Incentive Engineering through Subgraph Matching — with Application to Task Allocation

Mark Ebden[1], Dong Huynh[2], Luc Moreau[2], and Stephen Roberts[1]

[1] Dept. of Engineering Science, University of Oxford, OX1 3PJ, United Kingdom
mark.ebden@eng.ox.ac.uk, www.robots.ox.ac.uk/∼mebden
[2] Electronics and Computer Science, University of Southampton, Building 32, SO17 1BJ, United Kingdom

Abstract. We use provenance graphs to solve a problem within incentive engineering: motivating humans to accept proposals generated by agents. Across several provenance graphs created within the HAC-ER disaster-management system, we ran retrospectively a bespoke algorithm for subgraph matching in order to extract narrative information from the provenance data. The output of the algorithm comprised a series of text messages which, had they been generated at the time of the disaster trial, would have been transmissable with the specific intention of encouraging participants not to reject certain tasks.

The algorithm found all expected subgraphs within the provenance graphs, on an any-time basis and in a time *linearly* proportional to the number of nodes. Our algorithm is extendable to other situations in which agents present tasks to humans.

Keywords: incentive engineering, subgraph matching, task allocation, human-agent collectives, provenance graphs, disaster management

1 Introduction

We use an accountable information infrastructure based on provenance graphs [6] recording the formation of human-agent collectives (HACs) [7] to solve a problem within incentive engineering, namely motivating participants to accept proposed tasks.

One of three key findings of a recent article in disaster management was that particular task allocations suggested to people by a centralized planning agent "may need particular support" in order to succeed; the authors recognized "the need to ensure the agent's recommendations come with proper explanations in order to be trusted and further augment their capability" [15]. We propose that such explanations can result from the systematic interpretation of provenance graphs, leading to automatic generation of output in the form of messages to participants, and other results leveraging a historical narrative that humans can understand and accept. This helps to mitigate, for example, the tendency of some participants to favour joint tasks with those who are merely standing nearby.

To this end, subgraph-matching algorithms have the capability to translate provenance graphs into incentivizing messages to participants, where here a subgraph is a connected, smaller graph within our provenance graphs (one provenance graph exists

per scenario executed, usually over the course of 20-30 minutes). Central to this capability is the fact that certain important forms of incentivization rely on a sense of historical flow in the tasks undertaken within HACs: people are more willing to accept a task if they perceive it as fitting within a meaningful personalized context.

The main contribution of this paper is to develop a computationally effective method for engineering incentives in HACs. The generic nature of provenance data allows our method to apply to a range of other applications, but in the present work we generate an approach to messaging participants in an effort to improve the probability that task allocations will be accepted.

2 Methods

This section describes how, across provenance graphs which were created within the HAC-ER disaster-management system [14] (specifically, a beta version known as AtomicOrchid [15]), we ran retrospectively a bespoke algorithm for subgraph matching in order to extract narrative information from the provenance data. The output of the algorithm comprises a series of text messages which, if they were generated during a disaster trial, would be transmissable with the specific intention of encouraging participants not to reject certain tasks.

2.1 Applying Incentive Theory to HAC-ER

In each HAC-ER disaster trial, the aim is to sequentially form pairs of complementarily skilled responders (human participants) to travel together, pick up targets (there are four types of target, none of which can be handled by a single responder), and deliver the targets to designated locations. The responders are then free to form new pairs to address new targets. To facilitate the formation of pairs, a centralized planning agent provides each unoccupied responder with a proposal consisting of a target plus a teammate. The planning agent determines these proposals according to an algorithm optimizing for global performance, but each responder is allowed to reject tasks they judge to be personally unfavourable — for example, the proposed target or the proposed teammate might be deemed to be too far away.

A problem that has been encountered is that responders decide to accept or reject tasks without necessarily knowing each task's global significance or how personally enjoyable it would be. Incentives would be desirable in certain cases, to increase the acceptance rate particularly of the most useful tasks. Within the theory of incentives [8] there are a number of ways of classifying how people can be motivated and in which kinds of contexts, and increasingly researchers are considering how agents can guide human choice-making (see [2] and the references therein). For HAC-ER, we consider in Table 1 a 2×2 matrix describing non-exhaustively several types of incentive that could be used to motivate responders.

In this work we focus on incentives which can easily be derived from a provenance graph. The messages we generate thus fall within the four categories appearing in the corresponding column of Table 1:

S1. Reporting previously completed similar task: "You have completed this type of task before," or "Your teammate has completed this type of task before."

S2. Reporting previously experienced geography: "The task is in a geographic region you have visited previously," or "The task is in a geographic region your teammate has visited previously."
S3. Reporting previous joint success: "You have worked successfully with this responder before."
S4. Reporting common experience with third parties: "You have each worked with responder X before, successfully." (Where X is outside of the currently proposed pair of responders.)

Evident from these messages is the blurring of the line between the two rows of the 2×2 matrix; that is, there exist messages (such as those related to S1 or S2) which can be considered hybrids of teammate-based and target-based incentives.

	Easily derived from a provenance graph	**Requiring data beyond a provenance graph**
Based on the target	• Reporting previously completed similar task • Reporting previously experienced geography	• Reporting the global importance of the task • Reporting predicted opportunities to work in the target's geographic area
Based on the responder pair	• Reporting previous joint success • Reporting common experience with third parties	• Reporting predicted opportunities to work together in future • Reporting the teammate's performance ranking

Table 1. Categorization of incentives for HAC-ER responders to accept tasks.

2.2 Subgraph Matching

Towards the task of summarizing important information from provenance graphs, recent work has considered compressing such graphs via node aggregation and edge aggregation [12]. Here we instead seek to extract intact portions of a graph through subgraph matching (also known as the subgraph isomorphism problem, on attributed relational graphs), for which several algorithms exist [9]. The extension of pioneering work by Ullmann in the 1970s [19] led to fast algorithms including QuickSI [17] and VF2 [1]. The latter was recently beaten by LAD [18], and in the present work we present an algorithm which in Matlab runs more slowly than the LAD C code but, within our application domain, scales more efficiently with graph size.

The algorithm is available for download at github.com/OxfordML/ENPG. For each of the four types of incentive described in Section 2.1, a subgraph was defined with which to query the provenance graph (the provenance graph itself and all four subgraphs adopted the W3C standard PROV model [3, 11]). In querying these subgraphs, successful finds were tabulated and their parameters were used to generate messages. Recent

work considered the (reversible) translation of provenance graphs into Controlled English [16], with each sentence representing one node or relationship; whereas, each of our messages corresponds to several nodes and edges (a subgraph). A message template for each of the four incentive types was created, with any blanks filled in by text extracted automatically from the successfully located subgraphs (this approach also lends itself to straightforward one-off translation by designers to non-English languages if required). Our subgraphs comprised between four and eleven nodes, and they are as follows.

S1. Reporting previously completed similar task: Figure 1 gives the four-node provenance subgraph required to generate the first type of incentive. The name of node $\texttt{Agent}_1$ equals the responder's name and the name of node $\texttt{Entity}_4$ contains a particular keyword ('Safe'), as before. Node $\texttt{Entity}_2$ was labelled in a simple preprocessing step, of computational complexity linear in the number of graph nodes, to be a 'useful' instantiation of a responder. Usefulness was assessed by whether it represented a responder and whether it was connected to at least one entity representing a target (see $\texttt{Entity}_3$). Node $\texttt{Entity}_4$ contains a particular keyword ('Safe'), indicating that the target had been delivered successfully.

S2. Reporting previously experienced geography: Figure 2 gives the four-node provenance subgraph required to generate the second type of incentive. The name of node $\texttt{Agent}_1$ equals either the responder's name or the prospective teammate's name. The name of node $\texttt{Activity}_3$ contains a particular keyword ('PickUp') to indicate that a target had been picked up. The name of node $\texttt{Entity}_4$ contains a term such as 'Animal', 'Radioactive', etc to indicate it represents a target. Node $\texttt{Entity}_2$ is a useful instantiation of the responder (the concept of useful instantiations that was defined for the S1 subgraph is employed identically for all four subgraphs). The longitude and latitude are read from the node attributes of $\texttt{Entity}_4$, as these indicate the location of the target that the responders would need to have reached in order to manage the pickup. This location is then compared to the location of the *proposed* target in order to determine whether they fall within the same geographic region; numerous methods are suitable here but for simplicity we divide the HAC-ER area into two regions ('East' and 'West') based on longitude alone in order to determine whether the geography of $\texttt{Entity}_4$ matches that of the proposed new target.

S3. Reporting previous joint success: Figure 3 gives the six-node provenance subgraph required to generate the third type of incentive. The name of node $\texttt{Agent}_1$ equals the responder's name, and the name of node $\texttt{Agent}_5$ equals the prospective teammate's name. Nodes $\texttt{Entity}_2$ and $\texttt{Entity}_4$ are useful instantiations of these responders, as described for S1. Node $\texttt{Entity}_6$ contains a particular keyword ('Safe'), indicating that the target had been delivered successfully.

S4. Reporting common experience with third parties: Figure 4 gives the eleven-node provenance subgraph required to generate the fourth type of incentive. The name of node $\texttt{Agent}_1$ equals the responder's name, and the name of node $\texttt{Agent}_{10}$ equals

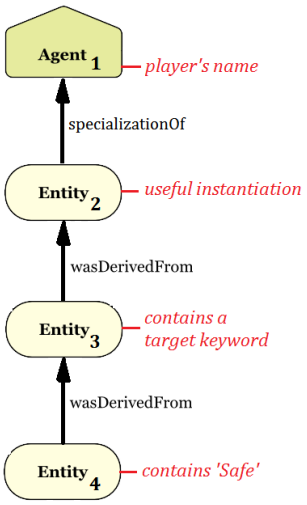

Fig. 1. The four-node subgraph to be detected for incentive S1.

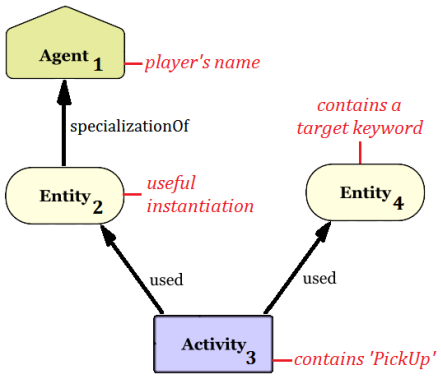

Fig. 2. The four-node subgraph to be detected for incentive S2.

the prospective teammate's name. Nodes $\texttt{Entity}_6$ and $\texttt{Entity}_{11}$ contain a particular keyword ('Safe'), as before. Other nodes are also equivalent to those for incentive subgraph S1.

Each of the four subgraphs was defined mathematically by a tuple $\mathcal{H}$ comprised of four constituents:

- An $m \times m$ adjacency matrix $\mathbf{H}$ in which the 1's were replaced with codified edge attributes, e.g. '8' for a PROV *specializationOf*
- An m-vector $\mathbf{t}$ listing codified node types, e.g. '1' for a PROV entity
- A set of m strings $\mathcal{N}$ listing the names of each node, e.g. an agent might be called 'medic95'
- A set of string pairs, $\mathcal{W}$, with each pair giving the type of node attribute (e.g. 'longitude') and its value (e.g. '-1.18' in degrees from the prime meridian)

The assignment of node types and edge attributes is described further in the appendix. The computer code is designed for application to provenance graphs in order to find simple subgraphs that can be expressed as tree structures (acyclic graphs). We ignored edge directionality in the provenance graphs. Explicitly stating the directionality was not required in any instance of relevance to current purposes, since directionality can be inferred using the edge attributes and node types; thus it was acceptable to convert the provenance graph and the four subgraphs to undirected graphs. Even for a simple Entity-wasDerivedFrom-Entity subgraph, where directionality would seem to be important, any ambiguity can be resolved by using node names and/or node attributes to help define the subgraphs. The advantage of ignoring edge directionality lies in allowing the algorithm to operate on a symmetric adjacency matrix, reducing its run time. This restriction could be relaxed if required.

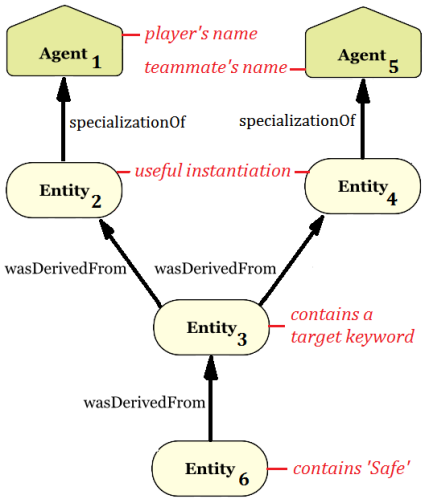

Fig. 3. The six-node subgraph to be detected for incentive S3.

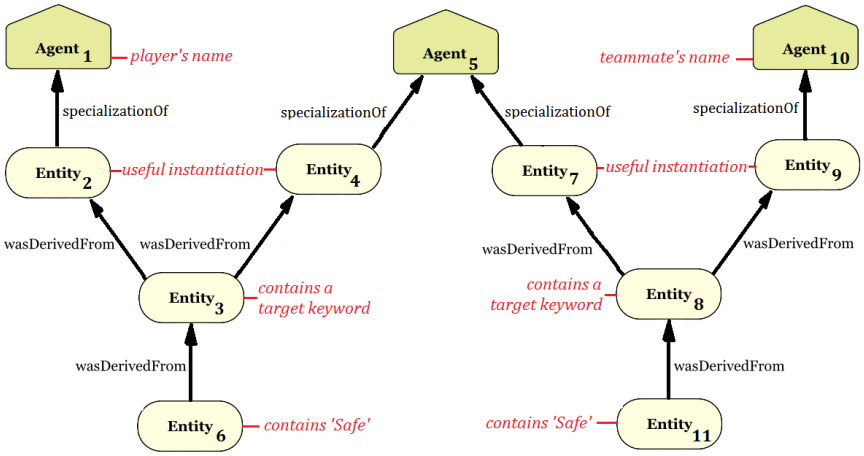

Fig. 4. The eleven-node subgraph to be detected for incentive S4.

With the n-node provenance graph and m-node subgraphs defined ($253 \leq n \leq 2098$, $4 \leq m \leq 11$), the code called a recursive function to explore simultaneously the nodes of the provenance graph and of each subgraph, terminating with an exhaustive list of all matches of the subgraph within the graph. Pseudocode is given in Algorithm 1. The function is called with $\mathcal{G}$ and $\mathcal{H}$ containing within them static information about the provenance graph and the subgraph, where $\mathcal{H}$ is defined earlier in this section and $\mathcal{G}$ is defined similarly (its adjacency matrix is $\mathbf{G}$). The m-vector $\mathbf{p}$ is initially comprised of zeros except for the first element, which is set to a node index from $\mathbf{G}$. If the function call is successful, $\mathbf{p}$ will fill up with other indices to nodes in $\mathbf{G}$, until the complete m-node subgraph is specified. The function is intended to be called n times, each with a different starting node within $\mathbf{G}$. The function returns two outputs: the set of subgraphs $\mathbf{Y}$ that have been identified so far (each being a valid configuration of $\mathbf{p}$), and a success code c which is 1 if at least one additional subgraph is being returned, or 0 otherwise.

The code was executed on seven provenance graphs collected during HAC-ER trials in November-December 2012 and in July-August 2014. Each trial comprised one computer agent directing six to eleven participants who completed a total of between four and sixteen tasks. The graph statistics are described in Table 2; graph density is defined as the number of directed edges divided by $n(n-1)$.

Algorithm 1 Identify in a provenance graph $\mathcal{G}$ all instances of a particular subgraph $\mathcal{H}$

```
1:  function FINDSUBGRAPHS (p, G, H)
2:      if p(i) > 0 ∀i then return (p, 1)
3:      else
4:          Y ← ∅, c ← 0
5:          h ← index of first zero entity in p
6:          for {i|p(i) > 0} do
7:              for each neighbour g of the p(i)th node of G do
8:                  if g's node type = t(h) AND g's name = N(h) AND
9:                     edge attributes between nodes g and p(h) ⊇ H(h, i) AND
10:                    attributes of node g ⊇ W(h) then
11:                    p(h) ← g
12:                    (X, r) ← FindSubgraphs (p, G, H)
13:                    if r = 1 then
14:                        Y ← Y ∪ X
15:                        c ← 1
16:                    end if
17:                end if
18:            end for
19:        end for
20:        return (Y, c)
21:    end if
22: end function
```

Graph name	Number of nodes	Number of edges	Density	Number of agent nodes
A	253	592	1.86%	12
B	875	2069	0.54%	7
C	1359	3279	0.36%	9
D	1485	3562	0.32%	9
E	1682	4173	0.30%	11
F	1784	4270	0.27%	7
G	2098	5142	0.23%	10

Table 2. Statistics for the seven provenance graphs under study. The number of agent nodes equals the number of human participants plus one for the central planning agent.

Graph	Time (s)	Number of subgraphs detected			
		S1	S2	S3	S4
A	0.3	8	8	4	0
B	1.9	16	20	8	0
C	4.8	24	24	12	26
D	6.1	32	32	16	24
E	5.4	18	22	9	13
F	3.6	16	16	8	10
G	6.2	22	26	11	13

Table 3. Results for the seven provenance graphs under study. The detection rate was 100%.

3 Results

For each of the seven graphs, subgraphs were detected and the corresponding messages listed in Section 2.1 were generated. The number of instances of each subgraph are recorded in Table 3, along with the CPU time required to find them. Each of these times represents the average over five runs on an Intel® Core™2 Duo (2.66 GHz) desktop computer. The reported subgraphs represented 100% of the instances expected after manual inspection of the provenance graphs using ProvStore visualization software [6].

The number of S1 subgraphs was consistently twice that of S3 subgraphs, owing to the nature of what the two subgraphs represented: each successfully delivered target ($\texttt{Entity}_4$ in Figure 1 or $\texttt{Entity}_6$ in Figure 3) corresponded to one instance of the S3 subgraph and to two instances of the S1 subgraph. The number of S2 subgraphs was usually the same as that of S1 subgraphs, but in three cases (B, E, and F) it was higher. This was due to the fact that in some cases a target would be picked up but not delivered — i.e. some tasks were incomplete. The number of S4 subgraphs was not very dependent on the numbers found for the other three subgraphs, as it indicated mainly the extent to which responders moved from one team to another. For each of the

two smallest provenance graphs (A and B), the HAC-ER trial evidently had not been conducive to team changes, and hence zero instances of S4 were detected.

Although the CPU time rose erratically with the number of nodes, the trend overall was linear. It would be almost exactly linear for sparse random graphs, as the computational complexity of our algorithm is $\mathcal{O}(n)$. The densities of the seven provenance graphs were very low, and each graph contained significant structure — for example, one agent node had over 250 edges — which explains the erratic rise, i.e. the high deviations (residuals) from a line of best fit. In a given graph, artificially reducing the number of nodes (removing leaves first, in order to simulate dynamics in reverse) also demonstrated a linear trend — see the ~~solid dot-markered~~ line in Figure 5. The ~~open circle~~ dot-markered line in the same plot show runtimes of LAD [18], for comparison. LAD was faster but in the figure both lines have been normalized to have a mean of unity, since our algorithm was implemented in Matlab only and LAD is available in C only. The supralinear scaling of LAD represents a potential concern for larger provenance graphs.

The CPU time is also a function of each subgraph's topological structure and the specifications for its nodes and edges. Although an initial inspection of Algorithm 1 reveals that the complexity is $\mathcal{O}(m!)$ in the worst case, in practice it is substantially better for provenance graphs when the information about node attributes and edge attributes is used to prune options at each step. To demonstrate the scalability of the algorithm, we artificially reduced the number of nodes in the S4 subgraph, decrementing progressively from 11 to 2, to discover that the algorithm was approximately $\mathcal{O}(m)$. However, there is no simple way to state the dependence of complexity upon m in the general case.

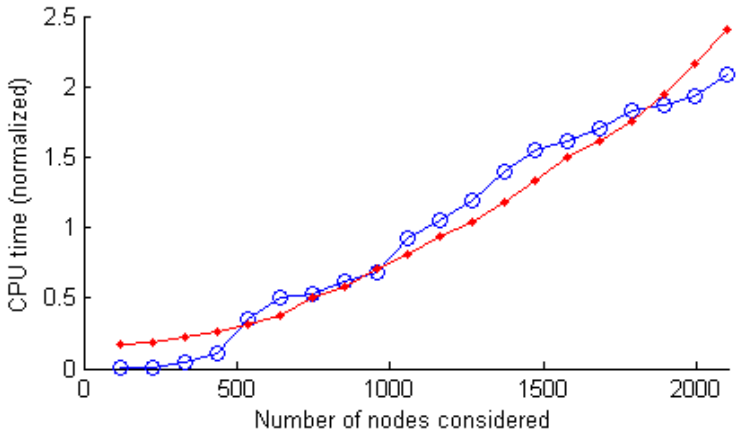

Fig. 5. The time required to find subgraphs in graph G (see Tables 2 and 3) or in smaller versions of G. The blue, open circle-markered line represents results from the authors' algorithm. The red, solid dot-markered line gives results from LAD [18]. Each line has been normalized to have a mean of unity.

4 Discussion

Within the subdiscipline of organizational behaviour known as incentive engineering, there is growing scope to use the information infrastructures emerging from the science of HACs. We have initiated this by providing viable feedback mechanisms to allow incentivization through interaction with agents and information sources. Namely we have made use of a system which captures human actions and decisions to ensure accountability, as a means to additionally feed information back to players with the objective of improving task-acceptance rates.

Our algorithm was successfully able to find all the expected subgraphs within HACER provenance graphs and to generate a variety of meaningful messages on an any-time basis, i.e. from whatever provenance data had been collected up to the point of execution. Since the computational complexity for subgraph matching with homogeneous nodes and edges is ~~NP-hard~~, we are particularly pleased that the algorithm scaled so well by taking into consideration the rich information available in provenance-graph nodes and edges. We have allowed subgraphs to be discovered in linear-order time.

NP-complete

The impact of the generated messages has not been optimized through experimental validation: until now, our focus has been the more application-agnostic aim of providing a mechanism to aid researchers situated at the intersection of incentive engineering and HACs [13]. We seek collaboration with those working in disaster management or any other application area who wish to apply this method in order to increase the task-acceptance rate among their participants. We note that subgraphs S3 and S4 are potentially suited to incentive networks [10] in which players practice *directed altruism*, and separately that crowdsourcing platforms can act as testbenches for incentivization mechanisms [4].

Regarding theoretical extensions of the work in future, we are considering the following three ideas: (a) incentives can result from reasoning mechanisms that exploit trust at the same time as provenance information — specifically, task-allocation proposals can be accompanied by information from trust models of the humans and agents involved; (b) graph compression could be achieved using spectral methods, extracting 'eigenprovenance' in an unsupervised learning methodology rather than one relying on predetermined subgraphs; (c) statistical classification techniques and other analytics-based methods [5] can be brought to bear on the issue of validating which of the four types of incentive are useful under which circumstances.

Acknowledgments

We gratefully acknowledge funding from the UK Research Council for project ORCHID, grant EP/I011587/1, www.orchid.ac.uk.

References

1. L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–72, 2004.
2. M. Fenster, I. Zuckerman, and S. Kraus. Guiding user choice during discussion by silence, examples and justifications. In *20th European Conference on Artificial Intelligence*. ECAI, Montpellier, France, 27-31 August, 2012.
3. P. Groth and L. Moreau. PROV-Overview. an overview of the PROV Family of Documents. Technical report, April 2013.
4. C.-J. Ho, A. Slivkins, and J. Wortman Vaughan. Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems. May 2014.
5. T.D. Huynh, M. Ebden, W. Jiang, J. Fischer, S. Roberts, and L. Moreau. Provenance network analytics. *Under preparation*, 2015.
6. T.D. Huynh and L. Moreau. ProvStore: a public provenance repository. In *Fifth International Provenance and Annotation Workshop*. IPAW, Cologne, 9-13 June, 2014.
7. N.R. Jennings, L. Moreau, D. Nicholson, S.D. Ramchurn, S.J. Roberts, T. Rodden, et al. On human-agent collectives. *Communications of the ACM*, 2014.
8. J.-J. Laffont and D. Martimort. *The Theory of Incentives: the principal-agent model*. Princeton University Press, 2002.
9. J. Lee, W.-S. Han, R. Kasperovics, and J.-H. Lee. An in-depth comparison of subgraph isomorphism algorithms in graph databases. *Proceedings of the VLDB Endowment*, 6(2):133–44, 2012.
10. Y. Lv and T. Moscibroda. Incentive networks. In *Twenty-ninth AAAI Conference*. Association for the Advancement of Artificial Intelligence, Austin, USA, 25-30 January, 2015.
11. L. Moreau and P. Missier. PROV-DM: The PROV Data Model, W3C Recommendation REC-prov-dm-20130430. October, 2013.
12. Luc Moreau. Aggregation by provenance types: A technique for summarising provenance graphs. In *GRAPHS AS MODELS 2015, an ETAPS Workshop*. Electronic Proceedings in Theoretical Computer Science, April 2015.
13. V. Naroditskiy, S. Stein, L. Tran-Thanh, M. Vlassopoulos, and N.R. Jennings. Referral incentives in crowdfunding. In *Conference on Human Computation and Crowdsourcing*. HCOMP, Pittsburgh, USA, 2-4 November, 2014.
14. S.D. Ramchurn, E. Simpson, J.E. Fischer, T.D. Huynh, Y. Ikuno, S. Reece, et al. HAC-ER: A disaster response system based on human-agent collectives. In *14th Int. Conf. on Autonomous Agents and Multi-Agent Systems*. AAMAS, Istanbul, Turkey, 4-8 May, 2015.
15. S.D. Ramchurn, F. Wu, W. Jiang, J. Fischer, S. Reece, C. Greenhalgh, et al. Human-agent collaboration for real-world disaster response. In *AAMAS Third International Workshop on Human-Agent Interaction Design and Models*. International Conference on Autonomous Agents and Multiagent Systems, Paris, 5-9 May, 2014.
16. D.P. Richardson, L. Moreau, and D. Mott. Beyond the graph: telling the story with PROV and Controlled English. In *Second Annual Fall Meeting of the International Technology Alliance*. ITA, Cardiff, UK, September, 2014.
17. H. Shang, Y. Zhang, X. Lin, and J.X. Yu. Taming verification hardness: An efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment*, pages 1–12, 24-30 August, 2008.
18. C. Solnon. AllDifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12-13):850–64, August 2010.
19. J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

Appendix: Codifying the PROV data

Values for the node-type encoding appear in Table 4. For efficiency of programming, it is possible to extend the table to include more specialized classes of these three cases. For example, we found it useful to assign a '4' for each entity representing a target, and '5' for each entity representing a useful agent instantiation (these concepts are elaborated in Section 2.2).

Values for the edge-attribute encoding appear in Table 5. The edge-attribute numbering system facilitates binary-coding. For example, in adjacency matrix $\mathbf{G}$, if the ith provenance node is connected to the jth provenance node by a SPECIALIZATION edge and a DERIVATION edge, $\mathbf{G}(i, j) = 24$. This system simplifies the $\supseteq$ operator in Algorithm 1.

Although two nodes in any of our provenance graphs can be connected by more than one type of edge, our subgraph-matching algorithm is written in such a way that it is not possible to specify multiple edges between two nodes within the query subgraph itself. This is not a practical limitation, since the designer of the subgraph can always choose its nodes and edges in such a way that only one critical edge need be specified. If additional edges occur in the provenance graph itself they are ignored during the subgraph query so as not to overlook any matches.

Node type	Value
ENTITY	1
ACTIVITY	2
AGENT	3

Table 4. Coded values for the three types of provenance node.

Edge type	Value
INFLUENCE	1
ALTERNATE	2
MENTION	4
SPECIALIZATION	8
DERIVATION	16
MEMBERSHIP	32
COMMUNICATION	64
START	128
END	256
USAGE	512
GENERATION	1024
INVALIDATION	2048
DELEGATION	4096
ASSOCIATION	8192
ATTRIBUTION	16384

Table 5. Coded values for the fourteen types of provenance edge attribute.