Effective Use of Analysts' Effort in Automated Tracing

Jane Huffman Hayes, Department of Computer Science, College of Engineering, University of Kentucky, Lexington, USA

Alexander Dekhtyar, Department of Computer Science, Cal Poly, San Luis Obispo, USA

Jody Larsen, Department of Computer Science, College of Engineering, University of Kentucky, Lexington, USA

Yann-Gaël Guéhéneuc, Department of Software and Computer Engineering, École Polytechnique de Montréal, Montreal, Canada

Abstract. Because of the large amount of effort it takes to manually trace requirements, automated traceability methods for mapping textual software engineering artifacts to each other and generating candidate links have received increased attention over the past fifteen years. Automatically generated links, however, are viewed as candidates until human analysts confirm/reject them for the final requirements traceability matrix. Studies have shown that analysts are a fallible, but necessary, participant in the tracing process. There are two key measures guiding analyst work on evaluation of candidate links: accuracy of analyst decision and efficiency of their work. Intuitively, it is expected that the more effort the analyst spends on candidate link validation, the more accurate the final traceability matrix is likely to be, although the exact nature of this relationship may be difficult to gauge outright. To assist analysts in making the best use of their time when reviewing candidate links, prior work simulated four possible behaviors and showed that more structured approaches save the analysts' time/effort required to achieve certain levels of accuracy. However, these behavioral simulations are complex to run and their results difficult to interpret and use in practice. In this paper, we present a mathematical model for evaluating analyst effort during requirements tracing tasks. We apply this model to a **simulation** study of twelve candidate link validation approaches. The simulation study is conducted on a number of different datasets. In each study, we assume perfect analyst behavior (i.e., analyst always being correct when making a determination about a link). Under this assumption, we evaluate the estimated effort for the analyst and plot it against the accuracy of the recovered traceability matrix. The effort estimation model is guided by a parameter specifying the relationship between the time it takes an analyst to evaluate a presented link and the time it takes an analyst to discover a link not presented to her. We construct a series of effort estimations based on different values of the model parameter. We found that the analysts' approach to candidate link validation – essentially the order in which the analyst examines presented candidate links – does impact the effort. We also found that the lowest ratio of the cost of finding a correct link from scratch over the cost of recognizing a correct link yields the lowest effort for all datasets, but that the lowest effort does not always yield the highest quality matrix. We finally observed that effort varies by dataset. We conclude that the link evaluation approach we call "Top 1 Not Yet Examined Feedback Pruning" was the overall winner in terms of effort and highest quality and, thus, should be followed by human analysts if possible.

Keywords. Requirements, measurement, traceability, analysts' effort, effort model, feedback

1 Introduction

Traceability is defined as "the ability to describe and follow the life of a requirement in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases" [1]. Traceability is a useful activity in that the resulting Requirements Traceability Matrix (RTM) (generically called a Traceability Matrix (TM)) can be used to support many undertakings, such as change impact analysis, regression testing, etc. In general, traceability matrices, if built, are generated manually or with general purpose techniques and are not kept up to date, largely due to the required time and effort.

To address the problem of traceability and updates, traceability research has concentrated on the problem of automating the tracing process. The key starting point for most such research on automating the tracing process is that a candidate traceability matrix can be built quickly and cheaply using an automated method (the main cost being the set-up cost of the analysis). In our prior work [2], we observed that even a high-recall, low-precision candidate TM already generates savings as compared to the analyst's need to examine every pair of low-level/high-level elements when measured in term of selectivity. Such research on automating the tracing process and others do not assume that automated techniques necessary yield better rankings. Indeed, a feedback loop could actually decrease the accuracy of the rankings of candidate links. Rather, we assume that adding techniques that increase precision and recall would, overall, improve the rankings. However, to the best of our knowledge, no previous works fully studied whether a combination of techniques (be them automated or manual) would improve the ranking.

Information retrieval (IR) methods have demonstrated usefulness in building links between textual artifacts [2][3][5][6]. These methods reduce the analyst effort to that of vetting, i.e., validating, the returned links. In our prior work [2][3][7], we concentrated on adapting various information retrieval techniques to the problem of tracing software artifacts and on evaluating the accuracy of the results. Information Retrieval (IR) methods examine the elements of a high level artifact (such as a concept specification) and return the elements deemed relevant from a lower level artifact (such as design or source code). These methods retrieve the majority of the relevant elements (high recall or coverage) but tend to also retrieve many false positives (low precision).

There are two directions in which research on automating tracing proceeds from this point. The first direction adopts, adapts, and develops new automated tracing methods, which keep high recall but provide improved precision. The second direction studies the interactions between the analysts and the automated tracing software and uses these interactions to improve the overall accuracy of the results. This paper follows the second direction. To further illustrate the importance of the analysts in tracing, we now present a motivating example.

1.1 Motivating Example: HeartPace

Traceability is sometimes perceived as an activity that provides little value to a software organization, often because TMs are built manually and the task is never completed (or not accurately). Thus, the organization does not reap the benefits of a TM. When TM building is supported by accurate automated methods, traceability is a value proposition, an activity that is central to the activities of a company. Consider the case of the HeartPace Company, Inc., builder of pacemakers. HeartPace is about to release a new product, the wireless pacemaker-plus. The FDA guidelines for the software validation of medical devices state: "software validation includes confirmation of conformance to all software specifications and confirmation that all software requirements are traceable to the system specifications [3]." Therefore, HeartPace must provide evidence that the software requirements are traceable to the system specifications, i.e., HeartPace must build the TM. The choices are to: (a) build the trace manually (which studies have shown to be time consuming and error prone); (b) build the trace automatically (which has been shown to be fast but prone to false positives); or (c) build a candidate trace automatically and then correct it (traces are called "candidate" until human analysts approved them). Studies have shown that (c) is indeed what practitioners do. The analysts feedback is not only the required final step but it also improves the TMs generated by automated methods [2][4].

Of the three options available to it, HeartPace chooses option (c). Because it is in business to make a profit, HeartPace wants to make the best use of its analysts' effort[1]. The effort to generate a candidate traceability matrix using an automated method is relatively small (even when the software installation, setup, data importation and output exportation are included in the overall count). HeartPace can select one of the many Information Retrieval techniques built into one of the available open-source requirements tracing tools [29] known to produce good results. HeartPace analysts expect to see a high-recall, low-precision candidate traceability matrix that they must examine further. If HeartPace has limited schedule and resources, which is inevitably the case, the candidate link validation method that builds the most accurate TM wins while, alternatively, if the accuracy of the final TM is fixed (ideally, it shall be 100% accurate with no missing links and no false positive links included), the candidate link validation method that reaches the required accuracy with the least effort is the preferred one. There are a number of candidate link vetting methods that can be used by the analysts. They can review each entry in the candidate TM by examining all candidate links for each high-level element (requirement) in the document order. They can review the candidate TM entries in a global order by link relevance estimated by the automated method. They can use a helper automated technique called "feedback processing" that re-orders outstanding candidate links based on the links that have already been vetted. The question the management and the analysts must answer is therefore: "which link vetting strategy to select?"

1.2 Contributions

This paper presents a *simulation study* that answers the HeartPace management question for a set of candidate link evaluation techniques. This study is organized as follows:

- We present the candidate link evaluation methods organized around three core behavioral approaches:
    - order in which candidate links are vetted by the analyst,
    - use of automated helper feedback processing technique to reorder outstanding candidate links based on information provided in the links already vetted by the analyst, and
    - pruning, i.e., analyst decision to not consider any links beyond a certain point as being summarily irrelevant/false positive.
- We compute for four different datasets the accuracy of various traceability matrices constructed by *a simulated analyst* following each of the candidate link evaluation techniques. We track the accuracy of the analyst-constructed TM after each step in evaluation.
- We apply an effort estimation model for traceability tasks developed by Dekhtyar, Hayes, and Smith [6] (specifically, we use the simplified model described in that work) to estimate the effort of building each TM constructed during the simulation process. Using this estimation technique we compare the effort used by each of the simulated analyst approaches to achieve the TMs with a specific preset accuracy. The simplified effort estimation model relies on a single parameter: the ratio between the effort that it takes to evaluate a single candidate link and that to manually discover a link that has not been recovered by an automated method. We construct effort estimations for each of the candidate link evaluation methods for a range of values of the parameter.

1.3 Paper Organization

[1] Making the best use of analysts' effort is difficult, which is the reason why "Cost-Effective" was named as the #2 Grand Challenge [5] facing requirement practitioners and researchers. The challenge reads "2. Cost-Effective - The return from using traceability is adequate in relation to the outlay of establishing it".

The paper is organized as follows. Section 2 introduces the effort model and prior findings on study of the analysts. Section 3 details the study. Section 4 presents the research question results and discussion. Section 5 addresses related work. Section 6 concludes and examines future work.

2 Automated Requirements Tracing and Analyst Behavior Models

Our study combines prior work on automating requirements tracing and analyst effort estimation. In this section, we briefly describe the automated requirements tracing techniques used in this study and the (simplified) analyst effort estimation models for requirements tracing tasks originally introduced by Dekhtyar, Hayes, and Smith [6].

2.1 Automated Requirements Tracing

In prior work, we reported on a number of Information Retrieval techniques used to generate candidate link lists [2][7]. For this study, we selected one technique, vector space retrieval using term frequency-inverse document frequency (tf-idf) term weighting, completed with standard Rochio feedback processing technique [8]. Feedback processing is used optionally – some analyst behavior models (a.k.a. candidate link vetting methods) described below take advantage of it, while other models do not. We selected this technique because it is the most commonly applied IR technique for tracing and it has been shown to outperform other techniques [2]. We briefly describe how these techniques are applied to the problem of traceability.

Tracing tasks involve two textual artifacts of the software lifecycle (e.g., a requirements document and a test plan). Both artifacts are split into individual elements. The tracing task is to build a mapping from the elements of one artifact (we refer to it as the source or high-level document) to the elements of the second artifact (which we call the target or low-level document). We assume that the elements of each textual artifact can be decomposed into sets of words. As in previous studies [2,4,7,9,17,18,19], we only assume sets of words and do not use other non-textual information from the elements of the artifacts, such as UML class diagrams or relations embodied in tables. We will consider such information in future work.

The IR technique, vector space retrieval, converts each textual element into a vector of keyword weights. If $V=\{t_1,\ldots,t_N\}$ is the list of all keywords found in the artifacts, then an element d is represented as a vector $d=\{w_1,\ldots,w_N\}$ of keyword weights, where each keyword weight w_i is computed as the product $w_i=tf_i \times idf_i$. Here, tf_i, called term frequency of the keyword, is the normalized frequency of the occurrences of the keyword w_i in our element d. Inverse document frequency, idf_i, of w_i is computed as $idf_i = \log(M/M_i)$, where M is the total number of elements in the document and M_i is the number of elements that contain w_i.

For each high-level document element, the vector space retrieval technique provides a ranking of low-level document elements based on their similarity scores with some high-level document. The similarity between two vectors is computed as the cosine of the angle between the vectors:

$$sim(q,d) = \frac{q \cdot d}{\sum_{i=1}^{N} q_i^2 \cdot \sum_{i=1}^{N} d_i^2}$$

The quality of the rankings is measured through precision and recall. Precision, the accuracy of the ranking, is the percentage of retrieved links that are correct. Recall, the coverage of the ranking, is the percentage of correct links that were retrieved. In some settings, such as for Independent Verification and Validation (IV&V) or Independent Software Nuclear Safety Analysis (ISNSA) of a safety-critical system, recall may be more important than precision (as the consequence of a missed link may be serious).

After the ranked candidate link list is built, it can be improved via a feedback processing mechanism. Relevance feedback analysis utilizes user input to improve the performance of the retrieval algorithms by adjusting the keyword weights of high-level document vectors according to the relevant and irrelevant documents vetted by the analyst [2]. In general, feedback processing involves examining a subset of the links in the candidate link list and determining whether each link is correct or not. Let q be a high-level requirement and D_q be the set of all low-level requirements retrieved by an IR technique. Suppose a subset

of D_q was examined and broken into two sets: R_q and I_q, of relevant (correct) and irrelevant (false positive) links. The standard Rochio feedback processing technique, which we use in this study, uses this information to change the vector q as follows:

$$q_{new} = \alpha q + \frac{\beta}{|R_q|} \sum_{d \in R_q} d - \frac{\gamma}{|I_q|} \sum_{d \in I_q} d.$$

where α, β, and γ are normalizing constants, which indicate the relative importance of the original vector (α), positive information (β), and negative information (γ). Positive feedback may affect the recall of the candidate link list (new relevant links may be retrieved) while negative feedback may affect its precision (other false positives can potentially be removed from the list).

2.2 Analyst Effort Estimation

Tracing manually (or with a generic technique such as the search of a word processor) generally follows three steps [9]:

- "manually assign keywords to all elements of all artifact levels or build detailed keyword thesauri,
- manually or semi automatically perform all searches for low-level elements that map to high-level elements, and
- render a decision for each discovered candidate link."

When automating these steps, "[t]he analyst's role would change from a human search engine to a verifier who checks the automatically generated candidate RTMs. Software would assume the responsibility of indexing the high- and low-level elements and determining (at the outset) pairs of similar elements." [9] In the automated scenario, the analysts will basically examine the candidate TM and perform two activities: (1) vet returned links (correcting errors of commission) and (2) search for missing links (errors of omission).

Prior work in automated traceability has assumed an implicit effort model; researchers assumed a higher quality candidate trace matrix would result in lower analyst effort in building the final trace matrix. Yet, this implicit effort model has not been widely formalized or validated [6]. To address this shortcoming, we must first define the analyst effort. A theoretical effort model of an analyst working with a result from an automated traceability technique was developed by Dekhtyar, Hayes, and Smith [6].

Consider a tracing task with n high-level elements, m low-level elements, with $N = n \bullet m$ theoretical candidate links. Let an automated tracing method return $K \leq N$ candidate links[2]. Let *True* be the total number of true links in the dataset (the size of the ground truth TM), *Hits* be the total number of true links retrieved, and *Misses* be the total number of false positive links retrieved. Then, the general model for analyst effort estimate can be described using the following formula [6]:

$$Effort(\Lambda, \Delta, \Omega) = \Lambda \cdot Hits + \Delta \cdot Misses + \frac{\Omega \cdot (True - Hits) \cdot (N - (Hits + Misses))}{N} \quad (1)$$

This model relies on three parameters: Λ, Δ, and Ω which represent the following quantities[3]:

Parameter	Meaning
Λ	Average amount of time to confirm an automatically retrieved candidate link
Δ	Average amount of time to reject an automatically retrieved candidate link

[2] An automated method returns a link if it evaluates the similarity of the high-level and low-level elements forming the link above a certain, pre-defined in the method threshold value.

[3] These parameters were labeled α, β, and γ in prior work [6]. We rename them for this paper to avoid confusion with the Rochio feedback parameters.

Ω	Average amount of time to discover a true link not retrieved by an automated method

These three parameters were introduced because it takes different amounts of time to search for missing links than to vet retrieved links [10]. Our prior work and anecdotal evidence show that analysts tend to not look for missing links [10], [11], which is unfortunate because spending all allotted time vetting retrieved links can never result in finding more *true* links than were on the retrieved candidate link list . Also, if analysts knew when to stop vetting links and start searching for missing links (referred to as pruning), they could best use their time to maximize the number of correct links in the final TM. The latter statement requires empirical validation. To facilitate the validation, we switch to a simplified model, in which we assume that $\Lambda = \Delta$, i.e., that it takes as much time to vet a true link as it does to vet a false positive link:

$$Effort(\Lambda,\Omega) = \Lambda \cdot K + \Omega \cdot \frac{(True - Hits) \cdot (N - K)}{N} \quad (2)$$

In earlier work, we looked to see if, theoretically, analysts might decide to stop vetting links at a certain point in time and start looking for missing links [6]**.** We found that there is a theoretical break point, though it might be subtle (examining 97 links to find 8 true links makes sense if the ratio of $\Omega/\Lambda = 12$, but if $\Omega/\Lambda = 8$, it is less effort (than for $\Omega/\Lambda = 12$) to examine 83 links to find 7 true links). Some automated techniques may appear to be preferential due to high recall and precision, but depending on the value of Ω/Λ, a technique with lower recall/precision may account for less effort. As such, the effort model described in Equation (2) can be reduced, with some minor loss of accuracy, to a model that depends on only one parameter, describing the analyst effort:

$$\sigma = \frac{\Omega}{\Lambda} \quad (3)$$

Given a specific tracing task, the ratio σ of the effort that it takes to correct a type II error to the effort that it takes to vet a link/correct a type I error is likely to depend on two main characteristics: the nature/size of the tracing task itself and the intrinsic qualities of the analysts performing the task. A larger number of potential links (i.e., larger sizes of the tracing tasks) at a constant density of true links leads to higher σ values—it takes longer to search for a link. Different analysts have their own intrinsic "speeds" of discovery and this speed can affect the ratio σ in either direction.

2.3 Analyst Simulation

In this paper, we undertake a number of studies to examine real datasets (with a simulated perfect analysts: if the analysts vet a link as true it is true) with a variety of analysts' behavior for a variety of values of σ to see where we find the smallest effort. The goal of this work is to provide guidance to analysts on what method to apply and when to stop vetting links.

We simulate analyst behavior because our goal is to provide project managers and analysts with some estimates of the accuracy that they could expect if they choose a specific requirements tracing technique (from the list discussed above) to complete the tracing process. As stated above, the efficacy of the tracing process depends highly on the nature of the tracing tasks at hand: the size of the task, the density of the trace links, the size of individual elements in the high-level and low-level artifacts, and the language in which the artifacts are written. It also depends on the analysts. Thus, there is no universal value for the effort model parameters. However, in practice, the model estimation parameter σ, the ratio between the time to evaluate a single link and the time to recover a missed link, can be determined by project manager in a reasonably straightforward way by a simple trial-and-error procedure (have an analyst evaluate a few links, and have an analyst attempt to recover a known link that has been removed from the TM).

Consequently, the results presented in this paper are akin to the old-style tables of logarithms or the current p-value tables still in use today: by determining the parameter σ, the manager can then look at the simulation study results and obtain effort estimates for the tracing task. The results presented here apply to four specific datasets, however, they can serve as starting points for such analysis.

We discuss various analysts' behaviors and methods next.

2.4 Methods of Candidate Link Validation

We graphically illustrate a typical tracing scenario **as example, recall that our study uses simulated, not real, analysts**. Two software engineering artifacts, a requirements specification and a software test plan, have been traced to each other using an automated technique. In this case, the analysts have built the trace matrix to address a question of interest: whether every requirement is being adequately validated by the test cases that map to it. As can be seen in Figure 1, a number of test cases have been retrieved in the candidate trace matrix, but only three of the six are True links, two were incorrectly retrieved (False Positive), and three were not retrieved (Missing links).

There are different orders in which analysts can assess individual candidate links from the candidate traceability matrix provided by an automated technique, as well as other options that the analysts can choose when running the technique (such as feedback). As each represents an analysts' decision, we generally refer to them as analysts' methods. Each method is represented by the following three attributes:

- ordering of the candidate links,
- using feedback, and
- pruning of the candidate links.

We examine these next.

2.4.1 Ordering of Candidate Links

The candidate links can be ordered globally or locally [14]. In global ordering, all the candidate links are ranked in descending order based on their relevance weight or similarity score, as given by the automated technique. In such a case, the document ordering is lost. For example, the first requirement from the requirement document may be listed far down the list, perhaps with a low relevance to one of the test cases (say 0.7).

In local ordering, the candidate links are grouped by the source or high-level element. The high-level elements (requirements, in our example) are presented in document order. For each high-level element, the candidate links from the target or low-level artifact (test cases, in our example) are presented in descending order of their similarity score. In such a case, the first requirement element from the requirement document would be listed first, along with all test plan elements that traced to it. In Figure 2, left side, we see the candidate links presented to the analysts in local order—each high-level element has a list of low-level elements in sorted order of relevance weight. In Figure 2, right side, we see the global ordering scenario. For example, when considering a requirements specification ID 788, our local ordering would list all test cases related to this requirement in rank order based on the similarity/relevance weight.

Based on an earlier study [23], it appears that some analysts may only examine the low-level links with the highest similarity scores for a given high-level element (akin to only looking at the first URLs that appears after one submits a search string to Bing or Google). This method is called Top 1 Not Yet Examined, because the analysts in essence only examine the top candidate link that has not previously been seen. Other methods included accept-focused (analysts that tend to accept all links and reject very few), preview (analysts who tend to spend much time examining links before starting to select or reject links), and iterative (analysts who work on links and then return to them again in an iterative fashion) [23]. Simulation of these is future work.

2.4.2 Using Feedback

Feedback processing was introduced above. In the "no feedback" scenario, the analysts provide a decision for each candidate link (Yes, it is a link (click on Link), No, it is not a link (click on Not a Link)). The confirmed links are added to the final traceability matrix (TM). Rejected links are removed from the final TM.

In the "feedback" scenario (shown in Figure 3), the analysts provide a decision for each candidate link (each test case of the test plan that was retrieved as relating to a requirement). A screenshot of the interactive RETRO.NET [14] tracing tool with analyst feedback provided for two elements of the MODIS dataset (see Section 3.5) is shown in Figure 4. After the decision is provided, the automated technique

executes one cycle of automated feedback processing for the given high-level element (requirements). The results of this feedback processing are immediately incorporated into the candidate links and may cause the reordering of the candidate links or the removal or addition of candidate links to the list: the set of retrieved links K from equation (2) depend on the q_{new} equation in Section 2.1, which itself depends on the analyst's feedback. In a non-feedback scenario, the set K would be different than that with feedback.

2.4.3 Pruning

In the "no pruning" scenario, the analysts will examine every candidate link retrieved by the automated method. In other words, the analysts do not know when all the true links have been retrieved and have been examined. In Figure 5, the analysts would examine all the links for requirement SDP3.3-4, though only the top link (in green) is a true link.

In the "pruning" scenario, the analysts stop examining candidate links for a given high-level element (requirement SDP3.3-4) after having examined the last true link (test case L1APRO1-I-1) for that element. Typically, pruning cannot be fully automated but heuristics exist and have been successfully used in the literature. One such heuristics purports that after a number of false links, the analysts would "know", i.e., can assume, that the remaining links cannot include true links. Another heuristic [28] uses a threshold t to prune the set of traceability links, keeping only links whose similarities values are greater than or equal to t $\in$ [0; 1]. The choice of the pruning heuristic is out of the scope of this paper, in which we only consider whether the lists of links are complete or pruned to assess the impact of these two possibilities on the simulated analysts' efforts.

We combined the attributes into the following methods:

- method 1 – Global, Feedback, Pruning (GFP),
- method 2 – Global, Feedback, No Pruning (GFNP),
- method 3 – Global, No Feedback, Pruning (GNFP),
- method 4 – Global, No Feedback, No Pruning (GNFNP),
- method 5 – Local, Feedback, Pruning (LFP),
- method 6 – Local, Feedback, No Pruning (LFNP),
- method 7 – Local, No Feedback, Pruning (LNFP),
- method 8 – Local, No Feedback, No Pruning (LNFNP),
- method 9 – Top 1 Not Yet Examined (NYE), Feedback, Pruning (TFP),
- method 10 – Top 1 NYE, Feedback, No Pruning (TFNP),
- method 11 – Top 1 NYE, No Feedback, Pruning (TNFP), and
- method 12 – Top 1 NYE, No Feedback, No Pruning (TNFNP).

Our **simulated** analysts always correctly vet a link. While in practice this assumption may not always hold, our reasons for making this assumption are quite straightforward. Automated tracing methods and techniques must be built and tested assuming analysts' perfect feedback. If our methods cannot provide better results with perfect feedback, the results will certainly not improve when the feedback is imperfect. Also, we simulate that the analysts can determine when a specific requirement is completely satisfied (for the pruning scenario), for the same reasons.

2.5 Measures

Our focus is on the amount of *analysts' effort* spent on a tracing task. As the direct measure of analysts' effort, we use the number of *observed candidate links* that the analysts must study and accept or reject during the run of the method. We use *precision* and *confirmed recall* (i.e., recall within the observed set of candidate links) to establish the quality of the final mapping produced by the analysts, where:

$$Precision = \text{Hits} / (\text{Hits} + \text{Strikes})$$

$$Recall = \text{Hits} / (\text{Hits} + \text{Misses})$$

with Hits and Misses as defined in Section 2.2 and Strikes as the number of links vetted as False by the analysts (false positives).

3 Analysts' Effort Study

We describe a series of **simulation studies** undertaken to examine the analysts' effort and whether feedback and pruning reduce this effort.

3.1 Research Questions

We explore the following research questions:
Research Question 1 (RQ1) – Does the analysts' method (ordering, feedback, pruning) impact effort? It is our expectation that local methods will outperform Top 1 and global methods, feedback methods will outperform non-feedback methods, and that pruning will outperform non-pruning methods.
Research Question 2 (RQ2) – Is there a point in time when the analysts should switch from vetting candidate links to searching for missing links? It is our expectation that a particular ratio of σ will perform best, regardless of dataset or analysts' method. Such a ratio will assist in the optimal utilization of the analysts' time.
Research Question 3 (RQ3) – Does the method combined with ratio of σ that results in the lowest amount of analysts' effort also yield the best trace matrix in terms of precision and confirmed recall? It is desirable that the minimal effort also yield the highest quality trace matrix, but prior work showed that scenarios exist where the analysts may prefer a method that did not yield the best matrix [6].
Research Question 4 (RQ4) – Does the lowest effort vary based on dataset? It is expected that the effort will vary by dataset.

3.2. Experimental Design

The independent variable is the effort ratio with seven levels (σ set to seven different values) and the analysts' methods with three levels. The dependent variables are confirmed recall, precision, and effort. We examined four datasets (three unique) from different domains, Gantt subset 1, Gantt subset 2, MODIS, and CM-1 subset 1. For each dataset, we applied the twelve methods described in Section 2. We examined actual curves from the traces of these datasets for a variety of σ values: 2, 5, 7, 10, 12, 15, and 20.

As mentioned above, the σ ratios largely depend on both the characteristics of the tracing tasks and the intrinsic qualities of analysts. Thus, predicting an appropriate σ ratio for each dataset is difficult. Instead we choose to sample a fairly large space of possibilities. The σ ratio of 2 corresponds to analysts who require twice the time to discover an error of omission as they need to discover an error of commission. This represents a possibly overly optimistic view of what is possible in practice. On the other end of the spectrum, a σ ratio of 20 corresponds to analysts taking 20 times more time to discover an error of omission than to discover an error of commission. While in practice the ratio can be even larger (a very large tracing task, or very thorough analysts), we felt that an organization that has constraints on the amount of effort to perform a tracing task (see our example above) would not want to operate under a ratio significantly higher than 20: in this scenario, if it took the analysts about 2 minutes to vet a single link (not unreasonable based on our experience with link vetting in some datasets), it would take them 40 minutes to discover a single missing one—quite a large amount of time.

We performed statistical analysis to compare the twelve methods to each other for the individual datasets. **Note that what is called Effort in the following figures refers to σ.**

3.3 Gantt subset 1

Gantt project is an open-source tool for building Gantt charts [15]. The project was broken into two datasets, each consisting of high level requirements and lower level requirements. The Gantt subset 1 dataset has ten high level elements and 35 low level elements (see Table 1), or 350 potential links. We chose to use this for several reasons. Gantt subset 1 is significantly smaller than CM-1 subset 1 (with over 1,100 combinatorial links). Also, it is from a very different domain than is CM-1 subset 1 and MODIS. Gantt subset 1 has small elements in both the low level and high level documents (as compared to CM-1 with very large low level elements). Gantt subset 1's size is similar to MODIS, but with more correct links.

Table 1 also provides information on the initial candidate trace matrix retrieved using VSM with tf-idf weighting as implemented in the RETRO.NET tool [14] (prior to ordering, pruning, etc.): 343 (of the 350) links were retrieved, all 34 true links were retrieved (so perfect recall) but at the price of many false positives (precision of only 0.0991).

Each of the twelve methods was applied to Gantt subset 1. The resulting curves for number of observed links, precision, and confirmed recall were obtained. Confirmed recall vs. observed candidate links for a subset of the methods (no pruning) is shown in Figure 6.

In this case, it can be seen that the Local Feedback No Pruning method slightly outperforms Global Feedback No Pruning. It quickly achieves perfect recall, in 130 observed candidate links, and then levels off. In contrast, the Top 1 methods require far more observed links yet exhibit a more consistent true link discovery behavior.

Figure 7 shows all 12 methods applied to Gantt subset 1. The Global Feedback Pruning method slightly outperforms Local Feedback Pruning. It requires 55 observed candidate links to find all the true links, whereas Local Feedback Pruning requires 57.

Figure 8 shows precision vs. observed candidate links for all 12 methods. Despite some slight variations in precision, it can be seen that all methods except the Top 1 methods with no pruning start with excellent precision and then drop off in a gradual way (as more links are observed, more false positives are examined thus lowering precision).

The effort model was then applied to each curve. Figure 9 shows all σ ratios for the Local No Feedback No Pruning method.

We found that the Top 1 Feedback Pruning method was the method that achieved the higest confirmed recall (of 0.7647, with precision of 0.8387) with the least amount of effort; this was achieved with effort ratio of 2 (calculated effort was 46.68). Effort ratio 2 resulted in the lowest effort required to find the 34 correct links. Other ratios reached effort of 340 before leveling out. We examine Gantt subset 2 next.

3.4 Gantt subset 2

The Gantt subset 2 dataset [15] has seven high level elements and 34 low level elements (see Table 2), or 238 combinatorial links. A distinct subset of the Gantt dataset (from Gantt subset 1) was analyzed and traceability links recovered in order to build this dataset. We chose to use this for the same reasons as listed in Section 3.3 above for Gantt subset 1, and to offer a comparison between datasets.

Table 2 also details the initial candidate trace matrix retrieved: 236 (of the 238) links were retrieved, all 34 true links were retrieved (perfect recall) but so were many false positives (precision of only 0.1441). Figure 10 shows the confirmed recall vs. observed candidate links for a subset of the methods (no pruning).

Figure 11 depicts confirmed recall vs. observed candidate links for all 12 methods. The Local Feedback Pruning method achieves perfect confirmed recall at 115 observed candidate links, just ahead of Global Feedback Pruning at 119 links. The curves for all but Top 1 have a different behavior than for Gantt subset 1: each method has a more gradual increase in confirmed recall, versus the close to vertical slope for Gantt subset 1. The Top 1 methods again have a more consistent slope.

Figure 12 shows precision vs. observed candidate links for all 12 methods. With some slight variations in precision, most of the methods (save two) have a similar behavior. The Top 1 methods with no pruning start with excellent precision and then drop off more rapidly than the other methods.

The effort model was then applied to each curve. Figure 13 shows all σ ratios for the Global No Feedback No Pruning method.

The "winner" is the effort ratio of 2, which has its lowest effort (62.54) at 15 observed links. The precision is 0.6667 and confirmed recall is 0.2941 at that point. Effort 2 ratio was also the winner for Gantt subset 1, but with a much lower minimal effort (46.68). Gannt subset 1 had a much higher quality trace matrix at its lowest effort, particularly in terms of confirmed recall (precision of .8387, confirmed recall .7647). We examine MODIS next.

3.5 MODIS

The NASA Moderate Resolution Imaging Spectroradiometer or MODIS dataset consists of 19 high level requirements and 49 lower level requirements [16] (see Table 3), or 931 combinatorial links.

Table 3 also describes the initial candidate trace matrix retrieved: 908 (of the 931) links were retrieved, all 28 true links were retrieved (perfect recall) as were many false positives (precision of only 0.0308). Figure 14 shows the confirmed recall vs. observed candidate links for a subset of the methods (with pruning).

Figure 14 shows that Global No Feedback Pruning slightly outperforms other methods, requiring 58 observed candidate links to achieve confirmed recall of 1.0, while Top 1 No Feedback Pruning is a close second at 59 observed candidate links.

Figure 15 shows that six of the twelve methods exhibit similar behavior, reaching confirmed recall of 1.0 in the 58 – 60 observed candidate links range. The Top 1 methods with no pruning have the slowest start, moving slowly to confirmed recall of 0.1 and then increasing in a pronounced way.

Figure 16 shows precision vs. observed candidate links for all 12 methods. It is clear that all methods except the Top 1 methods with no pruning start with excellent precision and then drop off quickly (as more links are observed, more false positives are examined thus lowering precision).

Figure 17 shows the effort ratios for Global Feedback No Pruning for MODIS.

Effort ratio 2 is the winner with the lowest effort, 50.72. At that point, confirmed recall is 0.2857 and precision is 0.7273. We examine CM-1 subset 1 next.

3.6 CM-1 subset 1

In this study, we used a subset of CM-1, called CM-1 subset 1. It has 22 high level elements and 53 low level elements (see Table 4), or 1,166 combinatorial links (22 x 53). CM-1 subset 1 is of interest as it is a subset of a larger dataset (on which a similar experiment was run and reported [14]). The gold standard (answer set) for CM-1 subset 1 has recently been improved by several researchers in the traceability community, decreasing its internal threat to validity. CM-1 subset 1 also has larger elements (based on average number of words per element in the low level document) than many datasets.

Table 4 also details the initial candidate trace matrix retrieved: 1,132 (of the 1,166) links were retrieved; all 45 true links were retrieved (perfect recall) as were many false positives (precision of only 0.0398). Confirmed recall vs. effort (observed candidate links) for all the methods is shown in Figure 18.

The curves are very similar to those of MODIS, with the same six methods exhibiting similar behavior, quickly reaching confirmed recall of 1.0. Four methods exhibit almost identical behavior: LFNP, LNFNP, GFNP, and GNFNP. As with MODIS, TFNP and TNFNP again are the slowest to rise to perfect confirmed recall, and have a more gradual slope than the other methods.

Next, we examined the σ ratios. Figure 19 shows ratio $\sigma = 2$ for all twelve methods.

To examine the curves more closely, Figures 20, 21, and 22 depict the $\sigma = 2$ curves for the Global methods, Local methods, and Top 1 methods, respectively. As can be seen, all methods using Pruning terminate more quickly than non-pruning methods (regardless of Local, Top 1 Not yet Examined (NYE), or Global). It appears that feedback methods may have resulted in less effort.

We examined the number of observed links and the number of true links seen for the minimum effort for each method for effort ratio 2 (the lowest for each method), see Table 5. The minimum effort values for each method were very similar. The local methods, regardless of feedback or pruning, examined 22 links at their lowest effort points, finding 13 true links for precision of 0.5909 and confirmed recall of 0.2889. GFP had the same values as the local methods. GFNP was the "winner" with lowest effort; it had confirmed recall of 0.2 and precision of 0.818. Other methods had slightly higher precision of 1.0, but had lower confirmed recall (lower than 0.1). Several methods had higher confirmed recall, 0.2889, but with much lower precision of 0.5909.

3.7 Statistical Analysis

Statistical analysis, using the student's t-test, was undertaken for the various measures, between methods. All assumptions of the student's t-test were met by the data and we used $\alpha = 0.05$. An exemplary table for the MODIS dataset and precision measure is shown in Table 6.

As can be seen in Table 6, in general, the local methods are statistically different from the Top 1 and global methods. The Top 1 methods with pruning are statistically different from all other methods. The Top 1 methods with no pruning are statistically significantly different from the pruning Top 1 and global methods. The global pruning methods have the same results as the Top 1 pruning methods; the global non pruning methods have the same results as the Top 1 non pruning methods.

3.8 Threats to Validity

The research is subject to a number of threats to validity. Threats to internal validity include the potential bias introduced by the use of a gold standard (the answer sets associated to each dataset). First, it is possible that the answer set is not correct. Second, it is possible that as builders of the answer set, we may have inadvertently biased the results. To mitigate these threats, we used datasets such as CM-1 subset 1, which have been used by the traceability community. The community has suggested, and we have

accepted, changes to the answer set for CM-1 subset 1. The MODIS dataset has also been in use for many years in the community and many research groups examined this answer set. The Gantt datasets were developed for use in tracing and satisfaction research, at least two research groups have examined their answer sets.

We mitigated threats to construct validity by using standard measures found in Information Retrieval and traceability studies. We mitigated threats to conclusion validity by undertaking statistical analysis and ensuring that the assumptions required of the statistical test were met.

A threat to external validity is that the datasets are not representative of datasets in use in reality. To address this threat, we used three different datasets. The datasets were of reasonable size, with the CM-1 subset 1 dataset consisting of over 1,100 possible links. Another possible external threat to validity is that of simulating perfect analysts. We argued that if a feedback method cannot perform well given perfect feedback, it will certainly not perform well given imperfect feedback. Hence, we believe that it makes sense to begin studies of analysts' effort by working with perfect feedback.

A further threat to external validity relates to the applicability of our model to other domains. Our work examined the domain of requirements tracing, specifically modeling the interaction of an analyst with a candidate list of links between requirement artifacts. Our understanding of this problem, our understanding of how analysts work with candidate links, and our modeling of this problem are all rooted in the requirements engineering domain. Such an effort model may look very different for another domain such as Human-Computer Interaction (HCI).

4 Results

We now examine each of the research questions and provide answers based on the study above and considering the threats to validity described previously.

Research Question 1 (RQ1) – Does the analysts' method (ordering, feedback, pruning) impact effort?

Based on our study, the answer to research question 1, is Yes: the analysts' method impact effort. Table 7 presents the "winning" minimum effort value for each analyst's method and for each dataset examined. The first column lists the method. Each dataset is represented by the subsequent columns, depicting the lowest effort value achieved for that method and that dataset at the ratio of σ that is shown in parentheses next to the dataset name. For example, for CM-1 subset 1 for GFNP, the minimum effort was 82.32 and the ratio of σ was 2. The final column of the table indicates the lowest effort (winner) and highest (of the minimum effort values) or loser for each dataset—the method is a winner unless specifically noted as loser. We found two winners. Two methods had the lowest effort for two datasets: the GFNP method provided the lowest effort for CM1 subset 1 and tied for Gantt subset 2, the TFP method was the best for Gantt subset 1 and MODIS. One other method was a winner: the GFP method had the lowest effort for Gantt subset 2 (tie). TNFNP was the loser most often, with the highest (minimum) effort for Gantt subset 1 and tied as the loser for CM1 subset 1 and Gantt subset 2.

Examining the winner and loser methods more closely, we first look at the analysts' effort required to achieve a fixed recall. As the retrieved candidate matrix for all four datasets had all the true links in it, we set the recall at 1.0.

Table 8 lists the two winner methods (GFNP and TFP) as well as the loser method (TNFNP) in column 1, the number of confirmed true links in column 2, the number of observed candidate links in column 3, the precision in column 4, and the dataset in column 5. The TFP method required the least number of observed links for all datasets and also achieved the best precision. For example, for Gantt subset 1, TFP required the least analysts' effort, requiring only 57 observed links to find the 34 true links (precision of 0.5965).

Next, we fix the number of observed candidate links for the winner and loser methods and examine the recall achieved for this amount of analysts' effort. We set the observed links to be the lowest number required to achieve recall of 1.0 (thus it varies by dataset). Table 9 lists the three methods in column 1, the number of observed links in column 2, the number of confirmed true links in column 3, the recall in column 4, the precision in column 5, and the dataset in column 6. The TFP method is the overall winner, achieving perfect recall and the highest precision for all datasets. The GFNP method is not far behind TFP for the three smaller datasets (all but CM1 subset 1). For example, for Gantt subset 1, the confirmed recall is 0.8824 compared to 1.0 for TFP, and the precision is 0.5263 compared to 0.5965. The TNFNP method performs poorly, not even discovering any true links for some datasets.

Research Question 2 (RQ2) – Is there a point in time when the analysts should switch from vetting candidate links to searching for missing links?

The answer to this question appears to be Yes. Looking again at Table 7, we see that the effort ratio that consistently provides the lowest effort is effort ratio 2. For example, for the TNFNP method (which was the overall loser) for CM1 subset 1, the minimum effort was 85.72 for effort ratio 2, 160.33 for effort ratio 5, 198.02 for effort ratio 7, 245.17 for effort ratio 10, 267.69 for effort ratio 12, 290.62 for effort ratio 15, 328.83 for effort ratio 17, and 328.83 for effort ratio 20. Comparing the minimum effort at effort ratio 2 to that of effort ratio 20, 383% more effort is required when the ratio is 20.

Research Question 3 (RQ3) – Does the method combined with ratio of σ that results in the lowest amount of analysts' effort also yield the best trace matrix in terms of precision and confirmed recall?

The answer to this question appears to be Not always. The precision and confirmed recall at the minimum effort point for each dataset were not always the best. Tables 7 through 9 show that, though the GFNP method had the lowest effort for some datasets, the TFP method was the winner for all datasets in terms of matrix quality at the lowest effort point.

Research Question 4 (RQ4) – Does the lowest effort vary based on dataset?

The answer to this research question is Yes. The student's paired t-test was applied with a one tailed distribution (the null hypothesis is equal variance between dataset 1 and dataset 2, the alternative hypothesis is that the variances are not equal; we use $\alpha = 0.05$), comparing the lowest effort value for each of the 12 methods for each dataset to each other. All required assumptions of the student's t-test were met by the data. Table 10 shows the p-values for each comparison, bolded items are statistically significant. For example, the minimum effort values of Gantt subset 1and Gantt subset 2 are statistically significantly different, with a p-value of 0.006. The CM-1 subset 1 and Gantt subset 2 datasets exhibit the greatest difference in minimum effort, with a p-value of 6.6E-14.

5 Related Work

Though the generation of mappings is a general problem, it has been investigated for software engineering in the area of requirements traceability or requirements tracing[4] and defined as: "the ability to follow the life of a requirement in a forward and backward direction [1]." Much work to date has concentrated on the recovery or generation of traceability links between software engineering artifacts (structured as well as non-structured artifacts).

Antoniol et al. [17] applied the vector space model (also known as term frequency-inverse document frequency [8]) to the problem of recovering traceability links between a textual user's manual and source code and between textual functional requirements and source code. They achieved high recall values (93% to 100%), but were only able to achieve 13% to 18% precision. Antoniol et al. [18] also applied a probabilistic method to the problem of recovering links between source code and documentation. Though high precision was achieved (83%), it was at the price of recall (39%). For the purposes of IV&V, recall must be high (typically 90% or higher) [2].

Marcus and Maletic [19] applied latent semantic indexing (LSI) to the problem of recovering traceability links between documentation and source code (using the same dataset as Antoniol et al. [17]) and found that LSI performs at least as well as the vector space model while requiring less pre-processing of the artifacts. They achieved recall of 91% to 100% and precision of 13% to 18%. When they relaxed recall to 71%, they achieved precision of 43%. Cleland-Huang et al. developed a method for dynamically

[4] Note that "requirements tracing" is often the moniker even when requirements are not being traced

generating traceability data in a speculative manner for performance models that may be affected by a proposed change [20]. Links were established and maintained between the performance models and key requirements data that had been derived from the performance models.

The current study focuses on the analysts' behavior in requirements tracing. In earlier work [7], we undertook a study to compare analysts: (1) performing tracing manually, (2) using a keyword-based technique, (3) using the output from a keyword-based technique, and (4) using our IR technique. The results showed that the analysts and the keyword-based technique achieved 63% recall and 39% precision, while using our technique achieved 85% recall and 40% precision. Next, we developed the requirements for a requirements tracing technique [13] and found that a number of the requirements had to do with the analysts, specifically the utility sub-requirement of believability (the analysts feels that the technique is useful), the communicability sub-requirement of discernability (the technique provides information, process flow, and the results to the analysts in an understandable way), and the final requirement of endurability (the technique makes the tracing task as pleasant as possible).

In later work, we introduced the notion of analysts' feedback. Here, the analysts provide feedback on the top N elements of each candidate link list (Yes, this is a link, No, this is not a link) and the feedback is used to modify the vectors for the high-level elements before re-executing the matching algorithm. Using feedback, we improved recall to close to 90% with precision close to 80% (for the MODIS dataset) [2].

Our first study of the analysts' effort was undertaken using a small dataset (MODIS) and a number of requirements traceability matrices [10]. Some RTMs had low recall but high precision, some had low precision but high recall, and varied combinations in between. It was our belief that analysts would make better feedback decisions if given a high quality RTM (high recall and high precision), but would make bad decisions if given poor quality RTMs. What we observed was that all analysts made bad decisions (struck true links and kept false positives). The study was very small, however, and we could not draw general conclusions.

Next, we undertook a small study with graduate students and found that those using our IR technique (called Requirements Tracing On-target or RETRO) achieved statistically significantly higher recall (70.1% versus 33% for the group using no automated technique) but lower precision (12.8% versus 24.2%), but took far less time to complete the tracing task (three times less minutes (41.8 minutes versus 120.66) [21]. The students were tracing textual requirements to textual design elements. We also conducted a small usability survey was also conducted. Students found the RETRO features that they used to be very useful. Students used most of the features available to them but may have misunderstood of the feedback.

Building on our study of the analysts' effort [10], Cuddeback et al. undertook a study at two Universities with 26 students using a "fake" version of RETRO (the technique was not applying IR algorithms to recover links but rather was just loading a table of pre-calculated links) to trace 32 requirements to 17 test cases for a Java style checker [11]. Again, traceability matrices of varying quality levels were given to the participants. We observed that half the students made the results worse and half did not. We observed that the results varied based on the quality of the initial traceability matrix (high recall, low precision quadrant; high recall, high precision quadrant; etc.) There was no correlation between the quality of the trace and the analysts' effort [11]. Also, there was no significant difference based on University [11].

Egyed et al. found that manual tracing can be applied in a time-effective manner when building traces from textual requirements to source code in a constrained problem (participants were given a subset of source code known to match to the requirements being traced) [22]. They undertook an experiment using 100 students who examined textual requirements and traced either to classes or methods for GanttProject (17 requirements traced to either 85 classes or 788 methods) or ReactOS (open-source implementation of Windows NT architecture) (16 requirements traced to either 123 classes or 544 methods) [22]. They found that code complexity increases trace recovery effort as does tracing to methods versus classes. They also found that the amount of time spent on a trace link was not necessarily correlated with correctness: "A higher tracing effort does not imply better quality. Data indicates that trace link recovery falls into two categories: fast and accurate or slow and inaccurate. [22]"

Recommender systems fall broadly into two categories: content-based recommenders focus on textual descriptions of products; collaborative recommenders make recommendations based on the preference of nearest-neighbors (or similar users) [24, 25]. Collaborative-filtering techniques often outperform content-based ones; however, they can only be employed after sufficient users have indicated their preferences for a product (leading to the cold-start problem) [24, 25]. Interactive recommender systems "usually rely on an online learning algorithm that gradually learns users' preferences. At each step of the interaction, the system generates a list of recommendations and observes the user's feedback on the recommended items

indicating the utility of the recommendations. The goal of such a system is to maximize the total utility obtained over the whole interaction session [26]." Bostandjiev et al. introduce TasteWeights, an interactive recommender for media content listed on Facebook such as movies, books, and TV shows [27]. TasteWeights uses a visual representation to both explain the recommendation process and to interact with the user to obtain preferences. The user is able to change the weight of an item, remove items, navigate context sources, and more. The authors found that explaining a recommendation process can increase user satisfaction and that recommendation accuracy and user experience was improved (per a user questionnaire) as a result of the provided interactivity [27]. Hariri et al. examine the importance of context in interactive recommender systems. Specifically, they designed a system to detect and adapt to changes in context based on the user's ongoing behavior by extending bandit algorithms to consider sudden variations in the user's feedback behavior as a result of contextual changes. They applied their system to two datasets (movie recommendations and college web sessions) and showed that their system performed at least as well as baseline algorithms [27].

The current work differs from the previous work in several ways. First, it specifically focuses on the analysts' behavior. Second, it examines a number of scenarios under which the analysts "navigate" the generated traceability links, *without* requiring real analysts. Third, it examines analysts' effort when using 12 different methods of working with a TM. Fourth, the methods have been applied to three datasets. Fifth, it provides guidance on how to best work with feedback-driven automated information retrieval tracing techniques. Finally, it does not suffer from the cold start problem in that even feedback on one candidate link from one analyst can be used to re-run the linking algorithms. Visualizations used by recommender systems to interact with users could be applied to help design better interfaces for trace link generation tools.

6 Conclusions and Future Work

We posited a theoretical and simplified model for analysts' effort, in which the simulated analysts take perfect decisions on the links that they review. Based on that model, we used the ratio of the amount of time to find a link from scratch to the amount of time to recognize a retrieved true link as the effort ratio to study the analysts' effort required when working with candidate links generated by an automated technique. We simulated 12 analysts' methods for examining links, varying ordering, feedback, and pruning. We varied the effort ratio from 2 to 20 (by steps of 3). We examined the effort and precision and confirmed recall for the various analysts' methods and effort ratios on four datasets.

We found that the analysts' method does impact the effort. In 48 scenarios (12 methods on four datasets), the Global Feedback No Pruning (GFNP) method and Top 1 Feedback Pruning (TFP) methods resulted in the lowest effort two times[5]. The "worst" method appeared to be Top 1 No Feedback No Pruning (TNFNP), which was the "loser" in three of the 48 scenarios. While GFNP and TFP were the winners more often than any other methods, they did not always yield the highest quality in terms of precision and confirmed recall at the lowest effort point. The TFP method was the overall winner in terms of quality, though GFNP had lower effort for some datasets. The datasets did impact the results. The lowest effort varied by dataset in a statistically significant way. Though perfect recall was always achieved, the precision and effort varied widely by dataset.

We make the following observations. For No Feedback methods, the Global, Local, and Top 1 represent three different ways of reordering and visiting the retrieved candidate links. The Top 1 Not Yet Examined No Feedback method orders candidate links the same way as the Local No Feedback method, but does not remove any candidate links from consideration. When all retrieved links are examined, Top 1 and Global methods give the same results – the links were simply reordered. As long as all are examined, the same precision/recall values at the end will be achieved. For local methods, there is a small number of observed links, because of the filtering condition. So, recall is the same, while precision is somewhat better. In general, it appeared that the two Top 1 Not Yet Examined methods with no pruning (regardless of feedback option) took the longest to reach perfect confirmed recall for all the datasets. The fastest methods to reach 100% confirmed recall were the pruned methods, which is not so surprising: assuming that candidate links

[5] It should be noted that many of the 48 scenarios resulted in ties for minimum effort value, thus there were far less than 48 scenarios that were "in the running" for best and worst.

with higher similarity scores are more likely to be true links, a Global method is a better way to reach all of them than the Top 1 method.

When examining the winner and loser methods at fixed effort and fixed recall, we found that the TFP method was the overall winner. It required less observed links to achieve recall of 1.0, with higher precision than the GFNP and TNFNP methods. For all of the datasets except for CM1 subset 1, the GFNP method was not far behind the TFP method in terms of observed links required to achieve perfect recall and precision. The TNFNP method, however, performed very poorly. For some datasets, that method had not yet found the first true link within the fixed number of observed candidate links.

Based on our results, when analysts are deciding how to configure the automated technique and how to work with the candidate link list, we advise them to:

- use feedback,
- use the Top 1 Not Yet Examined method, and
- use pruning.

In general, this combination was the overall winner in terms of lowest effort and highest quality compared to other methods.

For future work, we will improve our log RETRO.NET (a version of RETRO.NET that logs analysts' keystrokes and decisions) to support various ordering, feedback, and pruning methods and allow the analysts to select a user interface that presents the selected method. This improvement will enable the simulation of other behaviors/methods, such as accept-focused, preview, and iterative. Such improvement will help us better understand what cognitive patterns analysts may have before, during, and after tracing. Interactive recommender systems may provide inspiration for new and better visualizations and interfaces to elicit feedback from analysts. We will also consider imperfect simulated analysts, which do not always make the correct decision when examining a link. We are also working on applying machine learning to the problem of determining what methods may work best for a given dataset. Our findings to date provide guidance, but the results may not generalize and must be applied to never-before-seen datasets. Thus, we must discover the underlying attributes or characteristics of a dataset that indicate how well a tracing method or analysts' method may or may not perform on it. These attributes and characteristics depend on the words contained in the elements composing the considered artifacts but also on other information contained therein, such as UML class diagrams, relations embodied in tables, and so on. We will consider how to extract and use this information in a dedicated line of research work. Finally, because we ultimately help human analysts in making the best use of their time when reviewing candidate links, we plan to perform case studies and experimental studies with real human analysts to measure the benefit brought by the improvement and guidance derived from this work.

References

[1] O. Gotel and A. Finkelstein, "Extended Requirements Traceability: Results of an Industrial Case Study," *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, IEEE Computer Society, 1997, p. 169.

[2] J.H. Hayes, A. Dekhtyar, and S. Sundaram, "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods," *IEEE Transactions on Software Engineering*, vol. 32, 2006, pp. 4 - 19.

[3] U.S. Department of Health and Human Services, Food and Drug Administration, "General Principles of Software Validation; Final Guidance for Industry and FDA Staff," Jan. 2002.

[4] S. Sundaram, J. Hayes, A. Dekhtyar, and E. Holbrook, "Assessing traceability of software engineering artifacts," *Requirements Engineering*, vol. 15, 2010, pp. 313-335.

[5] J. Cleland-Huang, A. Dekhtyar, J.H. Hayes, G. Antoniol, B. Berenbach, A. Egyed, S. Ferguson, and J. Maletic, "Grand Challenges in Traceability," Sep. 2006.

[6] A. Dekhtyar, J.H. Hayes, and M. Smith, "Towards a Model of Analyst Effort for Traceability Research: A Position Paper," *Proceedings of Traceability of Emerging Forms of Software Engineering (TEFSE)*, 2011.

[7] J.H. Hayes, A. Dekhtyar, and J. Osborne, "Improving Requirements Tracing via Information Retrieval," *International Conference on Requirements Engineering, Monterey, California*, 2003, pp. 151-161.

[8] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, ACM Press, Addison-Wesley, 1999.

[9] J.H. Hayes, A. Dekhtyar, and S. Sundaram, "Improving After the Fact Tracing and Mapping to Support Software Quality Predictions," *IEEE Software*, vol. 22, 2005, pp. 30-37.

[10] J.H. Hayes and A. Dekhtyar, "Humans in the traceability loop: can't live with 'em, can't live without 'em," *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*,

New York, NY, USA: ACM, 2005, pp. 20–23.
[11] D. Cuddeback, A. Dekhtyar, and J. Hayes, "Automated Requirements Traceability: The Study of Human Analysts," *Requirements Engineering, IEEE International Conference on*, vol. 0, 2010, pp. 231-240.
[12] "CM1 DataSet,Metrics Data Program Website, CM-1 Project."
[13] J.H. Hayes, A. Dekhtyar, S. Sundaram, and S. Howard, "Helping Analysts Trace Requirements: An Objective Look," *International Conference on Requirements Engineering (RE'2004)*, 2004.
[14] A. Dekhtyar, J.H. Hayes, and J. Larsen, "Make the Most of Your Time: How Should the Analyst Work with Automated Traceability Tools?," *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, Washington, DC, USA: IEEE Computer Society, 2007.
[15] "GanttProject Home."
[16] G. SBRS, *MODIS Science Data Processing Software Requirements Specification Version 2*.
[17] G. Antoniol, G. Canfora, G. Casazza, A.D. Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," *IEEE Transactions on Software Engineering*, vol. 28, 2002, pp. 970-983.
[18] G. Antoniol, B. Caprile, A. Potrich, and P. Tonella, "Design-Code Traceability for Object Oriented Systems," *Annals of Software Engineering*, vol. 9, 1999, pp. 35–58.
[19] A. Marcus and J. Maletic, "Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing," *Proceedings of the Twenty-Fifth International Conference on Software Engineering 2003*, 2003, pp. 125-135.
[20] J. Cleland-Huang, C. Chang, G. Sethi, K. Javvaji, H. Hu, and J. Xia, "Automating speculative queries through event-based requirements traceability," *Proceedings of the IEEE Joint International Requirements Engineering Conference (RE '02)*, 2002, pp. 289-296.
[21] J. Hayes, A. Dekhtyar, S. Sundaram, E. Holbrook, S. Vadlamudi, and A. April, "REquirements TRacing On target (RETRO): improving software maintenance through traceability recovery," *Innovations in Systems and Software Engineering*, vol. 3, Sep. 2007, pp. 193-202.
[22] A. Egyed, F. Graf, and P. Grunbacher, "Effort and Quality of Recovering Requirements-to-Code Traces: Two Exploratory Experiments," *Requirements Engineering, IEEE International Conference on*, vol. 0, 2010, pp. 221-230.
[23] Wei-Keat Kong; Hayes, J.H.; Dekhtyar, A.; Dekhtyar, O., "Process improvement for traceability: A study of human fallibility," Requirements Engineering Conference (RE), 2012 20th IEEE International, pp.31,40, 24-28 Sept. 2012
[24] Z. Zhang, C. Liu, Y. Zhang, and T. Zhou. Solving the cold-start problem in recommender systems with social tags. CoRR, abs/1004.3732, 2010.
[25] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades. Facing the cold start problem in recommender systems. Expert Systems with Applications, 41(4, Part 2):2065 { 2073, 2014.
[26] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. TasteWeights: a visual interactive hybrid recommender system. In Proceedings of the sixth ACM conference on Recommender systems (RecSys '12). ACM, New York, NY, USA, 35-42.
[27] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2014. Context adaptation in interactive recommender systems. In Proceedings of the 8th ACM Conference on Recommender systems(RecSys '14). ACM, New York, NY, USA, 41-48.
[28] Nasir Ali, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links. Transactions on Software Engineering (TSE), 39(5):725--741, May 2013.
[29] Axiom, http://www.iconcur-software.com/, last accessed June 29, 2016.

Table 1 Gantt subset 1 dataset overview

Dataset Name	Gantt subset 1
# elements in requirements document (high-level)	10
# elements in design document (low-level)	35
# correct links	34
Total number of retrieved candidate links	343
Total number of correct links retrieved	34
Recall	1.0
Precision	0.0991

Table 2 Gantt subset 2 dataset overview

Dataset Name	Gantt subset 2
# elements in requirements document (high-level)	7
# elements in design document (low-level)	34
# correct links	34
Total number of retrieved candidate links	236
Total number of correct links retrieved	34
Recall	1.0
Precision	0.1441

Table 3 MODIS dataset overview

Dataset Name	MODIS
# elements in requirements document (high-level)	19
# elements in design document (low-level)	49
# correct links	28
Total number of retrieved candidate links	908
Total number of correct links retrieved	28
Recall	1.0
Precision	0.0308

Table 4 CM-1 subset 1 dataset overview

Dataset Name	CM-1 subset 1
# elements in requirements document (high-level)	22
# elements in design document (low-level)	53
# correct links	45
Total number of retrieved candidate links	1132
Total number of correct links retrieved	45
Recall	1.0
Precision	0.0398

Table 5 Effort ratio = 2 for σ – C1 Subset 1

Method	Min. effort	#observed@mineffort	#trueobserved@mineffort	Confirmed recall	Precision
GFNP	82.320755	11	9	0.2	0.81818
GFP	84.792453	22	13	0.2889	0.590909
GNFNP	83.365352	10	8	0.1778	0.8
GNFP	82.428816	9	8	0.1778	0.88889
TFNP	82.543739	7	7	0.1556	1
TFP	83.598628	6	6	0.1333	1
TNFNP	85.718696	4	4	0.0889	1
TNFP	85.718696	4	4	0.0889	1
LFNP	84.792453	22	13	0.2889	0.590909
LFP	84.792453	22	13	0.2889	0.590909
LNFP	84.792453	22	13	0.2889	0.590909
LNFNP	84.792453	22	13	0.2889	0.590909

Table 6 T-test for 12 methods applied to MODIS - Precision

Method's Precision Value	**Is Statistically Significantly Different Than:**
Local, No Feedback, Pruning (LNFP)	TNFP,TFP,GFP,GNFP
Local, No Feedback, No Pruning (LNFNP)	TNFP,TFP,GFP,GNFP
Local, Feedback, Pruning (LFP)	TNFP,TFP,GFP,GNFP
Local, Feedback, No Pruning (LFNP)	TNFP,TFP,GFP,GNFP
Top1 NYE, No Feedback, Pruning (TNFP)	LNFP,LNFNP,LFP,LFNP,TFNP,TNFNP,GFNP,GNFNP
Top1 NYE, No Feedback, No Pruning(TNFNP)	TNFP,TFP,GFP,GNFP
Top1 NYE, Feedback, Pruning (TFP)	LNFP,LNFNP,LFP,LFNP,TFNP,TNFNP,GFNP,GNFNP
Top1 NYE, Feedback, No Pruning (TFNP)	TNFP,TFP,GFP,GNFP
Global, No Feedback, Pruning (GNFP)	LNFP,LNFNP,LFP,LFNP,TFNP,TNFNP,GFNP,GNFNP
Global, No Feedback, No Pruning (GNFNP)	TNFP,TFP, GFP, GNFP
Global, Feedback, Pruning (GFP)	LNFP,LNFNP,LFP,LFNP,TNFNP,TNFNP,GFNP,GNFNP
Global, Feedback ,No Pruning (GFNP)	TNFP,TFP, GFP, GNFP

Table 7 Minimum effort value for each method and dataset

	CM1S1 (2)	*Gannt1 (2)*	*Gantt 2 (2)*	*MODIS (2)*	*Winner/loser*
GFNP	82.32	54.57	60.53	50.72	CM1S1, tied on Gantt2
GFP	84.79	48.72	60.53	43.76	Tied on Gantt2
GNFNP	83.37	59.56	62.55	50.72	
GNFP	82.43	59.44	62.55	47.70	
TFNP	82.54	65.77	64.88	56.96	
TFP	83.60	46.69	64.88	39.75	Gantt1, MODIS
TNFNP	85.72	68.96	65.92	56.96	Tied loser CM1S1, loser Gantt1, tied loser Gantt2
TNFP	85.72	67.48	65.92	43.71	Tied loser CM1S1, tied loser gantt2
LFNP	84.79	53.55	63.56	62.48	Tied loser MODIS
LFP	84.79	49.88	63.56	56.00	
LNFP	84.79	59.50	63.56	56.00	
LNFNP	84.79	59.50	63.56	62.48	Tied loser MODIS

Table 8 Observed candidate links and precision at fixed recall of 1.0 for “Winner” and “Loser” methods

Method	Confirmed True Links	Observed Candidate Links	Precision	Dataset
GFNP	34	111	0.3063	Gantt subset 1
TFP	34	57	0.5965	Gantt subset 1
TNFNP	34	313	0.1086	Gantt subset 1
GFNP	34	158	0.2512	Gantt subset 2
TFP	34	115	0.2957	Gantt subset 2
TNFNP	34	217	0.1567	Gantt subset 2
GFNP	28	329	0.0851	MODIS
TFP	28	71	0.3944	MODIS
TNFNP	28	622	0.045	MODIS
GFNP	45	662	0.068	CM1 subset 1
TFP	45	143	0.3147	CM1 subset 1
TNFNP	45	1082	0.0416	CM1 subset 1

Table 9 Confirmed recall and precision at fixed number of observed candidate links

Method	Observed Links	Confirmed True Links	Confirmed Recall	Precision	Dataset
GFNP	57	30	0.8824	0.5263	Gantt subset 1
TFP	57	34	1.0	0.5965	Gantt subset 1
TNFNP	57	7	0.2059	0.1228	Gantt subset 1
GFNP	115	26	0.7647	0.2261	Gantt subset 2
TFP	115	34	1.0	0.2957	Gantt subset 2
TNFNP	115	21	0.6176	0.1826	Gantt subset 2
GFNP	71	23	0.8214	0.3239	MODIS
TFP	71	28	1.0	0.3944	MODIS
TNFNP	71	0	0	0	MODIS (note that first match found at link 145)
GFNP	143	28	0.6222	0.1958	CM1 subset 1 (143 links)
TFP	143	45	1.0	0.3147	CM1 subset 1 (143 links)
TNFNP	143	12	0.2667	0.0839	CM1 subset 1 (143 links)

Table 10 Student's Paired T-test, One-tailed for Minimum Effort

Dataset	CM-1 subset 1	Gannt1	Gantt2	MODIS
CM-1 subset1	NA	**3.45E-08**	**6.60E-14**	**5.41E-09**
Gantt1	**3.45E-08**	NA	**0.006**	**0.027**
Gantt2	**6.60E-14**	**0.006**	NA	**0.000**
MODIS	**5.41E-09**	**0.027**	**0.000**	NA

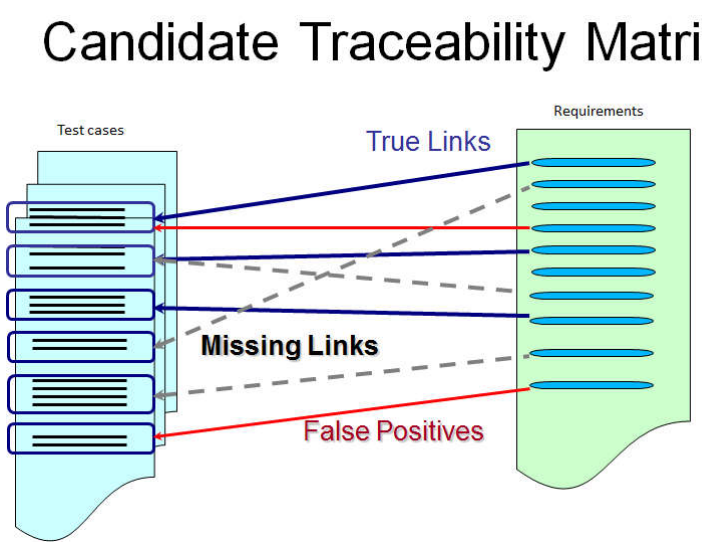

Fig 1 Tracing scenario: requirements to test plan

HighDocs	LowDocs	Weight
SDP3.3-4	L1APR01-I-1	0.868215278
SDP3.3-4	L1APR01-F-2.2.3-4	0.102804641
SDP3.3-4	L1APR01-F-4-3	0.083900716
SDP3.3-4	L1APR01-F-2.1-4	0.081284136
SDP3.3-4	L1APR03-F-1-2	0.078608219
SDP3.3-4	L1APR03-I-5	0.06709611
SDP3.3-4	L1APR03-F-3.2.1-2	0.055489017
SDP3.3-4	L1APR01-F-2.2.4-2	0.055132765
SDP3.3-4	L1APR01-F-2.1-1	0.051063439
SDP4.2-2	L1APR01-F-4-3	0.649469866

TM candidate links, local ordering

HighDocs	LowDocs	Weight
SDP3.3-4	L1APR01-I-1	0.868215278
SDP4.2-2	L1APR01-F-4-3	0.649469866
SDP3.3-4	L1APR01-F-2.2.3-4	0.102804641
SDP3.3-4	L1APR01-F-4-3	0.083900716
SDP3.3-4	L1APR01-F-2.1-4	0.081284136
SDP3.3-4	L1APR03-F-1-2	0.078608219

TM candidate links, global ordering

Fig 2 Global versus Local Ordering

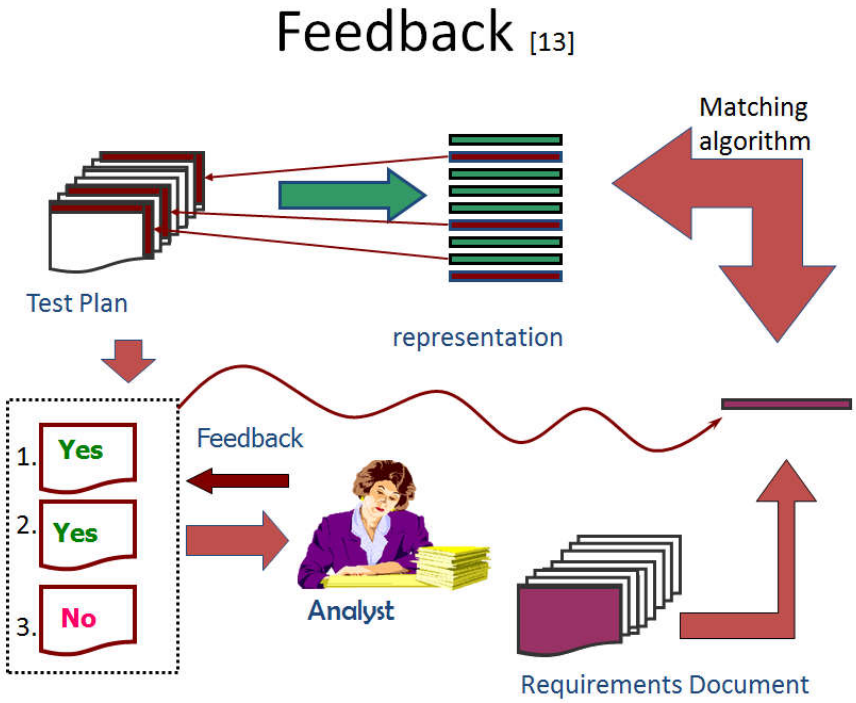

Fig 3 Feedback

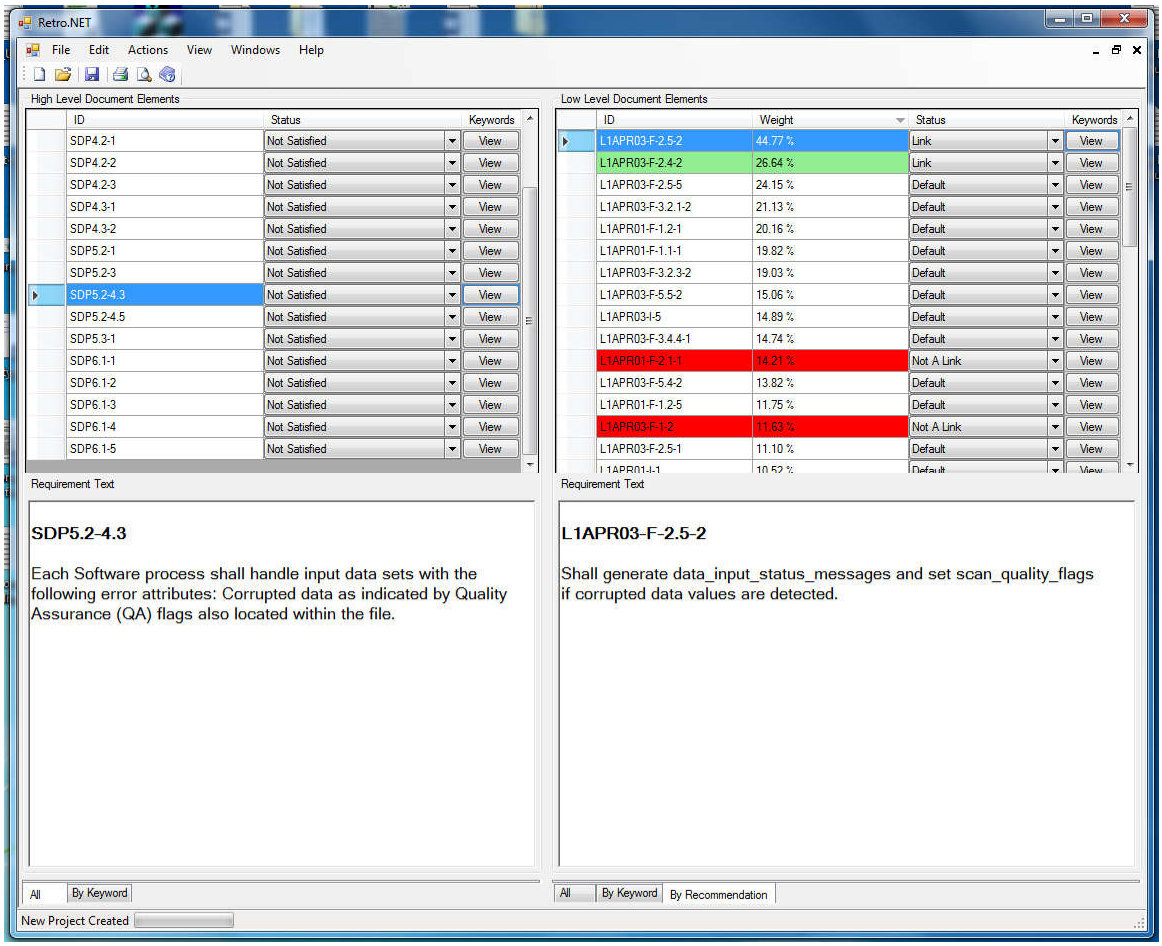

Fig 4 RETRO.NET screenshot of feedback.

HighDocs	LowDocs	Weight
SDP3.3-4	L1APR01-I-1	0.868215278
SDP3.3-4	L1APR01F-2.2.3-4	0.102804641
SDP3.3-4	L1APR01-F-4-3	0.083900716
SDP3.3-4	L1APR01-F-2.1-4	0.081284136
SDP3.3-4	L1APR03-F-1-2	0.078608219
SDP3.3-4	L1APR03-I-5	0.06709611

.
.
.

Fig 5 Pruning

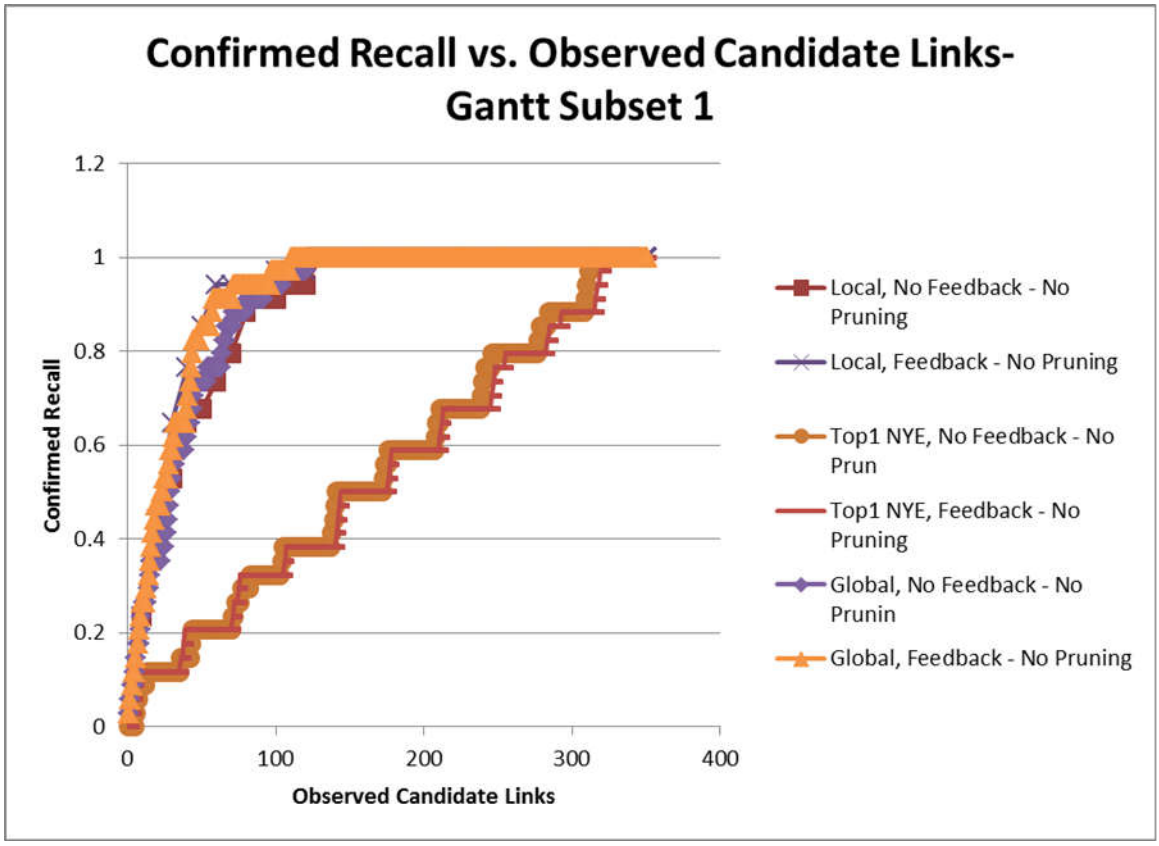

Fig 6 Confirmed Recall vs. Observed Candidate Links for Gantt Subset 1 – No Pruning

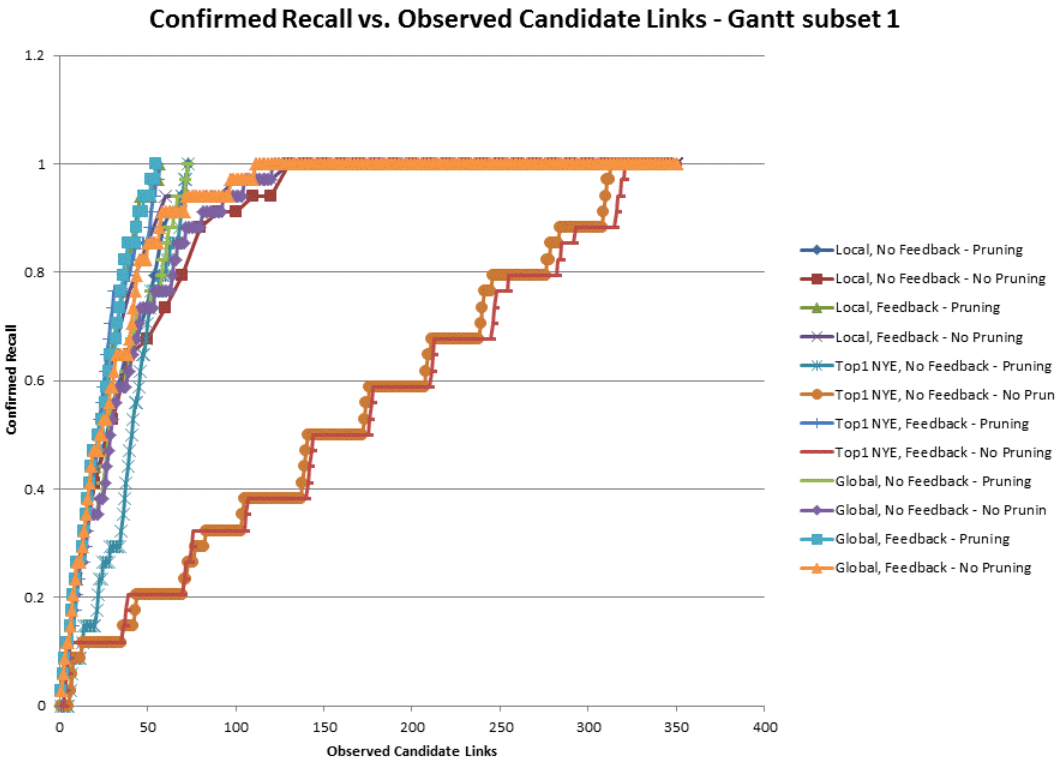

Fig 7 Confirmed Recall vs. Observed Candidate Links for Gantt Subset 1 – All Methods

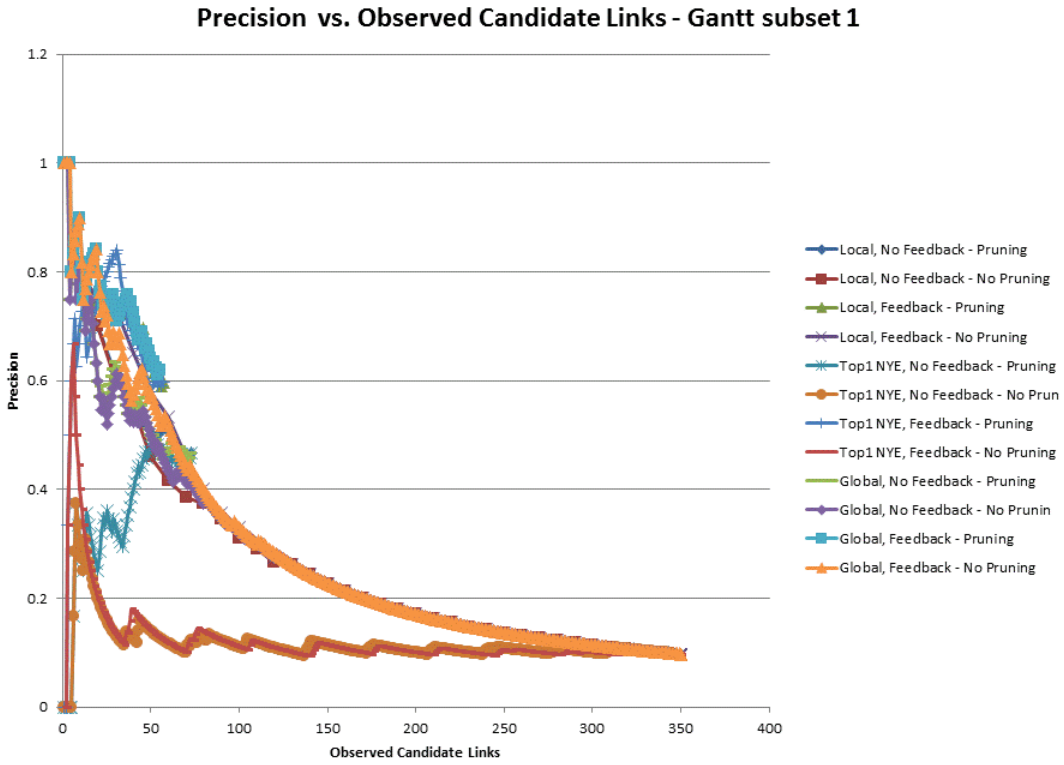

Fig 8 Precision vs. Observed Candidate Links for Gantt Subset 1 – All Methods

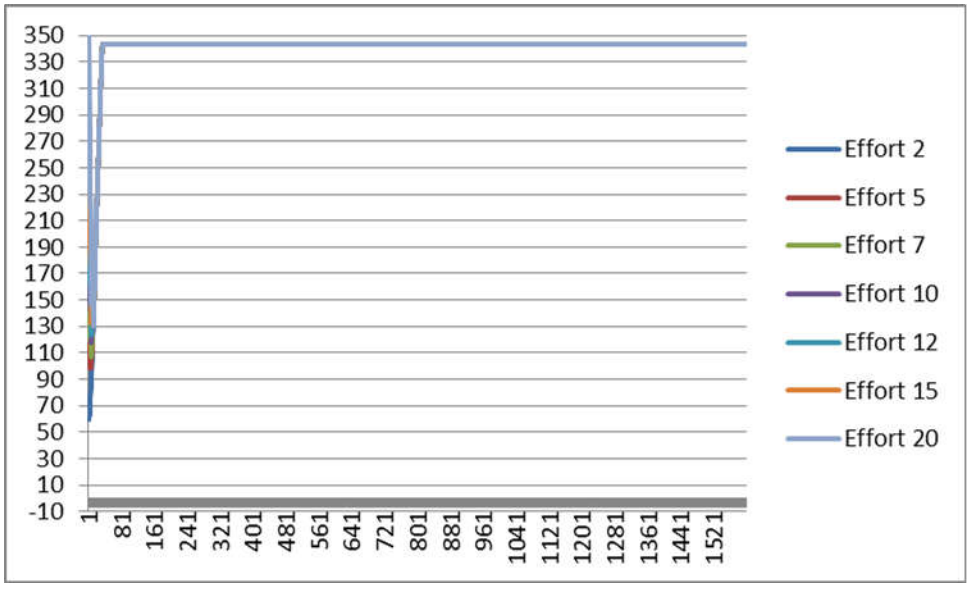

Fig 9 Local No Feedback No Pruning, all effort ratios (effort vs. observed candidate links)

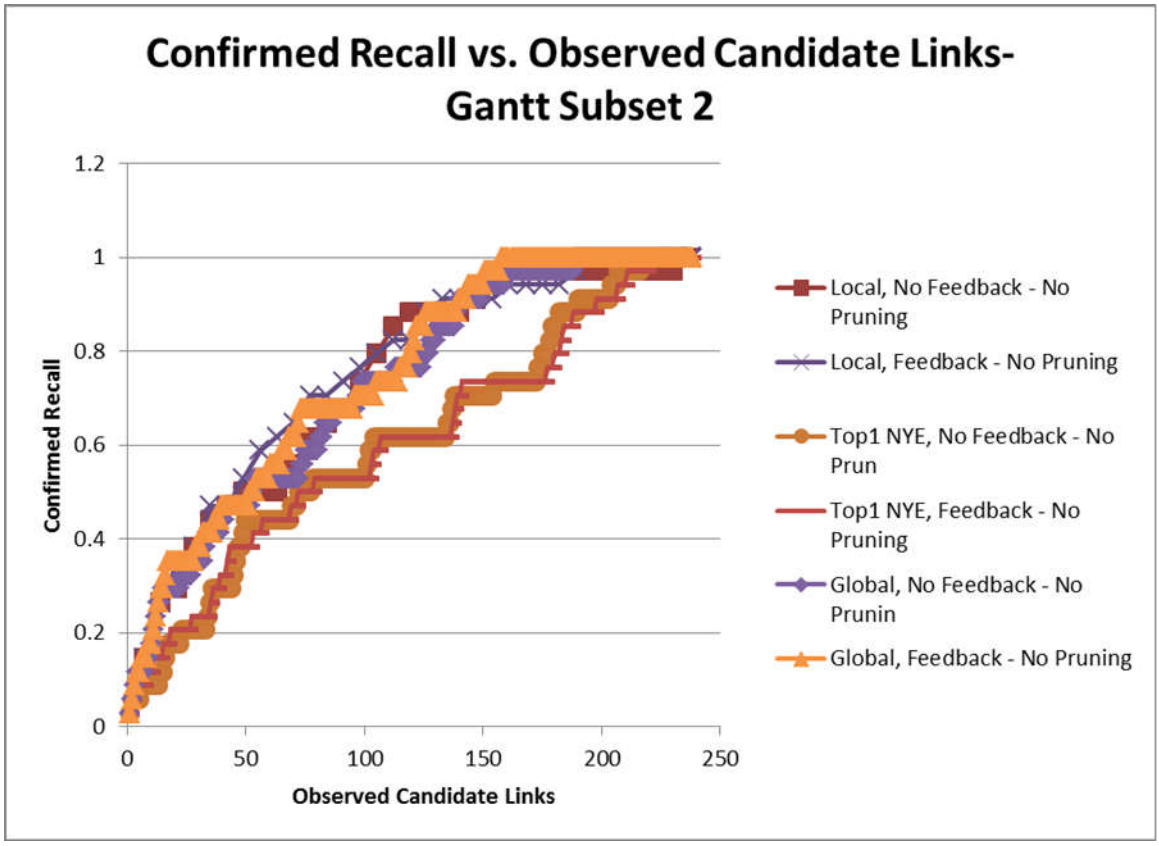

Fig 10 Confirmed Recall vs. Observed Candidate Links for Gantt Subset 2 – No Pruning Methods

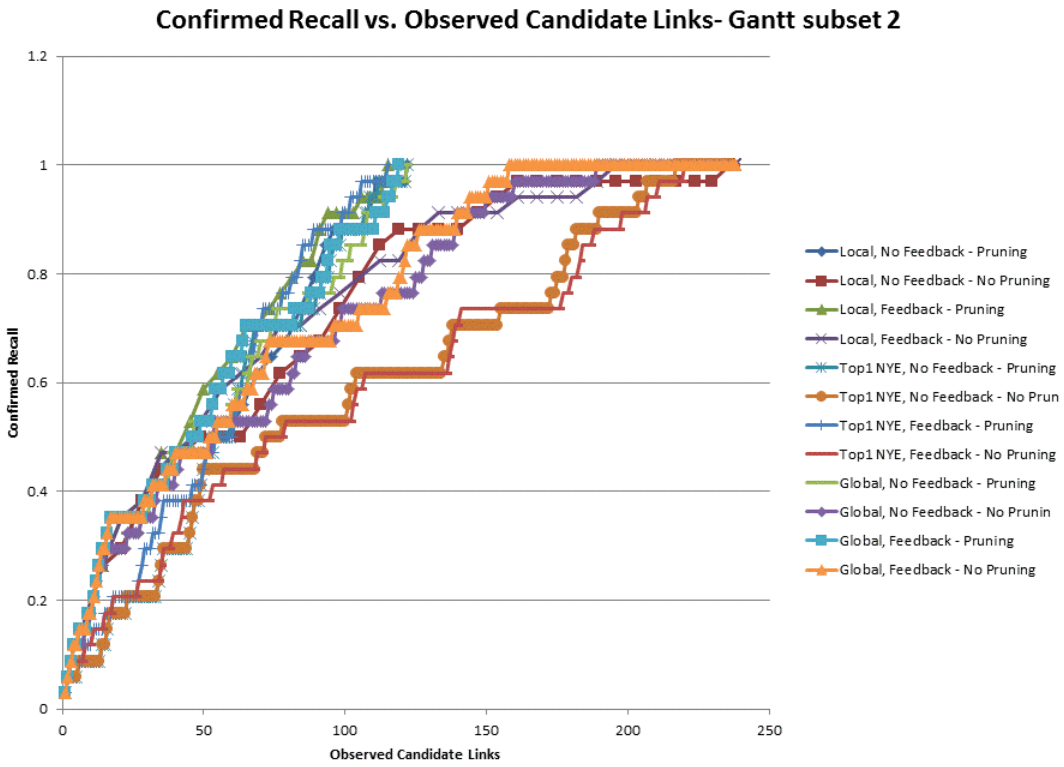

Fig 11 Confirmed Recall vs. Observed Candidate Links for Gantt Subset 2 – All methods

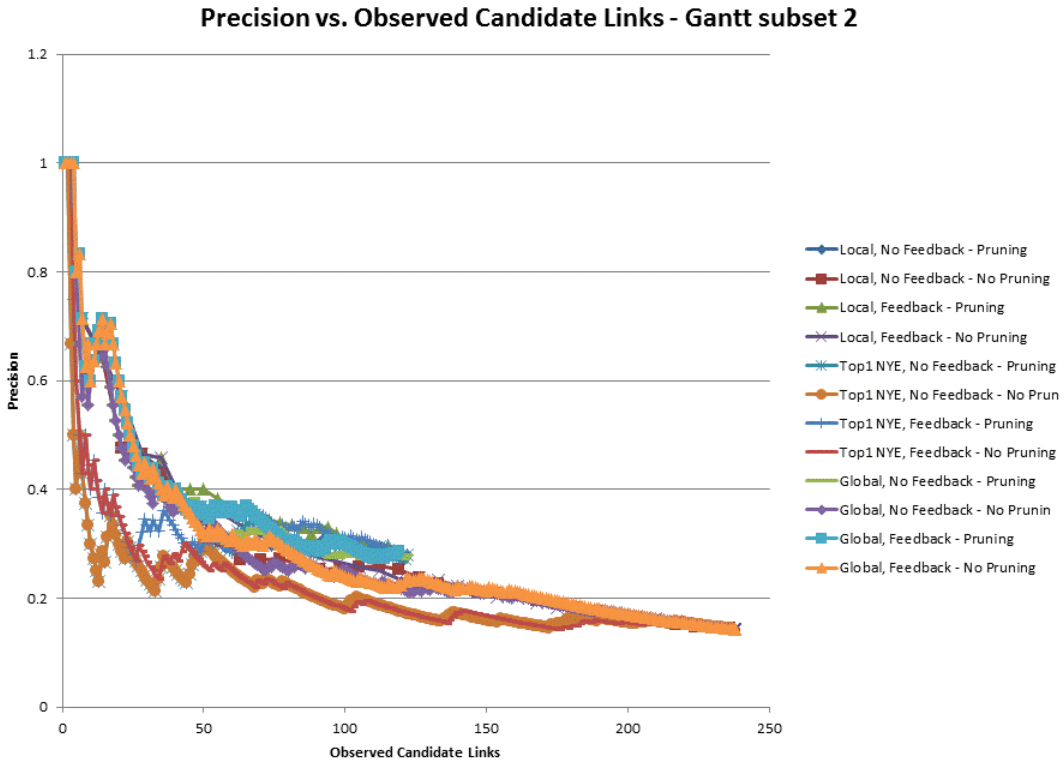

Fig 12 Precision vs. Observed Candidate Links for Gantt Subset 2

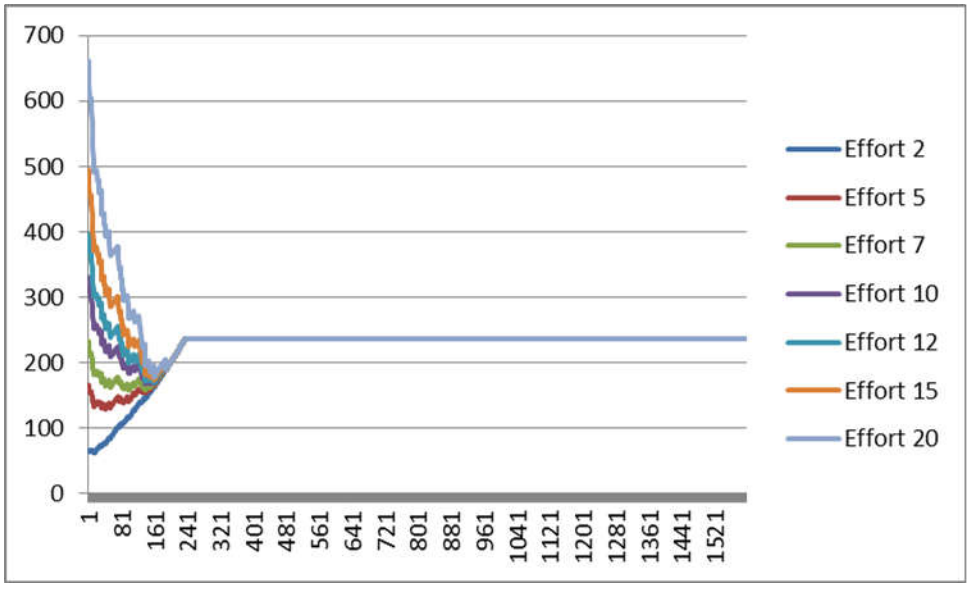

Fig 13 Global No Feedback No Pruning – all effort ratios (effort vs. observed candidate links)

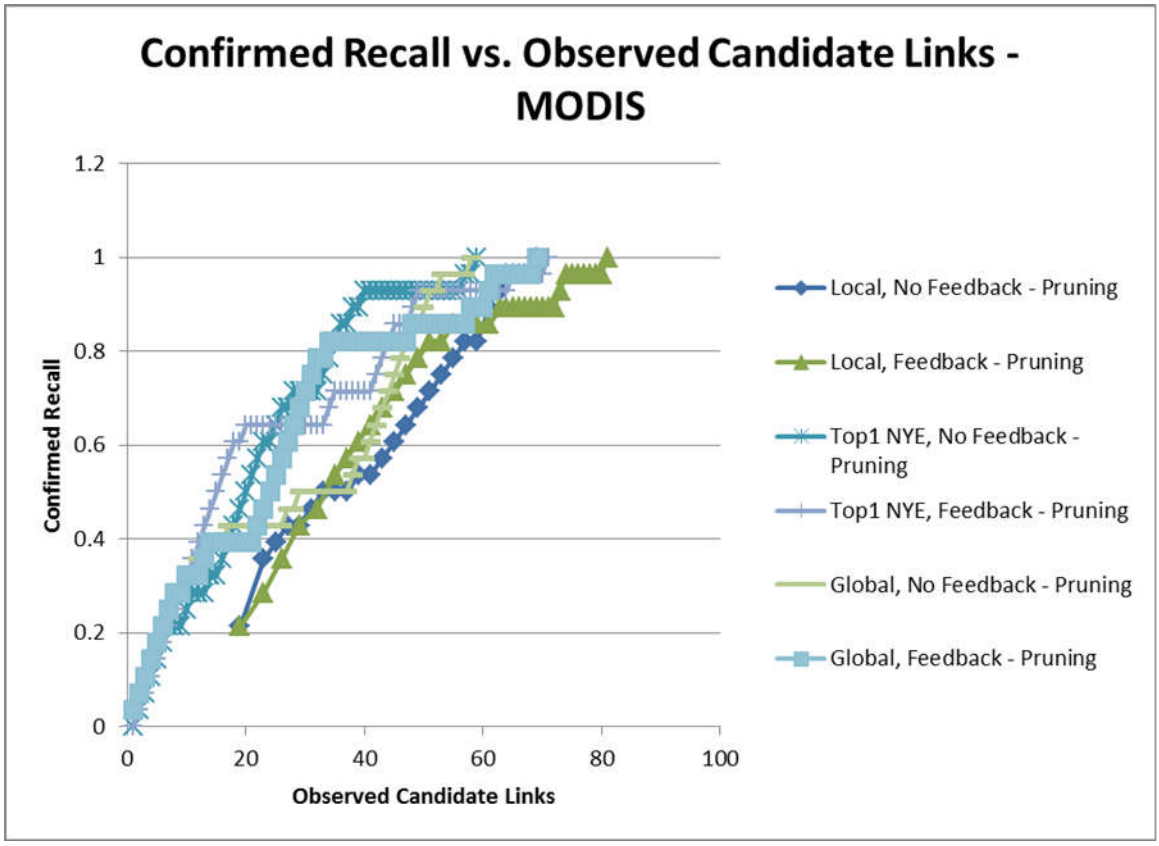

Fig 14 Confirmed Recall vs. Observed Candidate Links for MODIS – Pruning

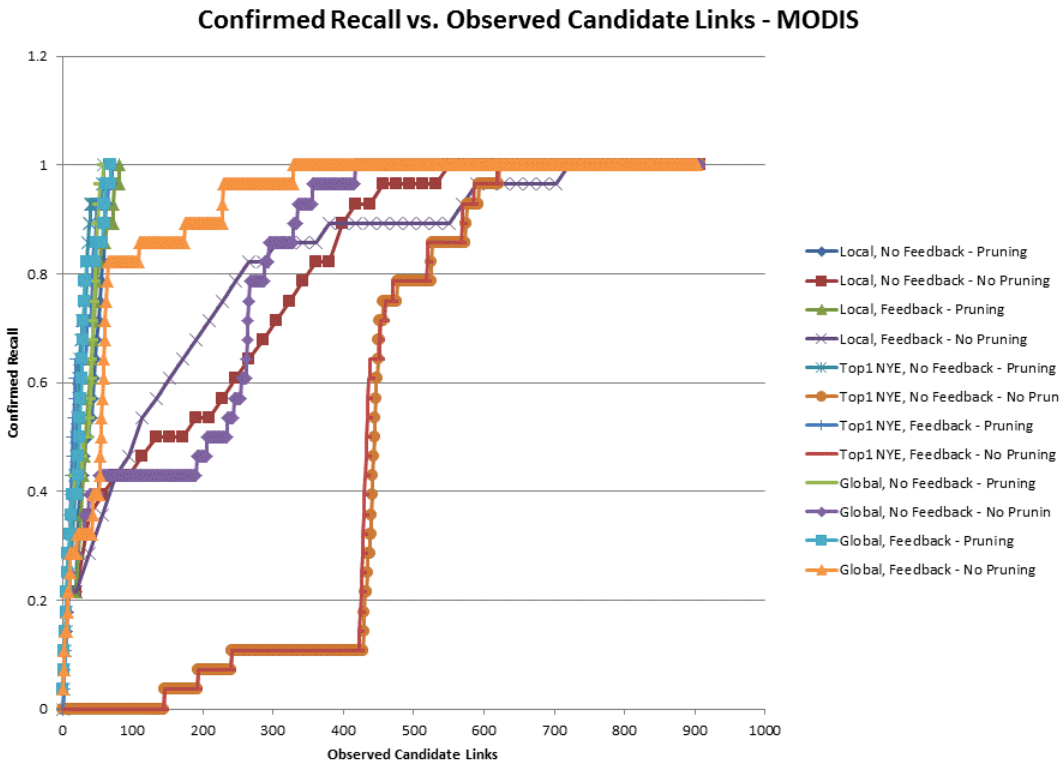

Fig 15 Confirmed Recall vs. Observed Candidate Links for MODIS – All Methods

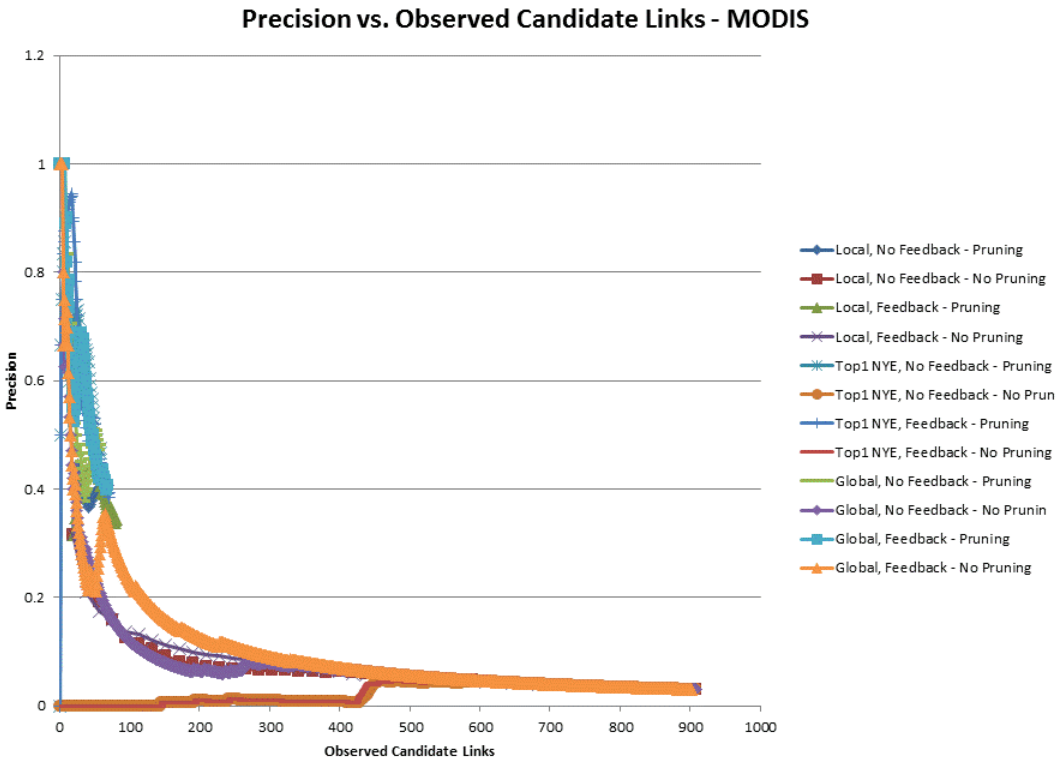

Fig 16 Precision vs. Observed Candidate Links for MODIS – All Methods

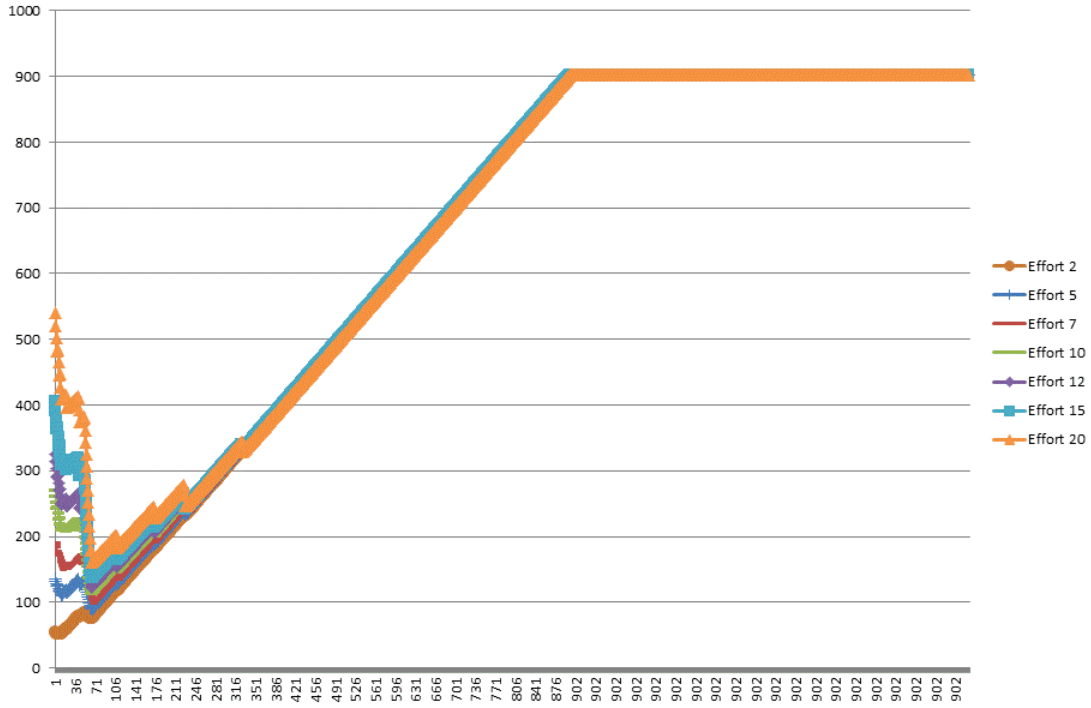

Fig 17 Global Feedback No Pruning – all effort ratios (effort vs. observed candidate links)

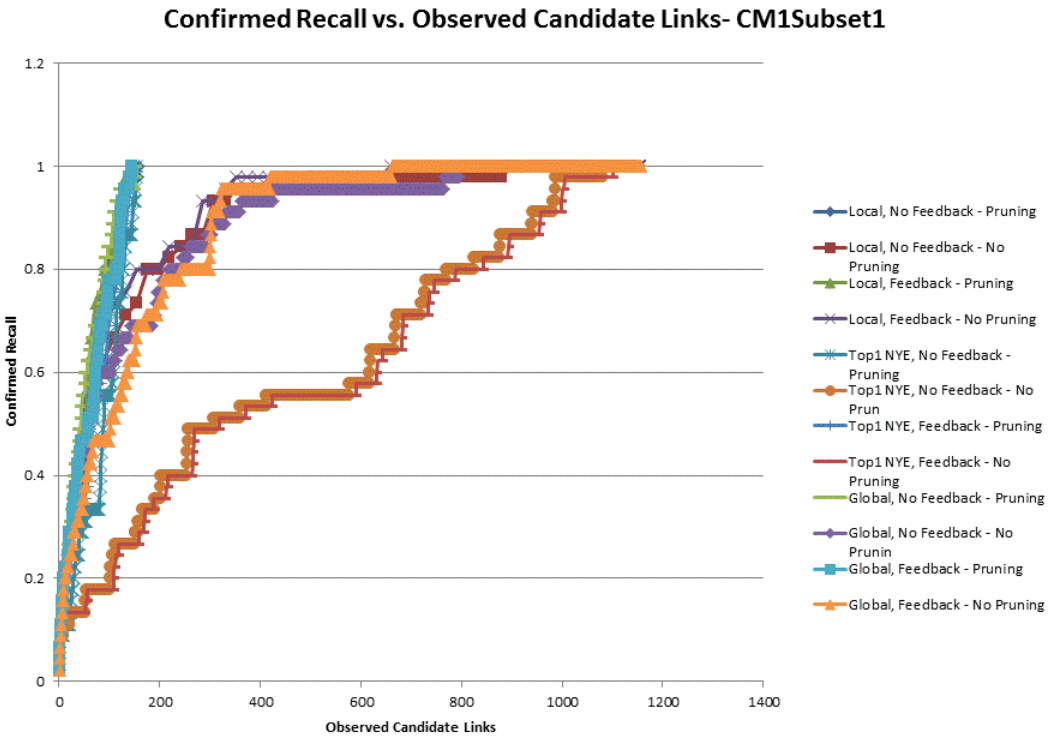

Fig 18 Comparison of twelve methods applied to CM-1 subset 1

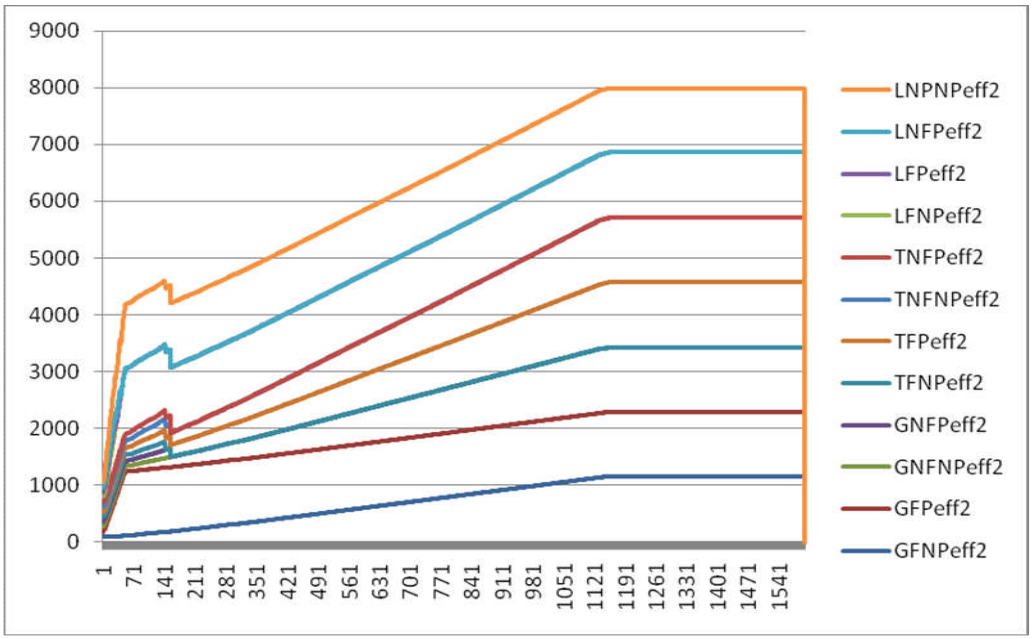

Fig 19 Comparison of twelve methods, effort 2 (σ=2) applied to CM-1 subset 1

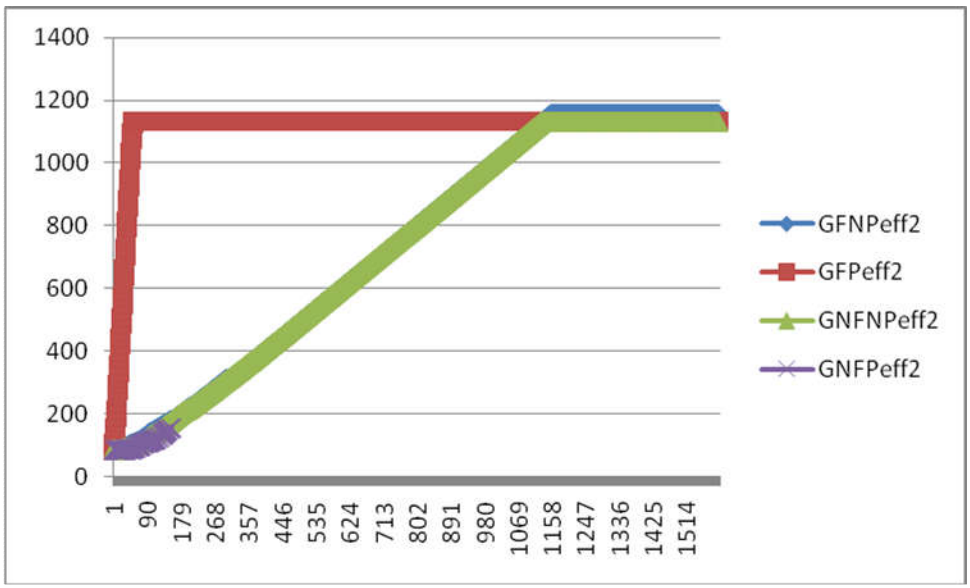

Fig 20 Comparison of Global methods, effort 2 (σ=2) applied to CM-1 subset 1

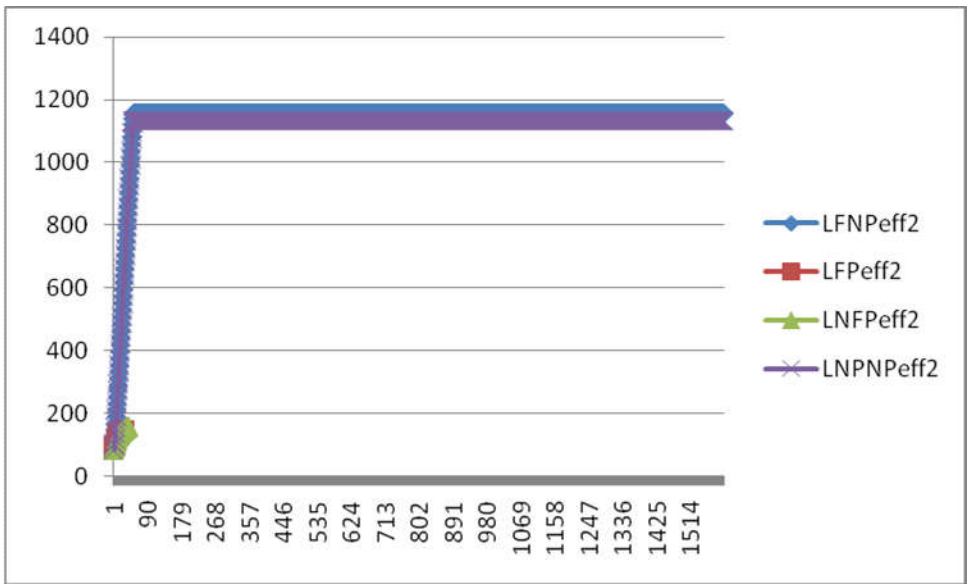

Fig 21 Comparison of Local methods, effort 2 (σ=2) applied to CM-1 subset 1

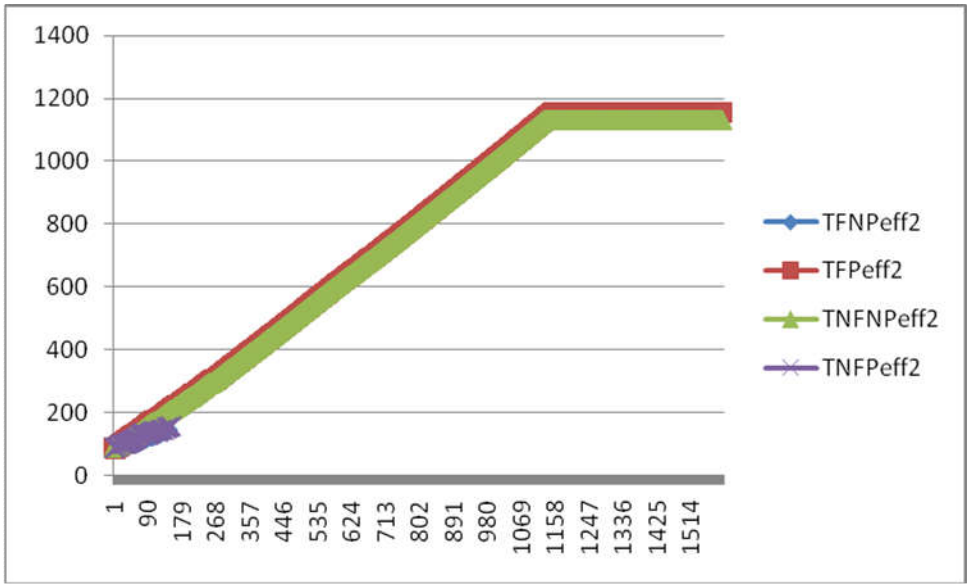

Fig 22 Comparison of Top 1 NYE methods, effort 2 (σ=2) applied to CM-1 subset 1