Chapter 2
Optimal State Change Attempts

In this chapter, we formalize the notion of a state change attempt. The idea is that a state change attempt, when successfully applied to a given tuple, will change the action attributes with the hope that these changes will result in a change in the state. For instance, considering the school data described in the previous chapter, SCAs represent policies designed with a certain goal in mind; we provide an example in which a change in the action variables that decreases class size may lead to better proficiency scores.

Definition 2.1 (State Change Attempt (SCA)). A *simple state change attempt* is a triple (A_i, vf, vt) where $vf, vt \in Dom(A_i)$ for some $A_i \in \mathbf{A}$. A (non-simple) *state change attempt* (SCA for short) is a set $\{(A_{i_1}, vf_i, vt_1), \ldots, (A_{i_k}, vf_k, vt_k)\}$ of simple state change attempts such that $i_j \neq i_k$ for all $j \neq k$.

When clear from context, we will refer to these concepts as *simple changes* and *changes*, respectively. Intuitively, a *simple* state change attempt modifies one attribute, while a state change attempt may modify more than one.

Definition 2.2 (Applicability of an SCA). Given a tuple t, an action attribute A_i, and $vf, vt \in Dom(A_i)$, a simple state change attempt (A_i, vf, vt) is *applicable* with respect to t if and only if $t(A_i) = vf$. The *result of applying a simple state change attempt* that is applicable with respect to t is t' where $t'(A_i) = vt$ and for state attribute $S_j \neq S_i$, $t'(A_j) = t(A_j)$. We use $\gamma(t, (A_i, vf, vt))$ to denote t', the tuple resulting from the application of SCA (A_i, vf, vt) to t.

A state change attempt $SCA = \{(A_{i_1}, vf_1, vt_1), \ldots, (A_{i_k}, vf_k, vt_k)\}$ is *applicable* with respect to t if and only if all (A_{i_j}, vf_j, vt_j) for $1 \leq j \leq k$ are applicable with respect to t.

Example 2.1. A simple state change attempt with respect to the school data from Example 1.1 could be $a_1 = (A_1, 8{,}700, 8{,}850)$, indicating that funding is increased from \$8,700 to \$8,850 per student, or $a_2 = (A_2, 62.3, 65)$ indicating that salaries are increased from 62.3 to 65% of the budget. Let $SCA = \{a_1, a_2\}$ be a state change attempt. If we assume that the values of the action attributes in the current

A. Parker et al., *Data-driven Generation of Policies*, SpringerBriefs in Computer Science,
DOI 10.1007/978-1-4939-0274-3_2, © The Author(s) 2014

environment are $t = (8{,}700, 64, 7, 3.2)$, then a_1 is applicable with respect to t, but a_2 is not. The result of applying a_1 to t is $\gamma(t, (A_1, 8{,}700, 8{,}850)) = t' = (8{,}850, 64, 7, 3.2)$. ■

Example 2.2. Looking now at the city government data from Example 1.2, a simple state change attempt could be $a_3 = (A_3, 6.5, 6.0)$, indicating that personal income tax is lowered from 6.5 to 6%, or $a_4 = (A_{10}, 15, 21)$, indicating that the funding for city hospitals is increased from \$12M per year to \$15M per year. Let $SCA = \{a_3, a_4\}$ be a state change attempt. Assuming that the values of the action attributes in the current environment correspond to the values of t_1 in Fig. 1.2:

$$t_1 = (2{,}500,\ 5.0,\ 6.5,\ 5.5,\ 350,\ 120,\ 70,\ 450,\ 430,\ 12,\ 150,\ 950)$$

then both a_3 and a_4 are applicable. The result of applying SCA is $\gamma(t, SCA) =$

$$(2{,}500,\ 5.0,\ 6.0,\ 5.5,\ 350,\ 120,\ 70,\ 450,\ 430,\ 15,\ 150,\ 950).$$ ■

The result of applying a state change attempt is therefore the result of applying each simple change. However, these changes do not occur without cost.

Definition 2.3 (Cost of a simple change attempt). Let $a = (A_i, vf, vt)$ be a simple state change attempt. The *cost of attempting a* is given by a real-valued function $cost : \{A_1, \ldots, A_m\} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, where $cost(A_i, vf, vt)$ is the cost of changing action attribute A_i from vf to vt.

Cost functions will be highly dependent on the application domain, and we assume them to be provided by a user. The *cost* of an attempt, $cost(SCA) = \sum_{a \in SCA} cost(a)$, is the sum of the costs of the simple state change attempts in SCA.

Example 2.3. Consider the same simple changes $a_1 = (A_1, 8{,}700, 8{,}850)$ and $a_2 = (A_2, 62.3, 65)$ from Example 2.1, and a third simple change $a_3 = (A_4, 3.8, 3.9)$ (i.e., increment the proficiency increase target from 3.8 to 3.9). A possible cost function could be defined in terms of monetary cost, in which: $cost(a_1) = 150 * s$ (where s is a constant set to the number of students affected), $cost(a_2) = 2.7 * A_1$, and $cost(a_3) = 0$ (no monetary cost associated with changing the proficiency increase target). ■

Example 2.4. A different cost function may be defined for the city government dataset, perhaps incorporating the political capital or risk necessary to undertake government actions. Consider again the state change attempt $SCA = \{a_3, a_4\}$ from Example 2.2 where $a_3 = (A_3, 6.5, 6.0)$ and $a_4 = (A_{10}, 15, 21)$. Lowering income tax may require striking deals with opposing political parties, and the effect on the budget may be negative at first, thereby making a_3 both financially and politically costly: $cost(a_3) = 200 + 0.5 * d$, where d indicates how many half percentage points the tax is lowered (1 for a_3). Influencing the city's policy towards a higher hospital budget may require political actions that are less costly given the expected benefits of such a change: $cost(a_4) = 5 + 2 * d'$, where d' indicates the number of \$100K increments are being proposed (60 for a_4). ■

To this point, we have looked at state change attempts that are always successful. However, in general, we cannot expect this to be the case—the funding per student may not change simply because one attempted to change it. We will assume state change attempts are only probabilistically successful—they only induce the change attempted according to a specified probability. Further, we will assume that the probability of any simple change occurring successfully depends on the entire set of changes attempted. For instance, when attempting to increase the proficiency target alone, one may expect a relatively small probability of the change actually occurring—the teacher's union is unlikely to accept an increase in their expected performance with no additional compensation. However, when attempting to increase the proficiency target along with an increase in teacher salaries, as in Example 2.3, there will be a higher probability that both changes will actually occur.

In general, such a change can cause four possible outcomes. It may be that both the proficiency target increases and the salaries increase, or that the proficiency target increases while the salaries do not change, or that the proficiency target stays the same while the salaries increase, or that both do not change. The probability of these outcomes is dependent on the total state change attempt: the school board is more likely to accept a salary increase along with an increase in the proficiency target while the teacher's union is more likely to accept a proficiency target increase as long as it also comes with a salary increase.

Example 2.5. Consider the situation described in Example 2.3. Here the state change attempt a_2 increases teacher salaries from 62.3 to 65%. On its own, this policy may anger taxpayers (who would foot the bill for the increase) and may only have a 10% probability of succeeding. Likewise, increasing per student funding might have a 15% probability of success. However, if the taxpayers happen to be willing to increase teacher salaries, then they will also tend to approve per student funding increases, perhaps leading to a joint probability of 9% that both of these will occur when attempted together.

Similar probability dependencies may be seen in the case of state change attempts in Example 2.4. The city government may be more likely to increase funding for the police department if other public safety increases are already being proposed (such as an increment of the budget for street lighting). Similarly, any policy involving an increment in funding is more likely to succeed along with an increment in the overall budget, as opposed to in combination with a reduction in funding for a different item. ■

Let SCA and $SCA' \subseteq SCA$ be state change attempts; suppose we have the conditional probabilities $pOccur(SCA'|SCA)$—this is the probability that only the actions in SCA' occur given that SCA is attempted. Such probabilities can either be derived from historical data or be explicitly stated by a user. When we say that a state change attempt SCA is "attempted" for a tuple t describing the current situation, this means that each $SCA' \subseteq SCA$ has the chance $pOccur(SCA'|SCA)$ of being successful, i.e., of having $\gamma(t, SCA')$ be the resulting tuple.

Example 2.6. Consider once again some state change attempts from Example 2.3: $a_1 = (A_1, 8{,}700, 8{,}850)$, and $a_2 = (A_2, 62.3, 65)$. If we apply the state change attempt $\{a_1, a_2\}$, there are four possible outcomes:

1. a_1 and a_2 could occur, increasing the funding per student to \$8,850 as well as increasing teachers' salaries to 65% of the budget;
2. a_1 could occur without a_2, increasing the funding per student without changing the budget teacher's salaries fraction of the budget;
3. a_1 could fail while a_2 occurs, leaving funding per student at \$8,700 but increasing teacher salaries; or
4. Both fail, leaving things as they are.

The probabilities of these outcomes are respectively given by the state change conjunction strategy as follows:

$$pOccur(\{a_1, a_2\} \mid \{a_1, a_2\}), pOccur(\{a_1\} \mid \{a_1, a_2\}),$$
$$pOccur(\{a_2\} \mid \{a_1, a_2\}), \text{ and } pOccur(\emptyset \mid \{a_1, a_2\}).$$

■

In the next section we will present effect estimators, which constitute an important part of the policy generation process.

2.1 Effect Estimators

The goal in this work is to allow an end user to take an event KB $\mathscr{K}$ and a goal G (some desired outcome condition on state attributes) that the user wants to achieve and find an SCA that "optimally" achieves goal G in accordance with some objective function (such as maximizing the probability of goal G being achieved while minimizing cost). We assume without loss of generality that all goals are expressed as standard conjunctive selection conditions [3] on state attributes. We now define *effect estimators*.

Definition 2.4 (Effect Estimator). For action tuple t and goal G, an *effect estimator* is a function $\varepsilon(t, G) \rightarrow [0, 1]$ that maps a tuple and a goal to a probability $p \in [0, 1]$.

Intuitively, $\varepsilon(t, G)$ specifies the conditional probability of goal G holding given that we are in a situation where the action attributes are as specified in t. This quantity can be estimated in many ways, some of which will be investigated later on in Chap. 3. As an initial example, one can imagine using some machine learning algorithm as an effect estimator to determine how often a school's reading score is above some number k given that the student-teacher ratio, funding per student, teacher salaries, etc. have some specific values.

2.2 State Change Effectiveness

We assume an environment where, just because a state change attempt is performed, it is not necessarily the case that all parts of the state change attempt will actually accomplish the attempted change. When one attempts to change the situation via a state change attempt *SCA*, any subset of *SCA* may succeed. For instance, if a certain policy involves decreasing the student/staff ratio and increasing the funding per student, it may be that the student/staff ratio increases as expected, but that the funding per student remains the same. Thus, to truly gauge the effectiveness of a state change attempt, we must consider the probability of each subset of the attempt occurring.[1]

Definition 2.5 (State Change Effectiveness). The probability of a state change attempt $SCA = \{(A_{i_1}, vf_1, vt_1), \ldots, (A_{i_k}, vf_k, vt_k)\}$ satisfying goal G when applied to the action tuple t is

$$pEff(t, G, SCA, \varepsilon) = \sum_{SCA' \in \mathscr{P}(SCA)} pOccur(SCA' \mid SCA)\varepsilon(\gamma(t, SCA'), G).$$

The computation of $pEff(t, G, SCA, \varepsilon)$ works by summing over all the state changes that may occur given that *SCA* is attempted: since any subset of *SCA* can occur, this summation ranges over $SCA' \subseteq SCA$. For each SCA' that may occur, one multiplies its probability of occurring given that *SCA* was attempted ($pOccur(SCA'|SCA)$) times the effectiveness of the given attempt according to ε (recall that $\gamma(t, SCA')$ is the action tuple resulting from the application of the state change SCA' to the original action tuple t). The following result shows that for arbitrary effect estimators, computing state change effectiveness is intractable.

Proposition 2.1. *For action attributes* **A***, state attributes* **S***, condition G, state change attempt SCA, action tuple t, and effect estimator* ε*, deciding whether* $pEff(t, G, SCA, \varepsilon) > 0$ *is NP-hard. Furthermore, if* $\varepsilon(.)$ *can be computed in polynomial time with respect to the action schema, the problem is NP-complete.*

Proof. Membership In NP: We show that deciding if $pEff(t, G, SCA) > 0$ is in NP with a witness $SCA' \subseteq SCA$ such that $pOccur(SCA') \cdot (1 - pOccur(SCA \setminus SCA')) > 0$. Since $pEff(s, G, SCA, \varepsilon)$ is a sum of non-negative terms, such an SCA' must exist when $pEff(t, G, SCA, \varepsilon) > 0$, and this can be checked in polynomial time with respect to $|\mathscr{A}|$ if $\varepsilon(.)$ can be computed in this time.

NP-hardness: We show by reduction from the NP-complete subset-sum problem, whereby we are given a finite set of integers I and an integer c and are asked to decide if there is a subset $I' \subseteq I$ such that $\sum_{i \in I'} i = c$ [1]. Let $I = \{i_1, \ldots, i_n\}$ and let $|\mathbf{A}| = \{A_1, \ldots, A_n\}$ with $Dom(A_i) = \{0, i_i\}$ for $A_i \in \mathbf{A}$. Let action

[1] In this work, we assume that each simple change attempt either succeeds or fails completely, i.e., no partial effects can occur.

tuple t be the all-zero n-tuple. Define $SCA = \{(A_j, 0, i_j) \mid 1 \leq j \leq n\}$, and define all $pOccur(SCA')$ to be zero unless $\sum_{A \in \mathbf{A}} \gamma(t, SCA')(A) = c$, in which case $pOccur(SCA') = 1/2$. let G be the condition $\top$, ε is a constant function that returns 1 for any KB and conditions, and suppose $\mathscr{K}$ is the empty event KB. Under these conditions $pEff(t, G, SCA, \varepsilon) > 0$ if and only if there is a subset of I summing to c.

($\Rightarrow$): Suppose $pEff(t, G, SCA, \varepsilon) > 0$ to show there is a subset of I summing to c. Since $pEff(t, G, SCA, \varepsilon) > 0$, there is $SCA' \subseteq SCA$ such that:

$$pOccur(SCA')(1 - pOccur(SCA \setminus SCA')) > 0.$$

This implies that $pOccur(SCA') > 0$, which in turn implies that for $t' = \gamma(t, SCA')$, $\sum_{A \in \mathbf{A}} t'(A) = c$. Since t is the zero n-tuple, all non-zero $t(A)$ result from changes in SCA', so we have that $\sum_{(A_j, 0, i_j)} i_j = c$. This gives the set $I' = \{i_j \mid (A_j, 0, i_j) \in SCA'\}$ which describes a subset of I whose sum is c.

($\Leftarrow$): Let $I' \subseteq I$ be the subset of I such that $\sum_{i_j \in I'} i_j = c$. Now consider state change $SCA' = \{(A_j, 0, i_j) \mid i_j \in I'\}$. Clearly $SCA' \subseteq SCA$ and $(\gamma(t, SCA'), \cdot) \in \sigma_G(\mathscr{K})$, so

$$pOccur(SCA') \cdot (1 - pOccur(SCA \setminus SCA')) \tag{2.1}$$

will be a term in the sum defining $pEff(t, G, SCA, \varepsilon)$. Since $\sum_{i_j \in I'} i_j = c$ we know that for $t' = \gamma(t, SCA')$, $\sum_{A \in \mathbf{A}} t'(A) = c$ and therefore that $pOccur(SCA') > 0$. Since $(1 - pOccur(SCA \setminus SCA'))$ is at least 0.5, this is the only term in Eq. 2.1 that may potentially be zero, proving that Eq. 2.1 is non-zero. Further, since all terms in the sum defining $pEff(t, G, SCA, \varepsilon)$ are zero or positive, this suffices to prove that $pEff(t, G, SCA, \varepsilon) > 0$. □

2.3 Optimal State Change Attempts

We now present various problems related to finding optimal state change attempts or policies. We assume that we are given $\mathbf{A} = \langle A_1, \ldots, A_n \rangle$ and $\mathbf{S} = \langle S_1, \ldots, S_m \rangle$, an event KB $\mathscr{K}$, an action tuple t describing the current values of the action attributes, a goal G over $\mathbf{S}$, and functions *cost* and *pOccur* as mentioned earlier:

1. **The Lowest Cost SCA Problem.** Given real number κ, does there exist an applicable change attempt SCA such that $cost(SCA) \leq \kappa$ and $pEff(t, G, SCA, \varepsilon) > 0$?
2. **The Highest Probability SCA Problem.** Given a real number $p \in [0, 1]$, does there exist a change attempt SCA such that $pEff(t, G, SCA, \varepsilon) \geq p$?
3. **The Optimal Threshold Effectiveness Problem.** Given a threshold $p \in [0, 1]$, and cost k, does there exist a change attempt SCA such that $pEff(t, G, SCA, \varepsilon) \geq p$ and $cost(SCA) \leq k$? This problem is the result of combining both the Highest Probability and Lowest Cost problems stated above.

4. **The Limited Cardinality SCA Problems.** Given a positive integer h, does there exist an *SCA* such that $|SCA| \leq h$ and *SCA* satisfies one of the conditions from the problems above? For instance, the limited cardinality highest probability SCA will, given a real number $p \in [0, 1]$, tell if there exists a change attempt *SCA* such that $pEff(t, G, SCA, \varepsilon) \geq p$ and $|SCA| \leq h$.

All of these problems are stated as decision problems that ask whether an *SCA* satisfying certain conditions exists. Search problems, to *find* such an SCA, can be analogously stated. We refer to any state change attempt that is a solution to one of these problems (say, problem P) as an *optimal state change attempt* (OSCA, for short) with respect to P.

Theorem 2.1. *If the effect estimator used can be computed in PTIME, the decision problems associated with the different definitions of optimal state change attempts belong to the following complexity classes:*

1. *The Lowest Cost SCA problem is NP-complete.*
2. *The Highest Prob. SCA problem is #P-hard and in PSPACE.*
3. *The Optimal Threshold Effectiveness Problem is #P-hard and in PSPACE.*
4. *All Limited Cardinality SCA problems are in PTIME.*

Proof. We prove each part in turn:

1. *Membership in NP*: Let ε be an effect estimator that can be computed in polynomial time with respect to $|\mathbf{A}|$. A witness change attempt *SCA*, along with $SCA' \subseteq SCA$ such that $cost(SCA) \leq \kappa$, $pOccur(SCA')(1 - pOccur(SCA \setminus SCA')) \cdot \varepsilon(t' = \gamma(t, SCA'), G) > 0$, and $(\gamma(t, SCA'), \cdot) \in \sigma_G(\mathscr{K})$ (implying $pEff(t, G, SCA, \varepsilon) > 0$) can be verified in polynomial time with respect to $|\mathbf{A}|$.

   NP-Hardness: The NP-hardness proof from Proposition 2.1 can be extended for this purpose by simply assuming that the only possible values in $Dom(A_i)$ are 0 and i_j (cf. the reduction in the proof) and assigning κ to be one greater than the sum of state change attempt costs. Therefore, any subset of *SCA* as defined for which there exists a subset of I summing to c can be seen as a state change attempt with the required property.
2. *# P-hard*: Let F be a SAT formula with variables $v_0, \ldots, v_n$, and N be a number. The problem of determining if the number of solutions to F is greater than or equal to N is #P-complete [2, 4]. We let there be an action attribute A_i for $i = 0, 1, \ldots, n$, with domain $Dom(A_i) = \{0, 1\}$. Let $t(A_i) = 0$ in the action tuple t for all A_i. All applicable simple state change attempts have the form $(A_i, 0, 1)$. Define the cost function to always return 0. We let *pOccur* always be 0.5, and let $\varepsilon(C_1, C_2)$ be one if $C_2 = F$ and C_1 exactly specifies a tuple t', where t' satisfies F, and zero otherwise. Define p to be $N \cdot 0.25$. The number of solutions to F is greater than or equal to N if and only if there is an applicable state change attempt *SCA* such that the cost of *SCA* is less than or equal to 0 and $pEff(t, F, SCA, \varepsilon) \geq p$.

Membership in PSPACE: All possible state change attempts can be checked by keeping track of only one at a time, which can clearly be done within the polynomial space constraint.

3. Analogous to part 2.
4. Since the size of the state change attempt is at most h, the space of possible state change attempts is bounded by a polynomial in $|\mathbf{A}|$. To see why this is the case, it suffices to recall that each $A_i \in \mathbf{A}$ has a *finite* domain $dom(A_i)$, that a state change attempt can specify at most one simple state change attempt per attribute in $\mathbf{A}$, and that $\mathscr{C}_h^n \in O(n^h)$, where $n = |\mathbf{A}|$. Therefore, in the worst case we have to check $\sum_{A \subseteq \mathbf{A}, |A| \leq h} \left(\prod_{A_i \in A} dom(A_i) \right)$ SCAs, which is a quantity in $O(n^h)$. □

In summary, in the proof of Theorem 2.1, all NP-hard reductions use the results of Proposition 2.1, the $\#P$-hard reductions use $\#SAT$ (the language $\{\langle F, n \rangle\}$, where F is a formula with exactly n solutions), membership in NP is shown by providing a witness, and membership in PSPACE and PTIME are shown by sketching algorithms, with the necessary properties, which are discussed in more detail in Sect. 2.4. In all cases, the source of the complexity can be seen to stem from $|\mathbf{A}|$.

2.4 Basic Algorithms for Computing OSCAs

We begin by giving an explicit polynomial time algorithm for solving the limited cardinality SCA problems; this corresponds to the procedure discussed in the proof of Theorem 2.1. We will then show how to extend this algorithm to solve all the problems posed in the last section.

This algorithm works by first enumerating each possible state change attempt with size at most h, then choosing the one which solves the appropriate problem. As discussed in the proof of Theorem 2.1, since there are only $O(|\mathbf{A}|^h)$ such state change attempts, this algorithm runs in PTIME with respect to the number of action attributes $|\mathbf{A}|$. The algorithm for enumerating state change attempts of size at most h along with their cost and probability of effectiveness is given as Algorithm 1, and we can show that this algorithm runs in time in $O(|\mathbf{A}|^h)$.

Proposition 2.2. *Algorithm 1 runs in time in* $O(|\mathbf{A}|^h)$ *and returns all* (SCA, c, ef) *where* $|SCA| \leq h$, $c = cost(SCA)$ *and* $ef = pEff(t, G, SCA, \varepsilon)$.

Proof. The size of R is at most $(|\mathbf{A}| \cdot \max_i(|dom(A_i)|))^h$, and Algorithm 1 terminates in $O(|R|)$ steps. Since we consider $\max_i(|dom(A_i)|)$ to be a constant, this is $O(|\mathbf{A}|^h)$. Further, to see that R is correct, clearly c and ef are correct for each (SCA, c, ef) (see line 10), so it remains to show that all SCA of size $\leq h$ are included in R. Consider any state chance attempt SCA to show there is a (SCA, c, ef) in R. We show this by induction on $|SCA|$. As a base case, when

Algorithm 1 limitedSCASet(t, G, ε, h)
Returns the set (SCA, c, ef), where c is the cost of state change attempt SCA and ef is the probability of effectiveness of SCA

```
1: Let R = ∅ // the set to be returned.
2: Add (∅, 0, pEff(t, G, ∅, ε)) to R. // Initialize R with empty state change attempt.
3: for each A_i ∈ A do
4:    for each value v ∈ dom(A_i) do
5:       continue if v = t(A_i) // Go to next value, t won't be changed by this SCA.
6:       // iterate over all members of R, growing those that are small enough.
7:       for each (SCA, c, ef) ∈ R do
8:          continue if |SCA| = h.
9:          Let SCA' = SCA ∪ {(A_i, t(A_i), v)}.
10:         Let c' be the cost of SCA' and ef' be pEff(t, G, SCA', ε).
11:         Add (SCA', c', ef') to R.
12:      end for
13:   end for
14: end for
15: return R
```

$|SCA|$ is zero, (SCA, c, ef) is in R. Supposing all SCA of size k are in R to show that any SCA_{k+1} of size $k+1$ is in R. Let $SCA_k \cup \{(A_i^*, vf, vt)\}$ be SCA_{k+1}. Because $|SCA_k|$ has size k, there is (SCA_k, c, ef_k) in R. Therefore when we run line 11 with $SCA' = SCA_k$, $A_i = A_i^*$ and $v = vt$, then $(SCA_{k+1}, c_{k+1}, ef_{k+1})$ will be added to R. Thus all SCA of size less than or equal to $k+1$ will be in R as (SCA, c, ef) with correct c and ef. □

Using Algorithm 1, we can now compute solutions to each of the limited cardinality SCA problems.

- **Lowest Cost Limited SCA Algorithm**: For cost threshold κ, let R be limitedSCASet(t, G, ε, h), eliminate all $(SCA, c, 0)$ from R, then eliminate all (SCA, c, ef) from R where $c > \kappa$. Return true if R is non-empty, false otherwise. $(SCA, c, ef) \in R$ has minimal c.
- **Highest Probability Limited SCA Algorithm**: For probability threshold p, let R be limitedSCASet(t, G, ε, h), eliminate all (SCA, c, ef) where $ef < p$ and return true if R is non-empty, false otherwise.
- **Optimal Threshold Effectiveness Limited SCA Algorithm**: For probability threshold p and cost threshold κ, let R be limitedSCASet(t, G, ε, h), and eliminate all (SCA, c, ef) from R where either $c > \kappa$ or $ef < p$. Return true if R is non-empty, false otherwise.

Each of those algorithms correctly computes the associated decision problem, as a corollary of Proposition 2.2.

Corollary 2.1. *Each of the following algorithms correctly computes the associated decision problem: Lowest Cost Limited SCA Algorithm, Highest Probability Limited SCA Algorithm, and Optimal Threshold Effectiveness Limited SCA Algorithm.*

To extend this technique to the non-limited, general versions of the various problems, one simply needs to solve the limited version of the problem with h equal to $|\mathbf{A}|$; however, in this case the algorithm will no longer run in polynomial time (cf. Theorem 2.1).

Again, each of the corresponding algorithms will correctly compute the associated decision problem, as a corollary of Proposition 2.2.

References

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.
2. Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 1 edition, April 2008.
3. Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I.* Computer Science Press, 1988.
4. L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

http://www.springer.com/978-1-4939-0273-6

Data-driven Generation of Policies

Parker, A.; Simari, G.I.; Sliva, A.; Subrahmanian, V.S.

2014, X, 50 p. 15 illus., Softcover
ISBN: 978-1-4939-0273-6