*_G(x) - 1$ edges from $\mathrm{R}_{G^*}(B')$. Hence, in line 4, $u_r(B'\{x\}) \geqslant u_r(B') - u(x) - (\deg_{G^*}(x) - 1) = u_r(B') - (\deg_{G^*}(x) - 1) + (\deg_{G^*}(x) - 1) = u_r(B')$, and thus in line 6, $u_r(B') \geqslant q$. Furthermore, when the algorithm halts B' is an independent set for G^* and therefore for G. Thus case (1) applies to $B = B'$. □

Because the above reduction runs in polynomial-time, Lemma 4 also constitutes an alternative proof of Theorem 1. Further, since VCG Subset Choice is a special case of CG Subset Choice, which in turn is a special case of CH Subset Choice, we also conclude:

Corollary 5. q-CG *Subset Choice* $\notin$ FPT (*unless* FPT = W[1]).

Corollary 6. q-CH *Subset Choice* $\notin$ FPT (*unless* FPT = W[1]).

7.3. $\{q, u_{\max}\}$-CG subset choice is in FPT

We consider another parameter for Hypergraph Subset Choice: The maximum vertex weight, denoted by $u_{\max}$. Note that for every instance (H, q), with $H = (V, E)$, for Hypergraph Subset Choice, there exists a positive integer value $u_{\max}$ such that for all $x \in V$, $u(x) \leqslant u_{\max}$. Thus, $u_{\max}$ is an implicit parameter for Hypergraph Subset Choice. In the following we show that, although q-CG Subset Choice is W[1]-hard (Corollary 5), the parameter set $\{q, u_{\max}\}$ is sufficient for capturing the non-polynomial time complexity inherent in CG Subset Choice.

Theorem 6. $\{q, u_{\max}\}$-CG *Subset Choice* $\in$ FPT.

The proof of Theorem 6 is organized as follows. First we observe that for every vertex in a conflict graph with positive rejection-value, we can always either reject that vertex or reject at least one of its neighbors (Observation 4 below). Using Observations 1 and 4, we define a branching rule (B2) that can be used to construct an fpt-algorithm for $\{q, \delta_{\max}\}$-CG Subset Choice (as before, $\delta_{\max}$ denotes the maximum vertex degree). Then we show that there exists a function $f(u_{\max}, q)$ such that $\delta_{\max} \leqslant f(u_{\max}, q)$. This allows us to conclude the existence of an fpt-algorithm for $\{q, u_{\max}\}$-CG Subset Choice.

Observation 4 shows that, for an instance (G, q) for CG Subset Choice and a vertex x with positive rejection-value, if a subset $B \subseteq V$ has maximum rejection-value then there exist a vertex y in the open neighborhood of x such that $y \in B$.

Observation 4. *Let G and q form an instance for CG Subset Choice and let $x \in V$ with $u_r(x) > 0$. Then G and q form a yes-instance for CG Subset Choice if and only if there exists $B \subseteq V$ with $u_r(B) \geqslant q$, and at least one vertex $y \in \mathrm{N}_G[x]$, with $y \in B$.*

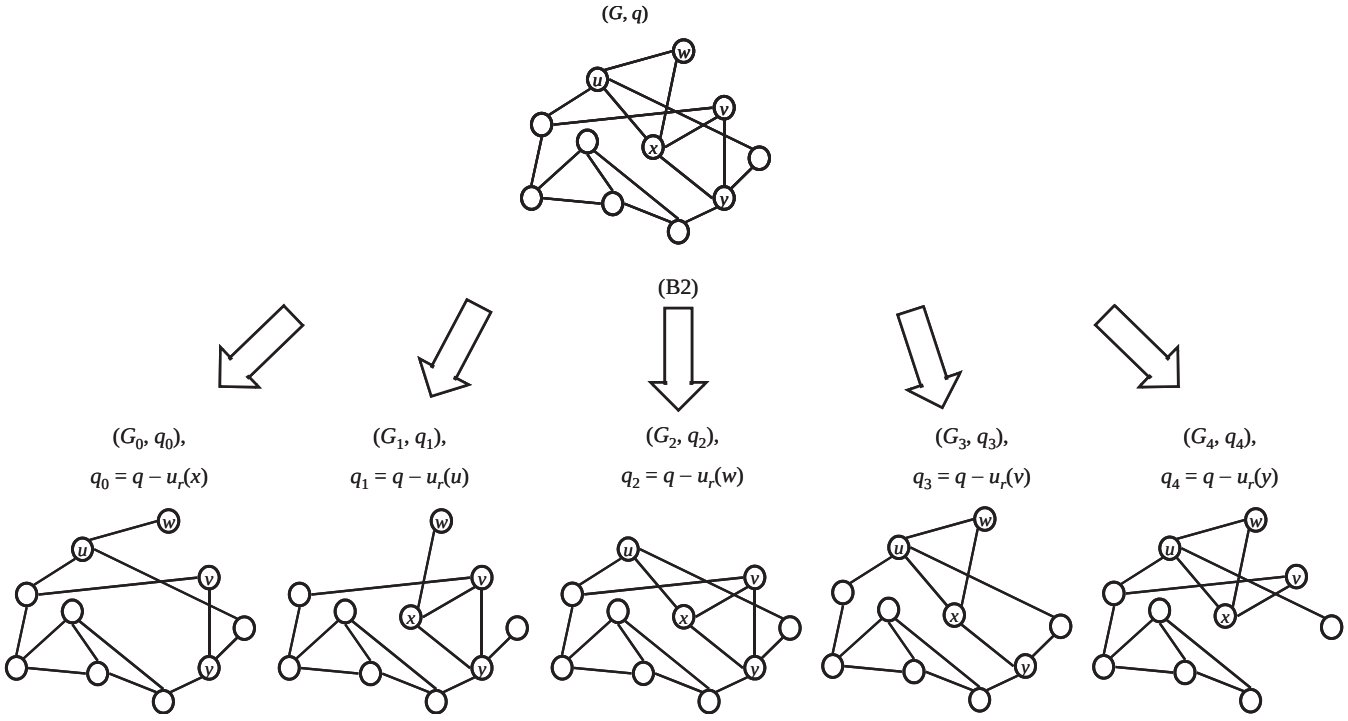

Fig. 5. Illustration of the application of branching rule (B2) to an instance (G, q) for CG Subset Rejection. The instances obtained after branching rule application on vertex x in G, are denoted (G_i, q_i) with $i = 0, 1, 2, \ldots, \deg_G(x)$. For clarity, the vertex and edge weights are not depicted; but the reader may assume that, in this example, each of the vertices u, v, w, x, and y, has positive rejection-value in G. In this example, $\deg_G(x) = 4$, thus (at most) five new instances are created by (B2).

Proof. ($\Rightarrow$) Let (G, q) be a yes-instance for CG Subset Choice and let $x \in V$ with $u_r(x) > 0$. Then there exists $C \subseteq V$ with $u_r(C) \geqslant q$. We show there exists $B \subseteq V$ such that $u_r(B) \geqslant u_r(C) \geqslant q$ and at least one vertex $y \in \mathrm{N}_G[x]$, with $y \in B$. We distinguish two cases: (1) There exists a vertex $y \in \mathrm{N}_G[x]$ with $y \in C$. Then $B = C$ proves the claim. (2) There does not exist $y \in \mathrm{N}_G[x]$ with $y \in C$. Then $x \notin C$, since $x \in \mathrm{N}_G[x]$. Further, since $u_r(x) \geqslant 1$ we know that $u_r(C \cup \{x\}) \geqslant u_r(C) + u_r(x) \geqslant u_r(C) + 1 > q$, and thus $B = C \cup \{x\}$ proves the claim. ($\Leftarrow$) Let G and q form an instance for CG Subset Choice and let $B \subseteq V$ be any subset with $u_r(B) \geqslant q$. Then (G, q) is a yes-instance for CG Subset Choice. □

The following rule (B2) uses Observations 1 and 4 to branch on vertices with positive rejection-value in a vertex-weighted conflict graph (refer to Fig. 5 for an illustration).

(B2) The Positive Vertex-or-At-Least-One-Neighbor Branching Rule #1. *Let s be a search tree node labeled by an instance (G, q), $G = (V, E)$ for CG Subset Choice and let $x \in V$, with $u_r(x) > 0$ and $\mathrm{N}_G(x) = \{x_1, \ldots, x_k\}, k \leqslant \delta_{\max}$. Then for each $x_i \in \mathrm{N}_G[x]$ with $u_r(x_i) > 0$ we create a child s_i of s and label it by (G_i, q_i), $G_i = (V \backslash \{x_i\}, E \backslash \mathrm{R}_G(\{x_i\}))$, $q = q - u_r(x_i)$.*

Proof. Let (G, q) be a yes-instance for CG Subset Rejection and let $x \in V$, with $u_r(x) > 0$, $\mathrm{N}_G(x) = \{x_1, \ldots, x_k\}$. Application of (B2) results in the creation of an instance (G_i, q_i) for each $x_i \in \mathrm{N}_G[x]$ with $u_r(x_i) > 0$. Here, each instance (G_i, q_i) represents the assumption that $x_i \in B$. From Observation 4 we know that there exists a subset $B \subseteq V$ with maximum rejection-value and $B \cap \{x_i \in \mathrm{N}_G[x] : u_r(x_i) > 0\} \neq \varnothing$. Further, from Observation 1 we know that there exists a subset $B \subseteq V$ with maximum rejection-value and $x_i \notin B$ for all $x_i \in \mathrm{N}_G[x]$ with $u_r(x_i) \leqslant 0$. We conclude (G, q) is a yes-instance for CG Subset Rejection if and only if at least one of (G_i, q_i) is a yes-instance for CG Subset Rejection. □

Application of (B2) to a search tree node s leads to the creation of at most $\deg_G(x) + 1 \leqslant \delta_{\max} + 1$ children of s. Further, for each newly created instance (G_i, q_i), we have $|G_i| \leqslant |G|$, $\delta_{\max_i} \leqslant \delta_{\max}$, $q_i \leqslant q - 1$. Thus we can use (B2) to build a search tree with $\mathrm{fan}(T) \leqslant \delta_{\max} + 1$ and $\mathrm{depth}(T) \leqslant q$.

Lemma 5. *$\{\delta_{\max}, q\}$-CG Subset Rejection can be solved in time $O((\delta_{\max} + 1)^q |V|)$.*

Proof. Analogous to the proof of Lemma 3, we define an fpt-algorithm for $\{\delta_{\max}, q\}$-CG Subset Rejection: The algorithm takes as input an instance (G, q) and recursively applies (B2) to some vertex x in G with $u_r(x) > 0$ until either a "yes"-answer is returned or (B2) cannot be applied anymore. If the algorithm halts

without returning the answer "yes" then we know that each leaf s_i in the search tree T is labeled by an instance (G_i, q_i), such that all vertices in G_i have non-positive rejection-value. Then, from Observation 1, we can conclude that (G, q) is a no-instance.

We next prove that the algorithm halts in time $O((\delta_{\max}+1)^q|V|)$. To find a vertex x to branch on (or know that none exists), and label a new node in the search tree, we need at most time $O(|V|)$. Since each application of (B2) leads to the creation of at most $\delta_{\max}+1$ new branches in the search tree, we know that $\text{fan}(T)\leqslant\delta_{\max}+1$. Further, whenever (B2) creates a node labeled by (G_i, q_i), for a parent labeled (G, q), then $q_i\leqslant q-1$, and thus $\text{depth}(T)\leqslant q$. We conclude that $\text{size}(T)\leqslant O((\delta_{\max}+1)^q)$. Combined with the time spent per node of the search tree we conclude the algorithm runs in time $O((\delta_{\max}+1)^q|V|)$. □

The following lemma shows that for any instance (G, q) for CG Subset Rejection we can bound the vertex degree by a function $f(u_{\max}, q)$.

Lemma 6. *Let (G, q) be an instance for CG Subset Rejection and let $u_{\max}$ be the maximum vertex weight in G. If there exists a vertex $x\in V$ with $\deg_G(x)\geqslant q+u_{\max}$ then (G, q) is a yes-instance.*

Proof. We know for every $x\in V$, $u_r(x)=-(u(x)+\sum_{e\in R_G(\{x\})}\Delta(e))\geqslant -u_{\max}+\deg_G(x)$. Let $x\in V$ be a vertex with $\deg_G(v)\geqslant q+u_{\max}$. Then $u_r(x)\geqslant q$ and thus (G, q) is a yes-instance for CG Subset Rejection. □

From Lemma 6 we conclude a refinement of the search tree algorithm described in Lemma 5: As soon as we encounter a node labeled by (G, q), such that $\delta_{\max}\geqslant q+u_{\max}$, we terminate the search and return the answer "yes." This way we ensure that for the resulting bounded search tree T, $\text{fan}(T)\leqslant\delta_{\max}+1\leqslant q+u_{\max}$.

Corollary 7. *$\{q, u_{\max}\}$-CG Subset Choice can be solved in time $O((q+u_{\max})^q|V|)$.*

Since time $O((q+u_{\max})^q|V|)$ is fpt-time for $\{q, u_{\max}\}$-CG Subset Choice, Corollary 7 proves Theorem 6. Note that, since VCG Subset Choice is a special case of CG Subset Choice, all results discussed above for CG Subset Choice also apply to VCG Subset Choice.

7.4. $\{q, u_{\max}, \varepsilon\}$-CH subset choice is in FPT

In this section we consider Subset Choice on general conflict hypergraphs and we prove the following theorem.

Theorem 7. *$\{q, \varepsilon, u_{\max}\}$-CH Subset Choice* $\in$ FPT.

Recall that ε denotes the maximum span in a hypergraph. Since, for every hypergraph $H=(V, E)$, there exists a positive integer ε such that $\text{span}(e)\leqslant\varepsilon$ for every $e\in E$, the integer ε is an implicit parameter for CH Subset Choice.

The proof of Theorem 7 is organized as follows. First, we observe that Observation 4 for conflict graphs directly generalizes to conflict hypergraphs (Observation 5). Then we consider a new input parameter $N_{\max}$, denoting the maximum neighborhood of a vertex in H (i.e., $N_{\max}$ is the smallest positive integer, such that for every x in H, $|N_H(x)|\leqslant N_{\max}$). Using Observations 1 and 5, we derive a branching rule (B3) that can be used to construct an fpt-algorithm for $\{q, N_{\max}\}$-CH Subset Rejection. We will show that there exists a function $f(\varepsilon, \delta_{\max})$ with $N_{\max}\leqslant f(\varepsilon, \delta_{\max})$. This allows us conclude an fpt-algorithm for $\{q, \varepsilon, \delta_{\max}\}$-CH Subset Rejection. Finally, we show that there exists a function $g(u_{\max}, q)$, with $\delta_{\max}\leqslant g(u_{\max}, q)$, and conclude an fpt-algorithm for $\{q, \varepsilon, u_{\max}\}$-CH Subset Choice.

Observation 5. *Let H and q form an instance for CH Subset Choice and let $x\in V$ with $u_r(x)>0$. Then H and q form a yes-instance for CH-Subset Choice if and only if there exists $B\subseteq V$, with $u_r(B)\geqslant q$, and at least one vertex $y\in N_H[x]$, with $y\in B$.*

Proof. Analogous to the proof of Observation 4, using as instance (H, q), such that H is a conflict hypergraph instead of conflict graph. □

From Observations 1 and 5 we conclude a branching rule (B3) for CH Subset Choice that allows us to construct a bounded search tree T, with $\text{size}(T)\leqslant f(q, N_{\max})$. Note that (B3) is identical to (B2) with the exception that it takes as input a conflict hypergraph instead of a conflict graph.

(B3) The Positive Vertex-or-At-Least-One-Neighbor Branching Rule #2. *Let s be a search tree node labeled by an instance (H, q), $H=(V, E)$ for CH Subset Rejection and let $x\in V$, with $u_r(x)>0$ and $N_H(x)=\{x_1, \ldots, x_k\}$, $k\leqslant N_{\max}$. Then for each $x_i\in\{x, x_1, \ldots, x_k\}$ with $u_r(x_i)>0$ we create a child s_i of s and label it by (H_i, q_i), $H_i=(V\backslash\{x_i\}, E\backslash R_H(\{x_i\}))$, $q_i=q-u_r(x_i)$.*

Proof. Analogous to the proof of (B2), using as instance (H, q), such that H is a conflict hypergraph instead of a conflict graph, and using Observation 5 instead of Observation 4. □

We now show how (B3) can be used to define an fpt-algorithm for $\{q, N_{\max}\}$-CH Subset Rejection.

Lemma 7. *$\{q, N_{\max}\}$-CH Subset Rejection can be solved in time $O((N_{\max}+1)^q|V|^2)$.*

Proof. Analogous to the proofs of Lemmas 3 and 5, we define an fpt-algorithm for $\{q, \mathrm{N}_{\max}\}$-CH Subset Rejection. The algorithm takes as input an instance (H, q) and recursively applies (B3) to a vertex x in H until either a "yes"-answer is returned or (B3) cannot be applied anymore (in which case, by Observation 1, we return the answer "no"). We can find a vertex x to branch on (or know that none exists), and label a new node in the search tree, in time $O(|V|^2)$ (the labeling can no longer be done in linear time, because a vertex in a hypergraph may have as many as $((|V|-1)(|V|-2))/2$ incident hyperedges). Since (B3) creates a bounded search tree T with $\mathrm{fan}(T) \leqslant \mathrm{N}_{\max}+1$ and $\mathrm{depth}(T) \leqslant q$, we conclude that $\mathrm{size}(T) \leqslant O((\mathrm{N}_{\max}+1)^q)$. In sum, we can decide $\{q, \mathrm{N}_{\max}\}$-CH Subset Rejection in time $O((\mathrm{N}_{\max}+1)^q |V|^2)$. □

Note that unlike $\mathrm{N}_G(x)$, $\mathrm{N}_H(x)$ may be larger than $\deg_H(x)$, and thus $\mathrm{N}_{\max}$ is not bounded by a function $f(\delta_{\max})$ in hypergraphs. Since $\mathrm{span}_H(e) \leqslant \varepsilon$ for all $e \in E$, we do know that, for every $x \in V$, $|\mathrm{N}_H(x)| \leqslant (\varepsilon-1)\deg_H(x)$, and thus $\mathrm{N}_{\max} \leqslant (\varepsilon-1)\delta_{\max}$. This observation allows us to conclude the following corollary.

Corollary 8. $\{q, \varepsilon, \delta_{\max}\}$-CH *Subset Rejection can be solved in time* $O(((\varepsilon-1)\delta_{\max}+1)^q |V|^2)$.

To show that $\delta_{\max}$ is bounded by some function $\mathrm{g}(u_{\max}, q)$, we observe that Lemma 6 for conflict graphs generalizes directly to Lemma 8 for conflict hypergraphs.

Lemma 8. *Let* (H, q), *with* $H = (V, E)$, *be an instance for CH Subset Rejection and let* $u_{\max}$ *be the maximum vertex weight in H. If there exists a vertex* $x \in V$ *with* $\deg_H(x) \geqslant q + u_{\max}$, *then* (H, q) *is a yes-instance.*

Proof. We know for every $x \in V$, $u_r(x) = -1 \cdot (u(x) + \sum_{e \in \mathrm{R}_H(\{x\})} \Delta(e)) \geqslant -u_{\max} + \deg_H(x)$. Let $x \in V$ be a vertex with $\deg_H(x) \geqslant q + u_{\max}$. Then $u_r(x) \geqslant q$ and thus (H, q) is a yes-instance for CH Subset Rejection. □

Using Lemma 8 we can refine the algorithm in Lemma 7: We terminate the search as soon as we encounter an instance (G, q) with $\delta_{\max} \geqslant q + u_{\max}$ and return the answer "yes." This ensures that $\mathrm{fan}(T) \leqslant q + u_{\max} - 1$. We conclude the following corollary:

Corollary 9. $\{q, \varepsilon, u_{\max}\}$-CH *Subset Choice can be solved in time* $O(((\varepsilon-1)(q+u_{\max}-1)+1)^q |V|^2)$.

Since time $O(((\varepsilon-1)(q+u_{\max}-1)+1)^q |V|^2)$ is fpt-time for $\{q, \varepsilon, u_{\max}\}$-CH Subset Choice, Corollary 9 proves Theorem 7.

8. Crucial sources of complexity

In this section we reconsider some of the results obtained in the previous sections, with the aim of identifying crucial sources of complexity in Subset Choice for value-structures that are represented by conflict hypergraphs. We start by reconsidering the result that $\{q, \varepsilon, u_{\max}\}$-CH Subset Choice $\in$ FPT (Corollary 9). Is the parameter set $\{q, \varepsilon, u_{\max}\}$ a crucial source of complexity for CH Subset Choice? Recall from Section 5.3 that we can conclude that $\{q, \varepsilon, u_{\max}\}$ is a crucial source of complexity for CH Subset Choice if and only if κ-CH Subset Choice $\notin$ FPT for every $\kappa \subset \{q, \varepsilon, u_{\max}\}$. In other words, we need to know if $\{q, \varepsilon, u_{\max}\}$ is a *minimal* parameter set for CH Subset Choice. What do we know about the parameterized complexity of κ-CH Subset Choice for different $\kappa \subset \{q, \varepsilon, u_{\max}\}$? To answer this question, in the following we first reconsider Theorem 3 and then Theorem 5.

Theorem 3 states that p-UCG Subset Choice is not in FPT (unless FPT = W[1]). We next show how this theorem implies the fixed-parameter intractability of κ-Hypergraph Subset Choice for the parameter set $\kappa = \{p, \varepsilon, u_{\min}, u_{\max}, \Delta_{\min}, \Delta_{\max}\}$. Here, p, ε and $u_{\max}$ are defined as before (i.e., the positive integer in the input, the maximum span, and the maximum vertex weight), and the new parameters $u_{\min}$, $\Delta_{\min}$, $\Delta_{\max}$ are defined as follows: Let $u_{\min} \geqslant 0$, $\Delta_{\min} \geqslant 1$, $\Delta_{\max} \geqslant 1$, be the smallest integers that for all $x \in V$, $-u_{\min} \leqslant u(x) \leqslant u_{\max}$ and for all $e \in E$, $-\Delta_{\min} \leqslant \Delta(e) \leqslant \Delta_{\max}$.

For UCG Subset Choice, the input hypergraph is a unit-weighted conflict graph. In other words, in UCG Subset Choice, the values ε, $u_{\min}$, $u_{\max}$, $\Delta_{\min}$, $\Delta_{\max}$ are all constants; specifically we have $\varepsilon = 2$, $u_{\min} = 0$, $u_{\max} = 1$, $\Delta_{\min} = 1$, and $\Delta_{\max} = 1$. This means that we can conclude the following corollary from Theorem 3.

Corollary 10. $\{p, \varepsilon, u_{\min}, u_{\max}, \Delta_{\min}, \Delta_{\max}\}$-CH *Subset Choice* $\notin$ FPT (*unless* FPT = W[1]).

Proof. The proof is by contraction. Assume $\{p, \varepsilon, u_{\min}, u_{\max}, \Delta_{\min}, \Delta_{\max}\}$-CH Subset Choice $\in$ FPT and FPT $\neq$ W[1]. Then there exists an algorithm solving Subset Choice in time $O(f(p, \varepsilon, u_{\min}, u_{\max}, \Delta_{\min}, \Delta_{\max})n^\alpha)$. In UCG Subset Choice we have $\varepsilon = 2$, $u_{\min} = 0$, $u_{\max} = 1$, $\Delta_{\min} = 1$, and $\Delta_{\max} = 1$, and thus we can solve UCG Subset Choice in time $O(f(p, \varepsilon, u_{\min}, u_{\max}, \Delta_{\min}, \Delta_{\max})n^\alpha) = O(f(p, 2, 0, 1, 1, 1)n^\alpha) = O(f(p)n^\alpha)$. This means that p-UCG Subset Choice $\in$ FPT. But then, from Theorem 3, we can conclude that FPT = W[1]. □

Recall from Section 5.3 that for a problem Π and parameter set κ, if κ-$\Pi \in$ FPT then κ'-$\Pi \in$ FPT for all $\kappa' \supseteq \kappa$. Thus, Corollary 10 implies that κ-Subset Choice $\notin$ FPT, for all $\kappa \subseteq \{p, \varepsilon, u_{\min}, u_{\max}, \Delta_{\min}, \Delta_{\max}\}$

(unless FPT = W[1]). In other words, we can conclude from Corollary 10 that κ-CH Subset Choice $\notin$ FPT, for $\kappa = \{\varepsilon\}$, $\kappa = \{u_{\max}\}$, and $\kappa = \{\varepsilon, u_{\max}\}$ (unless FPT = W[1]).

We now reconsider Theorem 5. This theorem states that q-VCG Subset Choice is not in FPT (unless FPT = W[1]). Since VCG Subset Choice is a special case of CH Subset Choice, with $\varepsilon = 2$, $u_{\min} = 0$, $\Delta_{\min} = 1$, $\Delta_{\max} = 1$, we conclude the following corollary.

Corollary 11. $\{q, \varepsilon, u_{\min}, \Delta_{\min}, \Delta_{\max}\}$-UC *Subset Choice* $\notin$ FPT (*unless* FPT = W[1]).

Proof. Analogous to the proof of Corollary 10. □

From Corollary 11, we conclude that κ-CH Subset Choice $\notin$ FPT, for $\kappa = \{q\}$, and $\kappa = \{q, \varepsilon\}$ (unless FPT = W[1]).

In sum, from the above analyses, we know that κ-CH Subset Choice is not in FPT (unless FPT = W[1]) for five of the six possible $\kappa \subset \{q, \varepsilon, u_{\max}\}$. The only proper subset that was not considered is $\kappa = \{q, u_{\max}\}$. It remains an open question whether or not $\{q, u_{\max}\}$-CH Subset Choice is in FPT. Hence, at present time, we do not know whether $\{q, \varepsilon, u_{\max}\}$ or $\{q, u_{\max}\}$ is a crucial source of complexity in CH Subset Choice.

Despite the open question posed above, we note that we did succeed in identifying crucial sources of complexity for some special cases of CH Subset Choice. Namely, from Theorem 4 we know that q-ECG Subset Choice $\in$ FPT. Since $\{q\}$ contains only one element it is minimal, and thus we know that $\{q\}$ is a crucial source of complexity for ECG Subset Choice (and thus also for UCG Subset Choice). Further, from Corollary 5 we know q-CG Subset Choice $\notin$ FPT (unless FPT = W[1]), from Corollary 10 we know $\{u_{\max}\}$-CG Subset Choice $\notin$ FPT (unless FPT = W[1]), and from Theorem 6 we know $\{q, u_{\max}\}$-CG Subset Choice $\in$ FPT. We conclude that $\{q, u_{\max}\}$ is a crucial source of complexity in CG Subset Choice (and thus also for VCG Subset Choice).

9. Subset choice when subset size matters

Throughout this article we have assumed no restrictions on the *size* of the chosen subset. Clearly, in many real-world settings size restrictions do apply; e.g., there may be an exact bound (as when you have exactly k job positions to fill), an upper-bound (as when you can afford at most k toppings on your pizza), or a lower-bound (as when you require at least k members on a committee). In this section we offer a word of caution: One cannot assume that the results for subset choice without size restrictions automatically generalize to subset choice with size restrictions. To illustrate, we consider the problem Exact-bound Subset Choice:

Exact-bound Subset Choice

Input: A weighted hypergraph $H = (V, E)$, $\bigcup_{2 \leqslant h \leqslant |V|} V^h$, for every $x \in V$ a weight $u(x) \in \mathbb{Z}$, for every $e \in E$ a weight $\Delta(e) \in \mathbb{Z} \backslash \{0\}$. Positive integers p and k.
Question: Does there exist a subset $A \subseteq V$ such that $u(A) \geqslant p$ and $|A| = k$?

Recall that USG Subset Choice *without* size restrictions is in P (Theorem 2). In contrast, we have the following theorem for Exact-bound Subset Choice on unit-weighted surplus graphs.

Theorem 8. USG *Exact-bound Subset Choice is* NP-*hard.*

To prove Theorem 8 we reduce from the NP-complete problem Clique (Garey & Johnson, 1979). For a graph $G = (V, E)$ and a subset $A \subseteq V$, we say A is a *clique* for G if for every two vertices $x, y \in A$, $(x, y) \in E$.

Clique (decision version)

Input: A graph $G = (V, E)$ and a positive integer k.
Question: Does there exist a clique $A \subseteq V$ for G with $|A| \geqslant k$?

For a graph $G = (V, E)$ and a positive integer k, we make two observations. First, if A is a clique for G then any subset $A' \subseteq A$ is also a clique for G. We conclude Observation 6.

Observation 6. *If G and k form a yes-instance for Clique then there exists $A \subseteq V$ such that A is a clique and $|A| = k$.*

Further, for any set $A \subseteq V$, with $|A| = k$, the number of possible pairs of vertices is given by $\frac{k(k-1)}{2}$. We conclude Observation 7.

Observation 7. *Let $A \subseteq V$ with $|A| = k$. Then A is a clique if and only if $|\mathrm{E}_G(A)| = \frac{k(k-1)}{2}$.*

Using Observations 6 and 7, we now prove Theorem 8.

Proof of Theorem 8. Let graph $G = (V, E)$ and positive integer k constitute an instance for Clique. From G and k we build an instance for USG Exact-bound Subset Choice consisting of G^*, p and k as follows: let $G^* = (V^*, E^*)$ be a weighted graph such that $V^* = V$ and $E^* = E$. For every $x \in V^*$ let $u(x) = -1$ and for every $e \in E^*$ let $\Delta(e) = +1$ (i.e., G^* is a unit-weighted surplus graph). Further, let $p = \frac{k(k-1)}{2} - k$. This transformation can be done in time $O(|V|^2)$. It remains to be shown that G and k form a yes-instance for Clique if and only if G^*, p and k form a yes-instance for Exact-bound Subset

Choice. Because Clique is NP-hard even for $k \geqslant 5$ (Garey & Johnson, 1979), in the following we can assume that $k \geqslant 5$.

($\Rightarrow$) Let G and k form a yes-instance of Clique. Then, from Observation 6, we know there exists $A \subseteq V$, such that A is a clique and $|A| = k$. Since $u(A) = \sum_{x \in A} u(x) + \sum_{e \in \mathrm{E}_{G^*}(A)} \Delta(e) = |\mathrm{E}_{G^*}(A)| - |A| = \frac{k(k-1)}{2} - k = p$, we conclude that G^*, p and k form a yes-instance for USG Exact Bound Subset Choice. ($\Leftarrow$) Let G^*, p and k form a yes-instance for USG Exact Bound Subset Choice. Then there exists $A \subseteq V = V^*$ with $|A| = k$ and $u(A) = |\mathrm{E}_{G^*}(A)| - |A| \geqslant p = \frac{k(k-1)}{2} - k$. But that means that $|\mathrm{E}_G(A)| = |\mathrm{E}_{G^*}(A)| = \frac{k(k-1)}{2}$ and thus, from Observation 7, we conclude that A is a clique for G of size k. □

Note that the polynomial-time reduction in the proof above also happens to be a parametric reduction from k-Clique to $\{k,p\}$-USG Exact Bound Subset Choice, with $p = \frac{k(k-1)}{2} - k$. Since k-Clique is known be W[1]-complete (Downey & Fellows, 1999), the reduction establishes that $\{k,p\}$-USG Exact Bound Subset Choice is W[1]-hard.

Corollary 12. $\{k,p\}$-USG *Exact Bound Subset Choice* $\notin$ FPT (*unless* FPT = W[1]).

Theorems 2 and 8 illustrate that classical complexity results do not automatically generalize to subset choice under subset-size restrictions. The same holds for parameterized complexity results (compare, for example, Theorem 2 to Corollary 12). To know which results do generalize, and which do not, a case-by-case analysis will need to be performed.

10. Discussion

In the Introduction we presented two models of subset choice, Generalized Subset Choice and Additive Subset Choice. We observed that Additive Subset Choice fails to be descriptively adequate in situations where the value of a subset is not equal to the sum of the values of its elements (Table 1). To model such situations, models of subset valuation need to incorporate the possibility of value interdependencies between choice alternatives (Fishburn, 1992; Fishburn & LaValle, 1996). Introducing such value interdependencies, however, may lead one into the quicksand of computational intractability. For one, we know that if no constraints whatsoever are imposed on the amount and structure of value interdependencies—as is the case for Generalized Subset Choice—choosing a subset of maximum value is computationally unfeasible for all but very small choice sets (viz., then the decision-maker must search through all $2^{|V|}$ subsets of $|V|$ choice alternatives). This does not mean that all interactive

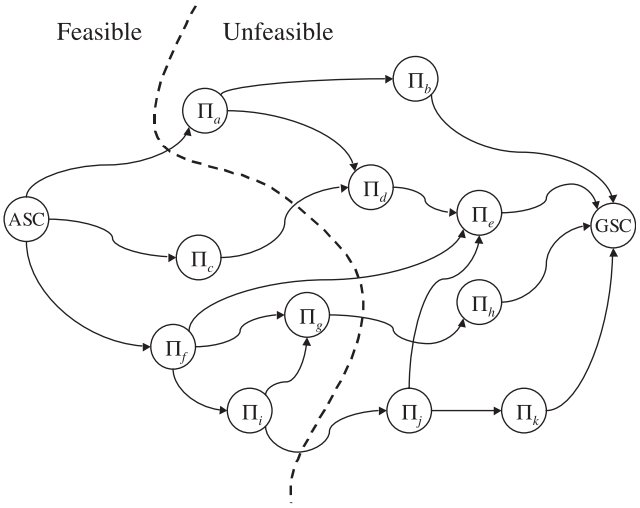

Fig. 6. Illustration of the space of possible subset choice models subsuming Additive Subset Choice (ASC) and subsumed by Generalized Subset Choice (GCS). Each path from ASC to GSC represents a chain of models $\Pi_1, \Pi_2, \ldots, \Pi_n$, such that ASC $\subseteq \Pi_1 \subseteq \Pi_2 \subseteq \cdots \subseteq \Pi_n, \subseteq$ GSC (The relation "$\subseteq$" is depicted in the figure with an arrow "$\rightarrow$"). Because ASC is computationally feasible, while GSC is computationally unfeasible, each chain contains a transition from the feasible to the unfeasible domain. The exact placement of the boundary between feasible and unfeasible models depends to some extent on assumptions about the size of problem parameters, processing speed and degree of parallelism, as explained in Section 1.1.

subset choice models are computationally unfeasible, but it does mean that the set of psychologically plausible interactive models is strongly constrained by the requirement of computational feasibility. Fig. 6 illustrates this point.

Fig. 6 shows a partition of the space of possible subset choice models into computationally *feasible* and *unfeasible* models, where a subset choice model is considered psychologically plausible *only if* it is feasible. The model Additive Subset Choice (ASC) is depicted on the far left in the computationally feasible part of the space, and the model Generalized Subset Choice (GSC) is depicted on the far right in the computationally unfeasible part of the space. The ASC model is subsumed under GSC as a special case (in short, ASC $\subseteq$ GSC). There are many (possibly intertwining) chains of models $\Pi_1, \Pi_2, \ldots, \Pi_n$, such that ASC $\subseteq \Pi_1 \subseteq \Pi_2 \subseteq \cdots \subseteq \Pi_n, \subseteq$ GSC. (The relation "$\subseteq$" is depicted in Fig. 6 with an arrow "$\rightarrow$"). Each such chain of models traces a path from the computationally feasible model ASC to the computationally unfeasible model GCS. Somewhere on that path a transition occurs from the feasible domain to the unfeasible domain—i.e., there are models Π_j and Π_{j+1}, $j = 1, 2, \ldots, n$, such that Π_j is feasible but Π_{j+1} is not. By tracing many different paths from ASC to GCS, and by observing where these transitions take place on a given path, we can gain insight into the properties of human value-structures

that are (in)sufficient to render subset choice computationally feasible. The space of possible models is vast, as there exist in principle infinitely many different subset choice models. In this article we investigated only a small number of subset choice models (Table 3). Our results nonetheless allow us to map out a significant part of the space of subset choice models. This is because computational feasibility of a task t is automatically inherited by every special case $t' \subseteq t$, and conversely, computational unfeasibility of a task t is automatically inherited by every generalization $t^* \supseteq t$ (Section 5.3). In the following we will explain this in more detail.

10.1. Special types of value-structures

We introduced the h-ary interaction model of subset choice in Section 2. Unlike the binary model of Fishburn and LaValle (1996), the h-ary model can accommodate *all* possible value interdependencies that may arise in subset choice. Using Eq. (4) we can always decompose a set of subset-value pairs into a set of vertex weights and hyperedge weights. This means that the h-ary model, in its general form, is computationally equivalent to Generalized Subset Choice, and consequently as implausible as a descriptive model. There exists a real possibility, however, that human value-structures are such that special types of weighted hypergraphs can model them. We next consider several conceivable restrictions that may apply to human value-structures.

10.1.1. Limited sensitivity

Humans are limited in the sensitivity with which they can detect and represent differences in value (Edwards, 1954). This limitation may express itself in human value-structures in several different ways. First, a decision-maker may distinguish only a limited number of value-levels for choice alternatives, causing the parameters $u_{\min}$ and $u_{\max}$ to be relatively small (e.g., in Fig. 1, $u_{\min} = 0$, and $u_{\max} = 9$). Second, a decision-maker may distinguish a limited number of value-levels for interactions, causing the parameters $\Delta_{\min}$ and $\Delta_{\max}$ to be relatively small (e.g., in Fig. 1, $\Delta_{\min} = 7$ and $\Delta_{\max} = 5$). Lastly, if the value associated with higher-order interactions is generally small, the decision-maker may fail to detect (or take into account) the presence of such interactions, causing the parameter ε to be relatively small (e.g., in Fig. 1, $\varepsilon = 2$). Using these parameters, we can define many different chains of models $\Pi_1, \Pi_2, \ldots, \Pi_n$, such that $\mathrm{ASC} \subseteq \Pi_1 \subseteq \Pi_2 \subseteq \cdots \subseteq \Pi_n, \subseteq \mathrm{GSC}$. Each model Π_j is defined by a different parameter setting, and for any two models Π_i and Π_j, we have $\Pi_i \subseteq \Pi_j$ if and only if the parameters $u_{\min}$, $u_{\max}$, $\Delta_{\min}$, $\Delta_{\max}$, and ε are set at least as large for Π_j as for Π_i. Consider, for example, the following three models:

(1) Π_i, where $u_{\min} = 5$, $u_{\max} = 5$, $\Delta_{\min} = 3$, $\Delta_{\max} = 2$, and $\varepsilon = 6$;
(2) Π_j, where $u_{\min} = 5$, $u_{\max} = 5$, $\Delta_{\min} = 1$, $\Delta_{\max} = 1$, and $\varepsilon = 2$;
(3) Π_k, where $u_{\min} = 10$, $u_{\max} = 10$, $\Delta_{\min} = 3$, $\Delta_{\max} = 2$, and $\varepsilon = 4$.

Here $\Pi_j \subseteq \Pi_i$ and $\Pi_j \subseteq \Pi_k$. The models Π_i and Π_k are not members of a common chain, however, because ε is larger for Π_i than for Π_k and, at the same time, $u_{\min}$ and $u_{\max}$ are larger for Π_k than for Π_i.

In Section 4 we studied the model UCG Subset Choice and found that it is NP-hard (Corollary 1). This means that, without imposing further constraints, the model Π^* with $u_{\min} = 0$, $u_{\max} = 1$, $\Delta_{\min} = 1$, $\Delta_{\max} = 0$, and $\varepsilon = 2$ is computationally unfeasible for all but small $|V|$. By implication, the same conclusion holds for all models that generalize Π^*, including the models Π_i, Π_j, and Π_k discussed above. In sum, our analyses show that limited sensitivity of humans is *not* sufficient to render interactive models of subset choice computationally feasible—not even when we assume that the decision-maker can distinguish at most *two* value-levels for alternatives, at most *two* value-levels for interactions, and can detect only *second*-order interactions.

10.1.2. Conflict versus surplus

We can distinguish between two different types of interactions, negative interactions (i.e., $\Delta(x, y, \ldots, z) < 0$) and positive interactions (i.e., $\Delta(x, y, \ldots, z) > 0$). Negative interactions may arise, for example, if choice alternatives share value-supporting properties (e.g., when a committee selects a subset of job applicants with overlapping skills) or when choice alternatives are in conflict (e.g., when a student selects a subset of courses with overlapping lecture times). Positive interactions, on the other hand, may arise when choice alternatives complement each other (e.g., the different parts of a computer system, see Section 2).

Intuitively, the presence of *both* positive and negative interactions seems to complicate subset choice, because one can never be sure whether adding alternatives will increase or decrease the total value. Is subset choice computationally easier in situations where either only negative or only positive interactions arise? We first consider the case with only negative interactions. In Section 4, we studied a particular value-structure in this domain: The unit-weighted conflict graph. In this value-structure, all alternatives have the same value, $u(x) = 1$, and all interactions are binary and have the same value, $\Delta(x, y) = -1$. We found that subset choice is of exponential-time complexity even if the value-structure is of this special type (unless P = NP, Corollary 1). Since

unit-weighted conflict graphs only contain negative interactions, Corollary 1 shows that the absence of positive interactions, by itself, does not make the task of selecting a maximum-valued subset computationally feasible.

To study the complexity of subset choice in the absence of negative interactions, we considered unit-weighted surplus graphs. The unit-weighted surplus graph is in a sense the "complement" of the unit-weighted conflict graph: all alternatives have the value $u(x) = -1$ and all interactions are binary and have the value $\Delta(x, y) = 1$. Interestingly, we found that if one's value-structure is modeled by this type of unit-weighted surplus graph then determining a maximum-valued subset is easy (Theorem 2): Starting with all alternatives and successively removing alternatives with degree 1 from V we obtain a subset A of maximum value. This tractability result of course generalizes to subset-choice models that are subsumed by USG Subset Choice, but not necessarily to more general models. Time-complexity of subset choice for value-structures that generalize the unit-weighted surplus graph remains an open problem.

10.2. Maximizing versus satisficing

Up to this point we have been assuming that the decision-maker chooses a subset of maximum value. Under this assumption, the h-ary interaction model of subset choice was shown to be computationally unfeasible, even when human value-structures are severely restricted (see Section 10.1 above). It may very well be that human decision-makers do not (generally) choose subsets with maximum value, but that they are content with any subset whose value exceeds some minimum threshold (cf. Herbert Simon's notion of "satisficing"). To model this possibility, we assume that a decision-maker has a preset threshold p, and s/he chooses a subset $A \subseteq V$, such that $u(A) \geqslant p$, if one exists. If no such subset exists, the decision-maker may adjust his/her threshold to $p' < p$ and try the task again for p'.

Although satisficing may seem easier than maximizing, the two types of tasks are generally of equivalent classical complexity—i.e., either both tasks are in P or both are NP-hard (see also Section 3). The reason is simple: Let $p_{\max}$ be the maximum value possible for some subset $A \subseteq V$. If we can determine a subset $A' \subseteq A$, such that $u(A') \geqslant p_{\max}$, and we can determine that no subset $B \subseteq V$ with $u(B) \geqslant p_{\max} + 1$ exists, then we have determined a maximum-valued subset, A'. Computing a maximum-valued subset is thus no harder than computing a satisfactory subset (or know that none exists) for thresholds $p_{\max}$ and $p_{\max} + 1$. For this argument to work, however, the aspiration level p must be able to take values as large as $p_{\max}$ and $p_{\max} + 1$. It is possible that human decision-makers have aspiration levels that are much smaller than $p_{\max}$. Is satisficing in those cases computationally easier than maximizing? We investigated this question by studying the parameterized complexity of subset choice for parameter p. The rationale behind this is the following: Determining a p-valued subset for small p is computationally feasible if and only if the exponential-time complexity inherent in subset choice can be confined to the parameter p.

We found that p-Subset Choice is W[1]-hard (Corollary 2). This means that there does not exist any procedure for computing p-Subset Choice that runs in time $O(f(p)|i|^{\alpha})$ (unless FPT = W[1]). In sum, satisficing is not any easier than maximizing—at least not when it comes to Subset Choice. Furthermore, we note that the W[1]-hardness result was obtained for the special case of p-UCG Subset Choice (Theorem 3). Hence, for every model Π, such that $\Pi \supseteq \Pi^*$, where Π^* with $u_{\min} = 0$, $u_{\max} = 1$, $\Delta_{\min} = 1$, $\Delta_{\max} = 0$, and $\varepsilon = 2$, we know that p-Π is W[1]-hard. This shows that desiring a p-valued subset, is *not* sufficient to render interactive models of subset choice computationally feasible even for small p—not even in the simplest of cases where we assume that the decision-maker can distinguish at most *two* value-levels for alternatives, at most *two* value-levels for interactions, and can detect only *second*-order interactions (cf. Section 10.1.1).

10.3. Choice versus rejection

In Section 10.2 we said that satisficing is not easier than maximizing. To be precise we should have said that satisficing is not easier than maximizing *if* the decision-maker satisfices *relative to the empty set*. Because this is often the most natural way to satisfice one may easily overlook the alternative—i.e., to satisfice *relative to the entire choice set V*. The latter form of satisficing is more natural in situations where the decision maker already owns all of the alternatives in V and s/he is to remove some alternatives from V so as to improve the value of the remainder (e.g., when one is presented with a fully loaded pizza and one can choose to remove some of its toppings; see Levin, Schreiber, Lauriola, & Gaeth, 2002). To model this situation, we assume that a decision-maker has a preset threshold q, and s/he chooses a subset $A \subseteq V$, such that $u(A) - u(V) \geqslant q$, if one exists. If no such subset exists, the decision-maker may adjust his/her threshold to $q' < q$ and try the task again for q'. In this case, the status quo is $u(V)$ instead of $u(\varnothing) = 0$. Is satisficing relative to $u(V)$ computationally equivalent to satisficing relative to $u(\varnothing) = 0$? Not necessarily. The reason is that p and q are more or less independent thresholds. For example, it is possible that $p_{\max}$ is small while $q_{\max}$ is large, and vice versa, all depending on $u(V)$.

We analyzed the parameterized complexity of q-Subset Choice and found that, unlike p-UCG Subset

Choice, q-UCG Subset Choice is in FPT (Theorem 4). This result shows that there exist value-structures for which determining a p-valued subset of V for small p is computationally unfeasible, while at the same time determining a q-valued subset of V for small q is feasible. These special value structures include unit-weighted conflict graphs (compare Corollary 2 to Theorem 4) and edge-weighted conflict graphs (compare Corollary 2 to Corollary 4), but not vertex-weighted conflict graphs or any of their generalizations (Theorem 5). If we introduce the possibility of limited sensitivity on the part of the decision-maker as in Section 10.1.1—i.e., relatively small $u_{\min}$, $u_{\max}$, $\Delta_{\min}$, $\Delta_{\max}$, and ε—the feasibility of choosing p-valued and q-valued subset diverges even further (Theorems 6 and 7).

Of course, our results do not show that p-Subset Choice is *always* harder than q-Subset Choice (compare, for example, Corollaries 10 and 11). There may very well exist value-structures for which the latter is harder than the former. All we have shown that this is not the case for value-structure that generalize the unit-weighted conflict graph (i.e., all models in Table 3 except USG Subset Choice). The reason that finding a q-valued subset is feasible for humans with limited sensitivity and conflict-graph value structures is that the decision-maker need only consider rejecting an alternative, $x \in V$, if it contributes negative value to the whole set, i.e., $u(V \backslash x)) > u(V)$. Because every removal of such an alternative will lead to an increase in total value, after at most q removals the value has improved by at least the amount q (see, for example, the proof of Theorem 7 and refer to Fig. 5 for an illustration).

10.4. Choosing subsets of fixed size

Lastly we considered the influence of subset size restrictions of the feasibility of subset choice. Specifically we studied the complexity of determining a maximum-valued subset A containing exactly k elements. At first sight the constraint, k, may seem to facilitate subset choice by constraining the space of possible choices. This intuition is incorrect, however, as it conflates size of the search space with the difficulty of finding a solution. To illustrate, consider again subset choice on unit-weighted surplus graphs. We showed that this task is easily solvable by simply successively removing all alternatives of degree 1 from V (Lemma 2). The resulting subset A has guaranteed maximum value. But what happens if we introduce the constraint that $|A|$ should be exactly k? The described polynomial-time algorithm for USG Subset Choice does not guarantee that $|A| = k$, and hence, does not automatically solve USG Exact Bound Subset Choice. Does there exist a different polynomial-time algorithm solving USG Exact Bound Subset Choice? Theorem 8 proves that there does not (unless $\mathrm{P} = \mathrm{NP}$). Furthermore, this unfeasibility result even applies if the decision-maker would be satisfied with a p-valued subset of size k for small p and/or k (Corollary 12).

10.5. Algorithms versus heuristics

Before closing, we would like to take this opportunity to briefly comment on a common reply to computational intractability results: That humans would use "heuristics" instead of algorithms to make decisions (Thagard & Verbeurgt, 1998; Martignon & Hoffrage, 2002; Martignon & Schmitt, 1999). In responding to this concern, we wish to clarify what we believe to be a confusion of different levels of explanation (see also Frixione, 2001; van Rooij, 2003). All models considered in this paper are situated at Marr's *computational level*—i.e., they specify a function that aims at describing the input–output mapping realized by the decision process. These models do not, in themselves, provide any description of the decision process. In fact the models make no claims about the decision process other than that it is a process that *somehow* realizes the described function. A description of the process would be situated at Marr's *algorithmic level*. At this level one would describe the effective procedure by which inputs are transformed into outputs. A heuristic explanation at this latter level violates the required fit between an algorithmic level theory and its computational level counterpart, as we elaborate next.

Let us define terms. First, let $\Pi : I \rightarrow O$ be a function that aims at describing the input–output mapping realized by a human decision process. An *algorithm* M for Π is a procedure that, when given any $i \in I$ as input, produces $M(i)$ as output, such that $\mathrm{M}(i) = \Pi(i)$, for all i. A *heuristic* H for Π is a procedure that when given any $i \in I$ as input, produces $H(i)$ as output, where *sometimes* $\mathrm{H}(i) = \Pi(i)$. Now two possibilities arise: Either (1) $\mathrm{H}(i) = \Pi(i)$ for all $i \in I$, or (2) there exists some $i \in I$ such that $\mathrm{H}(i) \neq \Pi(i)$. If (1) is the case, then H is in fact an algorithm for Π, and hypothesizing H as an algorithmic level description is perfectly consistent with hypothesizing function Π as a computational level description. If, on the other hand, case (2) is true, then Π and H are theoretically inconsistent. To appreciate this point it is important to keep in mind that Π is regarded as a *descriptive* model: If H is believed to be descriptive of the decision process then Π cannot be, and vice versa. Of course, if one believes that H is descriptive of the decision process, one may choose to reject Π and hypothesize a new input–output model Π' such that $\Pi'(i) = \mathrm{H}(i)$ for all $i \in I$ (provided, of course, that the available empirical data permit such a revision). This is easily done by simply defining $\Pi'(i) = \mathrm{H}(i)$. Once this theory revision is made, however, H has become an *algorithm* for the new computational level description Π' after all.

There are examples in the literature of algorithmic level descriptions that consist of non-algorithmic procedures (see e.g. Thagard's (2000) work on coherence reasoning). We think that such a loose fit between computational level and algorithmic level theories is a poor choice for cognitive psychologists. Even when exact correspondences are required, cognitive psychology has to deal with identifiability problems (Anderson, 1978), because any given function can be computed by many different algorithms. Allowing algorithmic level description to be inconsistent with computational level description can only make matters worse.

10.6. Conclusion

We have reviewed and discussed our main results from several different psychological perspectives: limited sensitivity, conflict versus surplus, maximizing versus satisficing, choice versus rejection, restricted subset size versus no size restriction. This has allowed us to map out part of the space of subset choice models depicted in Fig. 6. We would like to emphasize that the implications of our results far exceed what we can explicate in a single paper. We encourage researchers interested in subset choice to compare their own models with the ones considered here. If they find that their model is subsumed by one of our models then they can adopt the relevant feasibility results reported herein; conversely, if they find that their model subsumes one of our models then they can adopt the relevant unfeasibility results.

The more general aim of this paper was to demonstrate parameterized complexity analysis and how it naturally extends classical complexity theory in allowing more fine-grained investigation into the sources of complexity in a given task. As will be evident by now, the techniques of parameterized complexity allow us answer questions such as "Is subset choice facilitated by limited sensitivity for value differences?" and "Is choosing a satisfactory valued subset computationally easier than choosing a maximum valued subset?" It is our hope that the results reported in this paper motivate other researchers to adopt the same techniques for studying other choice models of interest.

Acknowledgments

The authors thank M. R. Fellows for inspiring discussions on parameterized complexity and thank A. J. J. Marley, M. Regenwetter, B. A. Mellers and an anonymous reviewer for helpful comments on earlier versions of this article. This research is supported by a research grant from the National Science and Engineering Research Council (NSERC) of Canada and a research grant from the University of Victoria awarded to U. Stege, and by NSERC Grant OGP0121280-97 awarded to H. Kadlec. Parts of this article appear in a doctoral dissertation submitted by I. van Rooij in partial fulfillment of the requirements for the Ph.D. degree in the Department of Psychology at the University of Victoria, BC, Canada. Some of the results reported in this article were presented at the 1st Annual Summer Interdisciplinary Conference, 2002, Squamish BC, Canada.

References

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Lawrence Erlbaum Publishers.

Barberá, S., & Pattanaik, P. K. (1986). Falmagne and the rationalizability of stochastic choices in terms of random orderings. *Econometrica*, *54*(3), 707–715.

Brandstädt, A., Le, V. B., & Spinrad, J. P. (1999). *Graph classes: A survey*. Philadelphia: SIAM.

Downey, R. G., & Fellows, M. R. (1999). *Parameterized complexity*. New York: Springer.

Downey, R. G., Fellows, M. R., & Stege, U. (1999a). Parameterized complexity: a framework for systematically confronting computational intractability. In F. Roberts, J. Kratochvil, & J. Nesetril (Eds.), *Contemporary trends in discrete mathematics: from DIMACS and DIMATIA to the future. DIMACS series in discrete mathematics and theoretical computer science*, vol. 49 (pp. 49–99).

Downey, R. G., Fellows, M. R., & Stege, U. (1999b). Computational tractability: The view from Mars. *Bulletin of the European Association for Theoretical Computer Science*, *69*, 73–97.

Edwards, W. (1954). The theory of decision making. *Psychological Bulletin*, *51*(4), 380–417.

Falmagne, J. -Cl., & Regenwetter, M. (1996). A random utility model for approval voting. *Journal of Mathematical Psychology*, *40*, 152–159.

Farquhar, P. H., & Rao, V. R. (1976). A balance model for evaluating subsets of multiattributed items. *Management Science*, *22*(5), 528–539.

Fellows, M. R. (2001). Parameterized complexity: The main ideas and some research frontiers. In P. Eades, & T. Takaoka (Eds.), *12th International symposium on algorithms and computation. Lecture notes in computer science*, vol. 2223 (pp. 291–307). Berlin: Springer.

Fellows, M. R. (2002). Parameterized complexity: The main ideas and connections to practical computing. In R. Fleischer, B. Moret, & E. Meineche Schmidt (Eds.), *Experimental algorithmics: From algorithm design to robust and efficient software. Lecture notes in computer science*, vol. 2547 (pp. 51–77). Berlin: Springer.

Fellows, M. R., McCartin, C., Rosamond, F., & Stege, U. (2000). Coordinatized kernels and catalytic reductions: An improved FPT algorithm for Max Leaf Spanning Tree and other problems. In S. Kapoor, & S. Prasad (Eds.), *Foundations of software technology and theoretical computer science Lecture Notes in Computer Science*, vol. 1974 (pp. 240–251). Berlin: Springer.

Fishburn, P. C. (1972). Interdependent preferences on finite sets. *Journal of Mathematical Psychology*, *9*, 225–236.

Fishburn, P. C. (1992). Utility as an additive set function. *Mathematics of Operations Research*, *17*, 910–920.

Fishburn, P. C., & LaValle, I. H. (1993). Subset preferences in linear and nonlinear utility theory. *Journal of Mathematical Psychology*, *37*, 611–623.

Fishburn, P. C., & LaValle, I. H. (1996). Binary interactions and subset choice. *European Journal of Operational Research*, *92*, 182–192.

Frixione, M. (2001). Tractable competence. *Minds and Machines*, *11*, 379–397.

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

Green, C. D. (1994). Cognitivism: Whose party is it anyway? *Canadian Psychology*, *35*, 112–123.

Gross, J., & Yellen, J. (1999). *Graph theory and its applications*. New York: CRC Press.

Grünwald, P. (2000). Model selection based on minimum description length. *Journal of Mathematical Psychology*, *44*, 133–152.

Karp, R. M. (1975). On the complexity of combinatorial problems. *Networks*, *5*, 45–68.

Kukla, A. (1989). Nonempirical issues in psychology. *American Psychologist*, *44*, 785–794.

Levin, I. P., Schreiber, J., Lauriola, M., & Gaeth, G. J. (2002). A tale of two pizzas: Building up from a basic product versus scaling down from a fully loaded product. *Marketing Letters*, *13*(4), 335–344.

Martignon, L., & Hoffrage, U. (2002). Fast, frugal, and fit: Simple heuristics for paired comparison. *Theory and Decision*, *52*, 29–71.

Martignon, L., & Schmitt, M. (1999). Simplicity and robustness of fast and frugal heuristics. *Minds and Machines*, *9*, 565–593.

Niedermeier, R. (2002). Invitation to fixed-parameter algorithms. Habilitationsschrift, University of Tübingen.

Oaksford, M., & Chater, N. (1998). *Rationality in an uncertain world. Essays on cognitive science of human reasoning*. East Sussex, UK: Psychology Press.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. New York: Dover Publications.

Parberry, I. (1994). *Circuit complexity and neural networks*. Cambridge, MA: MIT Press.

Parberry, I. (1997). Knowledge, understanding, and computational complexity. In D. S. Levine, & W. R. Elsberry (Eds.), *Optimality in biological and artificial networks?* (pp. 125–144). Hillsdale NJ: Lawrence Erlbaum Publishers.

Pitt, M. A., & Myung, I. J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences*, *6*, 421–425.

Regenwetter, M., Marley, A. A. J., & Joe, H. (1998). Random utility threshold models of subset choice. *Australian Journal of Psychology*, *50*(3), 175–185.

Robinson, D. N. (1999). Rationalism versus empiricism in cognition. In R. J. Sternberg (Ed.), *The nature of cognition* (pp. 79–110). Cambridge, MA: MIT Press.

Stege, U., van Rooij, I., & Hertel. A., HertelP. (2002). An $O(pn+1.151^p)$ algorithm for p-Profit Cover and its practical implications for Vertex Cover. In P. Bose, & P. Morin (Eds.), *13th International symposium on algorithms and computation. Lecture notes in computer science*, vol. 2518 (pp. 249–261). Berlin: Springer.

Thagard, P. (2000). *Coherence in thought and action*. Cambridge, MA: MIT Press.

Thagard, P., & Verbeurgt, K. (1998). Coherence as constraint satisfaction. *Cognitive Science*, *22*(1), 1–24.

Tsotsos, J. K. (1990). Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, *13*(3), 423–469.

Tsotsos, J. K. (1991). Is complexity theory appropriate for analyzing biological systems. *Behavioral and Brain Sciences*, *14*(4), 770–773.

van Rooij, I. (2003). *Tractable cognition*: *Complexity theory in cognitive psychology*. Ph.D. thesis, University of Victoria, Canada.

Wareham, H. T. (1998). *Systematic parameterized complexity analysis in computational phonology*. Ph.D. thesis, University of Victoria, Canada.