Defence Research and Development Canada
Recherche et développement pour la défense Canada

Abstracting PageRank To Dynamic Asset Valuation

Reginald Sawilla

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2006-243
December 2006

Canada

Abstracting PageRank To Dynamic Asset Valuation

Reginald Sawilla
Defence R&D Canada – Ottawa

Defence R&D Canada – Ottawa
Technical Memorandum
DRDC Ottawa TM 2006-243
December 2006

Principal Author

Original signed by Reginald Sawilla

Reginald Sawilla

Approved by

Original signed by Dr. Julie Lefebvre

Dr. Julie Lefebvre
Head/NIO Section

Approved for release by

Original signed by Dr. Cam Boulet

Dr. Cam Boulet
Head/Document Review Panel

Abstract

We present a method that quickly and dynamically calculates a relative value for all assets in an organization in any context in which dependencies may be specified. The idea is a novel application of the Google PageRank algorithm that extends it to all assets and all contexts. In fact, web pages become a special case of asset ranking in a functionality context. Additionally, our algorithm provides the capability to specify individual dependency weights that one asset has upon another and individual intrinsic values for each asset. An asset category dependency model is proposed and entity-relationship structures are given to assist in continued development.

Our scheme works in general and will provide asset valuation in any context, be it confidentiality, integrity, availability, or even political capital. In this document we combine the idea of asset dependency with an established web page ranking technique. Google's PageRank algorithm is generalized in two key areas, creating an algorithm we call AssetRank. It calculates a relative importance ranking for every asset in the organization from information to tanks to partners. The ranking will assist decision makers by highlighting which assets are the most highly depended upon by the organization.

Résumé

Nous présentons une méthode qui permet de calculer de façon rapide et dynamique une valeur relative pour tous les actifs d'une organisation dans tout contexte où des dépendances peuvent être spécifiées. L'idée proposée est une application novatrice de l'algorithme PageRank de Google, application qui l'étend à tous les actifs et à tous les contextes. En réalité, les pages Web deviennent un cas spécial de classement des actifs dans un contexte de fonctionnalité. De plus, notre algorithme permet de spécifier les pondérations des dépendances individuelles qu'un actif présente par rapport à un autre, ainsi que les valeurs intrinsèques individuelles de chaque actif. Un modèle de dépendance de catégories d'actifs est proposé ainsi que des structures entité-relations pour faciliter le développement ultérieur.

Notre système fonctionne en général et sert à évaluer les actifs dans tout contexte, qu'il s'agisse de confidentialité, d'intégrité, de disponibilité ou même de capital politique. Dans ce document, nous associons l'idée de dépendance des actifs à une technique établie de classement hiérarchique de pages Web. Nous avons généralisé l'algorithme PageRank de Google dans deux domaines clés, créant ainsi un algorithme que nous appelons AssetRank. Cet algorithme calcule le classement d'importance relative de chaque actif de l'organisation, qu'il s'agisse d'information, de chars d'assaut ou de partenaires. Le classement hiérarchique aide les décideurs en faisant ressortir les actifs dont l'organisation dépend le plus.

This page intentionally left blank.

Executive summary

Abstracting PageRank To Dynamic Asset Valuation

Reginald Sawilla; DRDC Ottawa TM 2006-243; Defence R&D Canada – Ottawa; December 2006.

Background: For administrators to have situational awareness of their network and to know its defensive posture, they need to know what are its exposed, critical assets. Changes in organizational priorities, network events, policies, topology, and threats invalidate a statically generated defensive posture. Since these factors may change frequently, a dynamic system is required that can rapidly provide an amended situational awareness. Ongoing research and commercial product development seek to provide a solution to the problem of finding what assets are exposed. The current state of the art is discussed in [1] but none of the solutions answer the question of which assets are most critical to an organization's current alignment.

Principal results: We present a method that quickly and dynamically calculates a relative value for all assets in an organization in any context in which dependencies may be specified. The idea is a novel application of the Google PageRank algorithm that extends it to all assets and all contexts. In fact, web pages become a special case of asset ranking in a functionality context. Additionally, our algorithm provides the capability to specify individual dependency weights that one asset has upon another and individual intrinsic values for each asset. An asset category dependency model is proposed and entity-relationship structures are given to assist in continued development.

Significance of results: This document proposes a dynamic asset valuation system that determines an organization's critical assets. Given a list of assets and their dependencies (within a user-specified context), this dynamic asset valuation system will rank the assets in order of their criticality. The ranked list can be used to accent a list of exposed assets thus producing a list of exposed, critical assets.

There have been other efforts in this area but they have failed to answer the requirements of automation, scalability, dynamicity, or granularity. This method builds upon mature and proven mathematical techniques and has been lauded by industry and defence experts.

Future work: Enterprise testing will reveal the best damping factor, dependency weights, and intrinsic values to use in organizational use. Also, PageRank is only one of many citation and web page ranking algorithms. We wish to see if other algorithms can be adapted to use a dependency topology and compare their ranking to that generated by AssetRank. In addition, an asset will have a different AssetRank in each of its contexts. Research is

needed into how values from diverse contexts may be combined into a single criticality value. Finally, research is needed to determine the best way to combine AssetRanks of communities of assets so that communities of equal function but different sizes may be effectively compared.

Sommaire

Abstracting PageRank To Dynamic Asset Valuation

Reginald Sawilla ; DRDC Ottawa TM 2006-243 ; R & D pour la défense Canada – Ottawa ; décembre 2006.

Contexte : Afin de connaître la situation de leur réseau et d'évaluer sa position défensive, les administrateurs doivent savoir quels actifs cruciaux sont exposés. Les changements survenant dans les priorités organisationnelles, les événements réseaux, les politiques, la topologie et les menaces invalident une position défensive statistiquement générée. Étant donné que ces facteurs peuvent changer fréquemment, il faut un système dynamique, capable de fournir rapidement une connaissance actualisée de la situation. Par la recherche continue et le développement de produits commerciaux, on tente de trouver une solution au problème de l'identification des actifs exposés. L'état actuel des connaissances est décrit dans [1], mais aucune des solutions proposées ne permet de savoir quels sont les actifs les plus importants pour l'harmonisation actuelle d'une organisation.

Principaux résultats : Nous présentons une méthode qui permet de calculer de façon rapide et dynamique une valeur relative pour tous les actifs d'une organisation dans tout contexte où des dépendances peuvent être spécifiées. L'idée proposée est une application novatrice de l'algorithme PageRank de Google, application qui l'étend à tous les actifs et à tous les contextes. En réalité, les pages Web deviennent un cas spécial de classement des actifs dans un contexte de fonctionnalité. De plus, notre algorithme permet de spécifier les pondérations des dépendances individuelles qu'un actif présente par rapport à un autre, ainsi que les valeurs intrinsèques individuelles de chaque actif. Un modèle de dépendance de catégories d'actifs est proposé ainsi que des structures entité-relations pour faciliter le développement ultérieur.

Importance des résultats : Ce document propose un système d'évaluation dynamique des actifs qui permet d'identifier les actifs les plus cruciaux d'une organisation. À partir d'une liste d'actifs et de leurs dépendances (dans le contexte spécifié par l'utilisateur), ce système d'évaluation dynamique classe les actifs en fonction de leur importance critique. La liste ainsi ordonnée peut être utilisée pour mettre en relief les actifs exposés qui sont essentiels.

Il y a eu d'autres travaux de recherche dans ce domaine, mais aucun n'a réussi à satisfaire aux exigences de l'automatisation, de l'évolutivité, de la dynamicité ou de la granularité. La méthode proposée est fondée sur des techniques mathématiques stables et éprouvées, et a été reconnue par l'industrie et les experts en matière de défense.

Travaux futurs : Les essais opérationnels révéleront le meilleur facteur d'amortissement, les pondérations des dépendances et les valeurs intrinsèques à utiliser dans un contexte organisationnel. En outre, PageRank n'est qu'un parmi de nombreux algorithmes de citation et de classement de pages Web. Nous souhaitons déterminer s'il est possible d'adapter d'autres algorithmes pour utiliser une topologie de dépendances et comparer leur classement à celui qui est produit par AssetRank. De plus, un actif aura un classement AssetRank différent dans chacun de ses contextes. Il faut poursuivre les recherches pour déterminer comment les valeurs de divers contextes peuvent être combinées pour former une seule valeur de criticité. Enfin, il faut également faire d'autres recherches pour trouver la meilleure façon de combiner les classements AssetRank de groupes d'actifs afin de pouvoir comparer efficacement des groupes de fonction égale, mais de tailles différentes.

Table of contents

Abstract i

Résumé i

Executive summary iii

Sommaire v

Table of contents vii

List of figures ix

List of tables x

List of algorithms xi

Acknowledgements xii

1 Introduction 1

2 Assets 2

2.1 Defined 2

2.2 Value 2

2.2.1 Purpose of asset valuation 2

2.2.2 Dynamicity 3

3 Assets as a web 3

3.1 Asset dependencies 3

3.2 PageRank 3

3.2.1 Value flow 3

3.2.2 Cycles 4

3.2.3 Equation 5

3.2.4 Weighted graph 5

3.2.5 Sources and Sinks 6

3.2.6 Partial independence 7
3.2.7 Intrinsic asset value 7
4 Dependency Diagram 8
4.1 High level view of asset dependencies 8
5 Implementation 10
5.1 Database environment 10
5.2 Data model 10
5.3 Dependency matrix instantiation 11
5.4 Intrinsic value computation 13
5.5 Steady-state computation 13
5.5.1 Matrix multiplication 13
5.5.2 Solving the system 14
5.6 Performance 15
6 Examples 17
6.1 Toy Military System 17
6.2 File valuation in MS Office 2003 23
7 Dependency templates 24
8 Future work 26
9 Conclusion 27
References 28

List of figures

Figure 1: Value flows to dependencies 4

Figure 2: Asset Cycles 4

Figure 3: Asset Availability Categorization 9

Figure 4: Entity-Relationship diagram 11

Figure 5: Performance Testing Up To 100,000 Assets 17

Figure 6: Toy Military System 18

Figure 7: Results of LabelledDependencyMatrix query 20

Figure 8: Entity-Relationship diagram with templates 25

List of tables

Table 1: Asset dependency weight values 6

Table 2: Iterations Required For Randomly Generated Systems 16

Table 3: Toy System Dependency data 19

Table 4: Toy System AssetRanks 21

Table 5: Toy System AssetRanks with an altered dependency 22

Table 6: Toy System AssetRanks with high officer value 23

Table 7: Highest 20 ranked dependencies of Microsoft Office 2003 24

List of algorithms

1 MatrixProduct: Compute the product of two matrices 14
2 SolveSystem: Compute Steady State Iteratively 15
3 CompareVectors: Compare Two Vectors Within A Tolerance 16

Acknowledgements

The author is grateful to Luc Beaudoin and Craig Burrell for generously sharing their experience, knowledge, and resources.

1 Introduction

Organizations today are overwhelmed with security bulletins alerting them to the threats of the day. In 2006, an average of 19 new vulnerabilities in commercial software were disclosed every single day of the year [2]. For administrators to have situational awareness of their network and to know its defensive posture, they need to know what are its exposed, critical assets. Changes in organizational priorities, network events, policies, and threats invalidate a statically generated defensive posture. Since these factors may change frequently, a dynamic system is required that can rapidly provide an amended situational awareness. Ongoing research and commercial product development seek to provide a solution to the problem of finding what assets are exposed. The current state of the art is discussed in [1] but none of the solutions answer the question of which assets are most critical to an organization's current alignment. Recent advances allow an administrator to generate a list of the vulnerabilities of his or her network but for large organizations the list will contain thousands of items, resulting in information overload. This document proposes a dynamic asset valuation system that provides a solution in determining an organization's critical assets. The list of critical assets accents a list of exposed assets thus producing a list of exposed, critical assets.

Confidentiality, integrity, and availability (CIA) are the essential security principles and every security technique must address them [3]. To properly prioritize resources, it is critical for an organization to know the value of its assets, not only in monetary terms, but also the CIA triad. Asset values change and the military in particular is faced with this problem. Mission priorities adjust frequently and a method is needed that can quickly and dynamically revalue all assets in the system based on the current priorities.

Our scheme works in general and will provide asset valuation in any context, be it confidentiality, integrity, availability, or even political capital. In this document we combine the idea of asset dependency with an established web page ranking technique. Google's PageRank algorithm is generalized in two key areas, creating an algorithm we call AssetRank. It calculates a relative importance ranking for every object in the organization from information to tanks to partners.

We begin in Section 2 by defining assets, their value, and dynamicity. In Section 3 we introduce the idea of a dependency topology for assets, describe the PageRank algorithm, and extend it to allow for dependency weights and the intrinsic value of an asset. Section 4 categorizes assets so that AssetRanks can be correctly interpreted. Details of our implementation are examined in Section 5 where the matrix multiplication method of [4] is applied. Section 6 illustrates AssetRank with the examples of a simple military system and the file dependencies of Microsoft Office 2003. In Section 7 a proposal is given for the creation of asset templates so that building and maintaining a large system is feasible. Finally, we present future work and concluding remarks in Sections 8 and 9.

2 Assets

2.1 Defined

We assume that an entity such as an organization desires to know a ranking of its assets according to their importance to the functioning of the entity. This ranking is always done within a context, whether they be the security contexts of confidentiality, integrity, availability, or the political contexts of power and public opinion. To facilitate the ranking of assets in general we adopt the strategy of a relative ranking of assets amongst themselves. The actual AssetRank value of the assets will not be important; we only seek to provide a ranked list.

An asset is any object that is of interest to the entity. It may be as tangible as a network router, intangible as reputation, or somewhere in the middle of the spectrum, as a service is. We adopt the terminology of categories, classes, and objects from object-oriented systems. Ultimately, the valuation is performed on objects. An object is a person, place, thing, or idea that has value to the entity. Hence, we define an asset to be any object and the collection of all assets form a system. The other terms we will use are defined in the following table.

Term	Definition	Example
Category	A collection of classes sharing a common attribute	Person
Class	A blueprint or prototype that defines the characteristics common to all objects of a certain kind	Defence Scientist
Object	An instance of a class	John Doe

2.2 Value

2.2.1 Purpose of asset valuation

Organizations today are inundated with data. CERT declares that in 2005 there were 5,990 new vulnerabilities discovered in commercial-off-the-shelf (COTS) software [2]. Enterprises have tens of thousands of computers along with supporting servers and network appliances that might be affected by vulnerabilities. Parallel research [5] is underway to determine what exposed assets are affected; however, this will produce a listing in the thousands.

Simply knowing what assets are vulnerable is insufficient. It is also necessary to know what assets are most critical to the organization. A listing of vulnerable assets that is sorted in order of their importance to the organization would be extremely valuable to system administrators.

We discuss value in various contexts but not in the context of replacement cost. An inexpensive device that is relied upon by 20% of the organization is worth much more than its

sticker price. A confidentiality and integrity value can be assigned to an asset that reflects its importance in protecting these attributes. Replacement cost is a value that is already easily obtained given adequate administrative record keeping. Throughout this paper we will primarily work in the context of availability but the concepts are general and may be applied in any context where a dependence can be specified.

2.2.2 Dynamicity

Determining asset value would not be as challenging if the value remained static and was easily predictable. The problem is that, especially in a military environment, the value of assets changes rapidly in response to operational needs. For example, suppose that a nation state is preparing to launch air strikes against certain targets. In the weeks leading up to the mission, the confidentiality of the site selection is extremely important. Once the mission is underway, the confidentiality of the site selection is not at all important but now the availability of the targeting systems is critical. A system for dynamic asset valuation must be able to rapidly revalue assets, taking into account the current context of the organization.

3 Assets as a web

3.1 Asset dependencies

Assets may be considered to have the same link structure as world wide web pages. If an asset a_1 depends upon asset a_2 in some context then we can imagine a link from a_1 to a_2. In the contexts of confidentiality, integrity, or availability values, the dependencies assets have upon each other create a web-like structure. In fact, web pages may be deemed a special case of assets in the information category. When viewed as an asset, a web page has a functional dependency upon every page it links to. Algorithms that use this link structure to rank web pages might also be extended to rank assets.

3.2 PageRank

The best known web page ranking algorithm is PageRank. This is due both to its success at accurately ranking web pages and to the fact that it is published in the open literature. The original publication is [6] and it is explained by others in detail in [7]. We present the algorithm here in terms of assets and then extend it to include link and asset weights.

3.2.1 Value flow

A portion of each asset's value flows to the assets that it relies upon. Thus, an asset's rank is determined by the assets that depend upon it. The amount of value that is distributed to dependants is a function of both the value of the dependant asset and the number of assets

that are depended upon. This is depicted in Figure 1 where the weight of the asset border depicts its value. Assets a_1 and a_2 in the left cluster have an average value. Since they both depend upon a_3, some of their value is transferred to it and so its value is increased. The border of asset a_2 in the right cluster indicates that it has a high value. Since it depends upon a_3, some of its value is transferred to a_3. Since it has above average importance, it transfers more value. Thus, the right-side a_3 is ranked higher than the left-side a_3.

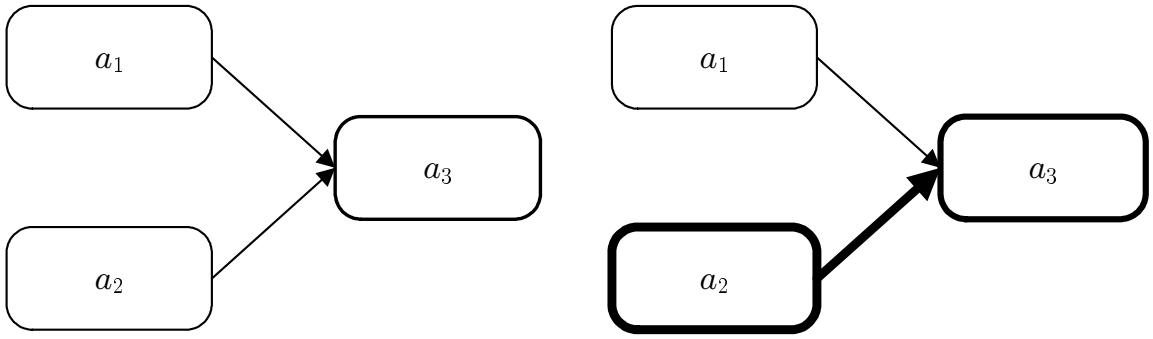


Figure 1: *Value flows to dependencies*

3.2.2 Cycles

A problem occurs when assets form a dependency cycle. A simple case is illustrated with web pages in Figure 2. Here, page A links to page B. Page B has a single link to Page C and Page C in turn has a single link to Page B. Pages B and C have no outgoing links other than those forming the cycle. Pages B and C form a cycle that would accumulate value indefinitely and disproportionately to their importance. That is, value from web pages further up the web hierarchy continues to flow to these web pages but there is no way for the value to flow out of this cycle.

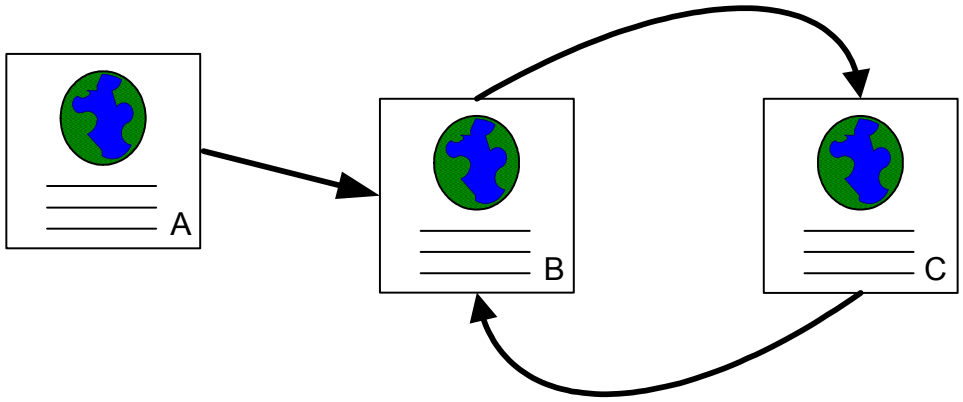


Figure 2: *Asset Cycles*

This problem is resolved by introducing a damping factor. The damping factor is a value between 0 and 1 that is multiplied by every asset in the system during computation of the

rank. As we will see, this avoids the accumulation of value in cycles and allows the system to stabilize.

3.2.3 Equation

The notation that will be used throughout the paper is given below.

- a is an asset and will often be subscripted
- F_a is the set of assets that a depends upon
- B_a is the set of assets that a provides for (assets that depend upon a)
- $N_a = |F_a|$ is the outdegree, or number of dependencies, of a
- $\delta \in (0, 1)$ is the damping factor

An initial equation for the AssetRank of a is

$$x_a = \delta \sum_{v \in B_a} \frac{x_v}{N_v} + (1 - \delta) \ .$$

This equation is identical to that used in the computation of PageRank and it states that the AssetRank of a is obtained as follows. For every asset v that depends on a, compute its AssetRank and divide that by the number of assets that it depends upon. Multiply the sum of all of these quotients by the damping factor δ and then add $1 - \delta$ to the result. The $1 - \delta$ value is a default AssetRank given to every asset.

We will represent assets as a directed graph with assets as vertices in the set V and dependencies as arcs in the set A. Write the dependency graph for n assets as an $n \times n$ matrix $D = [d_{rc}]$ where the entry d_{rc} is $1/N_{a_c}$ if $(a_c, a_r) \in A$ and 0 otherwise.[1] Let X be an n element non-zero vector and $\mathbf{1}_n$ be an n element vector of ones. Then the AssetRanks are given by X such that $X = \delta DX + (1 - \delta)\mathbf{1}_n$. In our implementation this is computed iteratively using the Jacobi method so that $X_t = \delta DX_{t-1} + (1 - \delta)\mathbf{1}_n$ where $X_0 = \mathbf{1}_n$.

3.2.4 Weighted graph

An asset may not depend upon other assets equally. Some may be critical to its functioning while others provide minimal utility. The PageRank scheme does not provide flexibility to factor in these different dependencies but we have added this ability in AssetRank.

For the sake of discussion, the scheme in Table 1 may be used to weight dependencies. Experimentation will determine appropriate values for the weights and how they are incorporated into the calculations. The weights may be used linearly or squared for example.

[1] The subscripts r and c are meant as mnemonics for the reader for the *row* and *column* indices of the dependency matrix.

Dependence	Weight
None	0
Low	1
Medium	2
High	3

Table 1: *Asset dependency weight values*

In the special case of web pages this capability could be used to rank links at different values depending upon where in the page they appear. A link clustered with many other links would be valued lower while a link prominently stated in the first paragraph of the page could be weighted higher. In effect, the standard PageRank algorithm gives a weight of 1 to all links.

The equations given in Section 3.2.3 are extended to incorporate these weights as follows. Let $a_c,\ a_r \in V$ be assets. If a_c depends on a_r, denote the weight of its dependence by w_{rc} and add the arc (a_c, a_r, w_{rc}) to A.[2] The entries d_{rc} of the $n \times n$ matrix D are given by

$$d_{rc} = \begin{cases} \dfrac{w_{rc}}{\displaystyle\sum_{(a_c, a_{r'}, w_{r'c}) \in A} w_{r'c}}, & \text{if } (a_c, a_r, w_{rc}) \in A \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Equation (1) states that the rcth entry of D is given by the weight w_{rc} divided by the sum of all weights in the column c. This normalizes the weights in each column and causes each column to sum to 0 or 1.

3.2.5 Sources and Sinks

Graph theory defines a source node to be one with indegree 0 and a sink node to be one with outdegree 0. We adopt the same nomenclature so that an asset upon which no other asset depends is called a *source asset* and an asset that does not depend upon any other asset is called a *sink asset*. The zero-sum columns in the matrix D defined by Equation (1) above are caused by sink assets.

Similarly to cycles, sink assets inappropriately concentrate value. There are two equivalent ways to manage sink assets that are described in detail in [7]. First, there is the method used by PageRank's inventors [6]. This method would treat sink assets as having a link to

[2] Intuitively, one may wish to label the weights w_{cr}; however, for the matrix to be stochastic, the dependency data must be given in columns. Labelling the weights as we have allows the indices to agree with the position of the data in the dependency matrix. An alternative is to write the data in rows, thus allowing the weight indexing to read more naturally. The calculations would then be performed with the transpose of the matrix.

every other asset. The second method is the one chosen here. We cope with sink assets by adding the sink compensator node a_{n+1} to the graph. For every sink asset a_c, add the arc $(a_c, a_{n+1}, 1)$. The node a_{n+1} is itself a sink asset so add the arc $(a_{n+1}, a_{n+1}, 1)$. The sink compensator will concentrate value because all sink assets are now funneling their value to it. Once the system has stabilized, the sink compsensator asset may be discarded or its value may be proportionately returned to the system.

Due to normalization of the columns and the addition of a sink compensator, the matrix D is stochastic. It is proven in [7] that due to the inclusion of the damping factor, the system will stabilize. We have found in practice that real systems stabilize in roughly 20 iterations while randomly generated dependency systems stabilize in approximately 45 iterations. We speculate that this occurs because real systems cluster assets while random systems rarely have clustered assets.

3.2.6 Partial independence

Some assets completely depend upon other assets to function while others function to one degree or another on their own. For example, a computer completely depends upon power for its availability. It provides no utility to the organization without it. On the other hand, an email application provides full utility when connected to a mail server but can also work offline and allow the user to read previously downloaded emails. The user also can create new emails and queue them for delivery when it connects to the mail server at a later time.

An asset is *partially independent* if it has one or more dependencies but still has utility in the absence of its dependencies. As discussed previously, the technique used to avoid pooling value with fully independent sink assets was to add a dependency to the sink compensator asset. Thus, the functional independence of the asset is reflected by the addition of this dependency. Partially independent assets are handled in a similar manner. If an asset a_i has low, medium or high utility in the absence of its dependencies, then a dependency is added from a_i to the sink asset a_{n+1} with a weight corresponding to its utility. This maintains consistency with the way that sink assets are managed and evaluated.

3.2.7 Intrinsic asset value

The dependency structure presented in Section 3.2.4 values all assets exclusively by the degree to which other assets depend upon them. In many cases this may be sufficient but in other cases assets will have intrinsic value in and of themselves. For example, we may want high value to flow from a general to those assets that he or she depends upon. This may be accomplished by giving the general an intrinsic value that is higher than other people. The intrinsic value may also be an effective way to combine asset values from different contexts. An asset with a high political value may be given an increased intrinsic value so that its dependencies are ranked higher in the system. Further research and experimentation will determine if this approach is advantageous in practice.

The $1-\delta$ portion of the equation $X = \delta DX + (1-\delta)\mathbf{1}_n$ contributes a static value to the AssetRank of each asset. This value is propagated along the asset's dependency path. We propose the ability of altering an asset's intrinsic value by adjusting the static contribution value.

Multiplying $1-\delta$ by $\mathbf{1}_n$ has the effect of assigning an intrinsic asset value of $1-\delta$ to each asset. Instead, we will use an intrinsic value vector, denoted IV, by which $1-\delta$ will be multiplied. In the PageRank algorithm, $IV = \mathbf{1}_n$.

Intrinsic asset values are assigned to each asset with the default value being 1. By [7, Lemma 2.1] and our choice of the initial AssetRank value being given by $X_0 = \mathbf{1}_n$, the total value of all assets sums to n.[3] If we normalize the intrinsic value vector by setting[4]

$$\overline{IV} \leftarrow n\frac{IV}{||IV||_1}$$

then $||\overline{IV}||_1 = n$. It can be seen from the proof of [7, Lemma 2.1] that this will maintain the overall system value at n. The AssetRanks are then given as the solution to the equation

$$X = \delta DX + (1-\delta)\overline{IV} \ . \tag{2}$$

4 Dependency Diagram

4.1 High level view of asset dependencies

AssetRank provides a ranking of all assets in the system. The values are not particularly useful to compare incomparable assets. The values are best interpreted within the context of similar classes of objects. We group these classes into categories and propose using a category-class-object hierarchy.

A starting point for a high-level partitioning is given in Figure 3 which depicts that an organization depends upon its operations to function. The operations in turn depend upon people who depend upon information. To obtain information they interact with desktop software that depends upon back office software. The back office software relies upon a logical communications path to communicate back to the client and this logical path depends upon physical computer infrastructure to function. The computer infrastructure depends upon other infrastructure such as the power grid and eventually everything depends upon natural resources such as coal, water, or wind. One may imagine all of the dependencies within each category that are also required along with the interdependencies between categories. For example, an organization depends upon other organizations for the power grid and server software depends upon people to keep it operational.

[3] Or $n+1$ in the case where a sink asset is present.

[4] The $L1$-norm $||X||_1$ of a vector X is the sum of its entries.

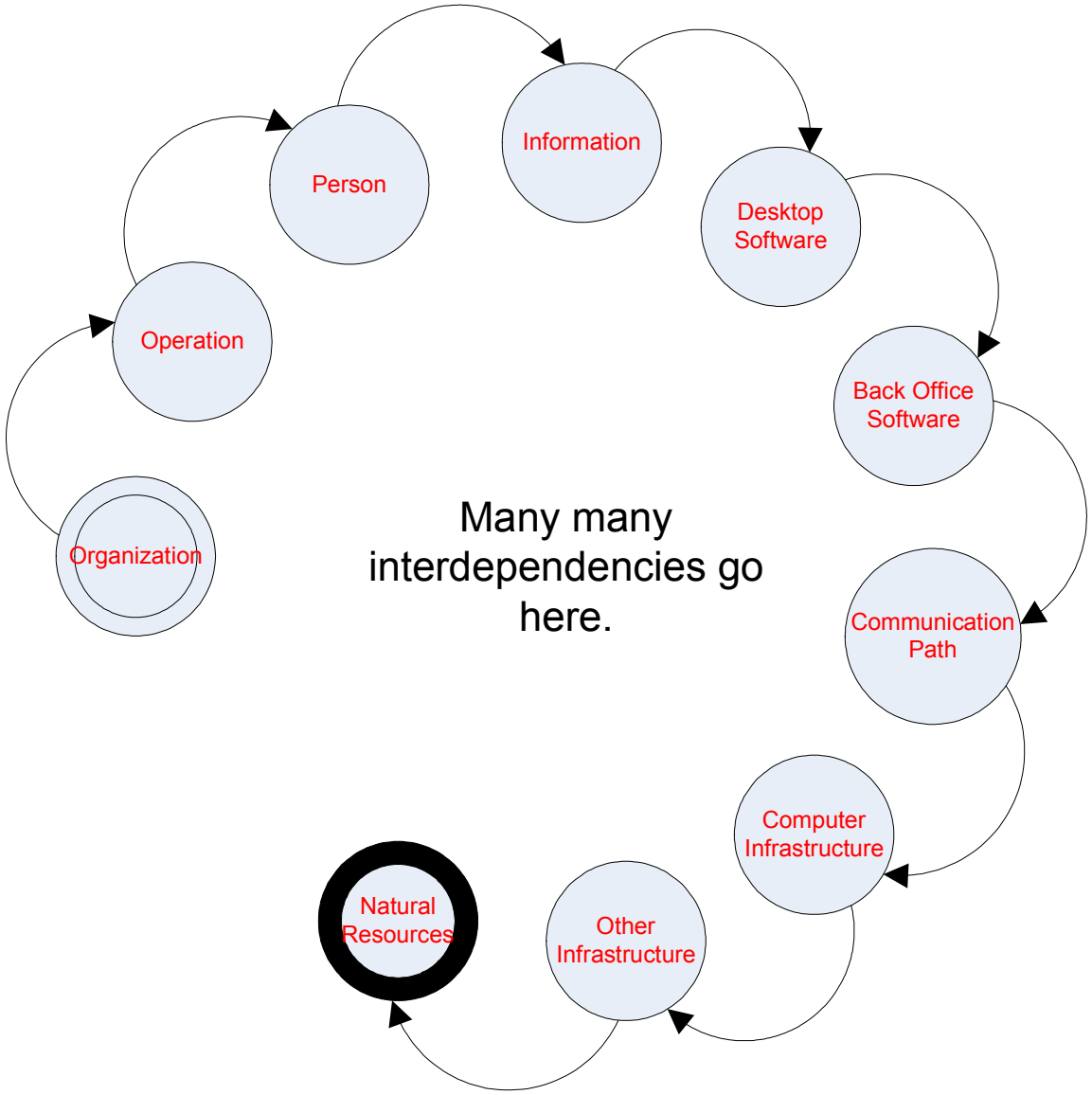


Figure 3: *Asset Availability Categorization*

Within each category are classes and objects instantiating the classes. For example, the web page `http://www.gc.ca` is within the information category. Within this category their exists the class Web Page and the object located at `http://www.gc.ca` is a specific instantiation of the Web Page class. Another example is an engineer Jane Doe who is an instantiation of the Engineer class which is within the Person category.

5 Implementation

5.1 Database environment

We expect real-life systems to have millions of asset instances. This will result in a dependency matrix containing millions of rows and millions of columns. As such, efficiency of storage and computation are important. Given the volume of interdependencies and the fact that asset information will already be stored in databases, we assume that the dependency relationships will be stored in a SQL database. There are two methods that we see to realize the computation. One, perform all data manipulation and computation in the database, or two, transfer the data to a highly optimized program and then return the results to the database.

In this system, the only arithmetically intensive operation is calculating the steady state of Equation (2). In [4] we presented a SQL implementation of matrix multiplication and compared it with other approaches. The method takes advantage of the SQL join operation to multiply rectangular matrices. The dependency matrix in AssetRank is sparse and this multiplication method takes full advantage of that fact. It performs extremely well since its complexity depends only upon the number of non-zero entries in the matrix while the best known methods also depend upon the size of the matrix.

In the interest of rapid application development, the prototype environment was a Microsoft Access 2003 database. The dependency data was stored in tables according to the data model given in Section 5.2. Visual Basic for Applications was used for coding while SQL queries were used for all database operations. The prototype was written with an easy upsizing to a full SQL server in mind.

5.2 Data model

Dependencies are stored in the database relations Category, Instance, and Dependency as depicted in Figure 4. Standard Entity-Relationship (ER) database system terminology and notation is used in this section and may be found in a database theory text such as [8].[5]

The Category relation stores the category names along with an automatically generated unique identification number.

For the prototype, classes were not implemented so the Instance relation directly references the Category relation. We envision that in future versions Instance will reference the Class relation instead. The Class relation will provide templates of instances and will reference

[5]Diagrams were created with the ER Diagram template from Microsoft Office Visio for Enterprise Architects 2003 and thus reflect the specific conventions of the software. An arrow from one relation to another indicates a one-to-many relationship, in this case due to foreign keys. The specific position of the arrows does not convey information.

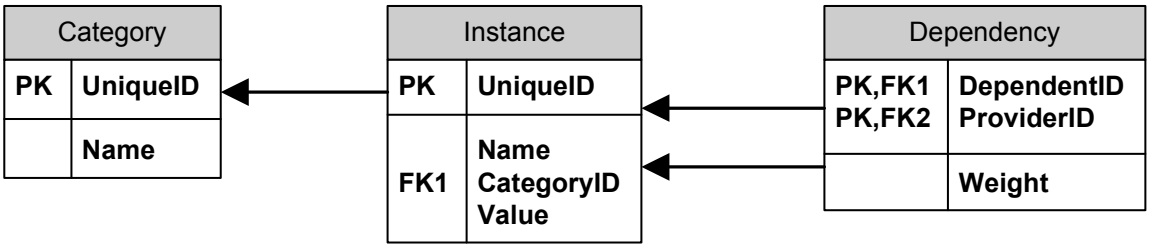


Figure 4: *Entity-Relationship diagram*

the Category relation. The attribute Name is the name of the Instance, CategoryID is the unique identification number of the category the instance belongs to, and value is the intrinsic value of the instance.

The two attributes DependentID and ProviderID in the Dependency relation are unique identifiers from the Instance relation and together they form the primary key. If asset a depends upon asset b then there will be a tuple in Dependency where DependentID is the Instance.UniqueID of a and ProviderID is the Instance.UniqueID of b. The weight with which a depends upon b is specified by the Weight attribute.

5.3 Dependency matrix instantiation

The dependency matrix D in Equation (2) may be computed using a SQL crosstab query. The crosstab query depends upon several other queries which will be presented first.

To begin, we require a query that adds a dependence for every sink asset as described in Section 3.2.5. The following query returns a tuple for every sink asset that adds a dependence upon the sink compensator node. Sink assets are defined to be all assets in the database that provide for an asset but do not depend upon any assets themselves. The Microsoft Access *DLookUp* function returns an integer that is the unique identification number of the sink compensator node in our database. The number 1 is specified as the value for the weight attribute but any number would work. Since each matrix column will be normalized to sum to 1, and the weight value returned by the query is the only value in the column (by definition of the query), the result will be the number 1 regardless of the value chosen here.

Query 1. *AddSink*

```
SELECT DISTINCT D1.ProviderID AS DependentID, CLng(DLookUp(
       "[UniqueID]","Instance","[Name] = 'Sink'")) AS ProviderID,
       1 AS Weight
FROM Dependency AS D1 LEFT JOIN Dependency AS D2 ON
```

```
     D1.ProviderID = D2.DependentID
WHERE D2.DependentID Is Null;
```

Query 2. *AddSinkUnionDependency*

```
SELECT * FROM Dependency
UNION
SELECT * FROM AddSink;
```

Query 2 simply returns the union of the tuples given by the AddSink query and the Dependency tuples in the database. To normalize the columns in our desired matrix, the sum of weights of the dependencies of each asset is computed by the following query.

Query 3. *SumOfWeight*

```
SELECT DependentID, Sum(Weight) AS SumOfWeight
FROM AddSinkUnionDependency
GROUP BY DependentID;
```

The normalized dependencies may now be computed.

Query 4. *NormalizedDependency*

```
SELECT D.DependentID, D.ProviderID,
       Weight/SumOfWeight AS NormalizedWeight
FROM AddSinkUnionDependency AS D INNER JOIN SumOfWeight AS SoW ON
     D.DependentID = SoW.DependentID;
```

We are now able to present the cross-tab query. An example of these queries in action will be given in Section 6.

Query 5. *LabelledDependencyMatrix*

```
TRANSFORM NZ(First(ND.NormalizedWeight),0)
SELECT I1.Name
FROM (Instance AS I2 INNER JOIN NormalizedDependency AS ND ON
     I2.UniqueID = ND.DependentID) INNER JOIN Instance AS I1 ON
     ND.ProviderID = I1.UniqueID
GROUP BY ND.ProviderID, I1.Name
ORDER BY I1.Name
PIVOT I2.Name;
```

5.4 Intrinsic value computation

An intrinsic value is required for each instance with the default being 1. The values across all instances are normalized so that the sum of all values is n where n is the number of assets in the system. If there are sink assets then a sink compensator asset is added which brings the system value up to $n + 1$. To compute the normalized intrinsic value vector $\overline{IV}$ of Equation 2 we first compute $n/||IV||_1$ and assign it to the variable CountDivSum with the aid of the query InstanceValueMultiplier.

Query 6. *InstanceValueMultiplier*

```
SELECT Count([UniqueID])/Sum([Value]) AS CountDivSum
FROM Instance INNER JOIN (SELECT DISTINCT DependentID
                          FROM AddSinkUnionDependency) AS D ON
     Instance.UniqueID = D.DependentID;
```

The normalized instance values are then obtained with the query InstanceNormalizedValue. It is necessary to compute the join with the subquery (named D in the query) because the system will likely contain instances listed in the Instance relation but not involved in any dependencies. This can occur with old instances that have become disconnected or new instances which have not yet been connected.

Query 7. *InstanceNormalizedValue*

```
SELECT Instance.UniqueID AS Row, [Value]*CountDivSum AS NormalizedValue
FROM Instance INNER JOIN (SELECT DISTINCT DependentID
                          FROM AddSinkUnionDependency) AS D
     ON Instance.UniqueID = D.DependentID;
```

5.5 Steady-state computation

5.5.1 Matrix multiplication

It is interesting to note that solving for X in Equation (2) in fact calculates the eigenvector of the right hand side corresponding to the eigenvalue 1. This may be done with elementary row operations on the dependency matrix or iteratively using the Jacobi method. The Jacobi method is computationally more efficient and is often used in practice to solve a linear system of equations. One may immediately notice that the matrix is sparse and large, even for a very simple system. Instantiating the matrix and naïvely solving it using the Jacobi method with nested loops would not be scalable for very large systems.

We are only interested in the non-zero values in the matrix since zero values simply indicate that no dependency exists. The SQL join operation is highly optimized to pair matching

entries from one relation with another. The author has developed a multiplication algorithm in a SQL query that multiplies two logical matrices [4]. It is especially efficient for sparse matrices because null entries are not processed. This query is used in the computation of the steady state vector without requiring data to be moved outside the database. Equally important is that this method does not require an instantiated Dependency matrix. This saves on space and the computation of the cross-tab query presented in Section 5.3.

Matrix elements are represented by a relation Matrix(Row, Col, Data) where each matrix is given a different relation name. The query to multiply two matrices is given in relational algebra notation as Algorithm 1.

Algorithm 1 MatrixProduct: Compute the product of two matrices

Input: Two relations $A(Row, Col, Data)$ and $B(Row, Col, Data)$ where A represents an $m \times n$ matrix and B represents an $n \times p$ matrix. All non-specified matrix elements are assumed to be zero.
Output: A relation $product(Row, Col, Data)$ that represents the product of the two matrices. All non-specified matrix elements are assumed to be zero.

$product \leftarrow {}_{A.Row,B.Col}\mathcal{G}_{\mathbf{sum}(A.Data \cdot B.Data)}(\sigma_{A.Col=B.Row}(A \times B))$
return $product$

In SQL, Algorithm 1 is implemented as:

Query 8. *MatrixProduct*

```
SELECT A.Row, B.Col, Sum(A.Data*B.Data) AS Data
FROM A INNER JOIN B ON A.Col = B.Row
GROUP BY A.Row, B.Col;
```

While the values for Row and Col make the most conceptual sense when they are the row and column numbers of the matrices to be multiplied, they need not adhere to this condition. For example, in the AssetRank system they are the unique identification numbers of the assets.

5.5.2 Solving the system

Solving the system using the Jacobi method involves repeatedly iterating over $X_i = \delta D X_{i-1} + (1-\delta)\overline{IV}$. Algorithm 2 accomplishes this by specifying an initial value of 1 for each asset in line 2 by filling the vector X_0 with the ProviderIDs and assigning a Rank of 1 to each.

The work of the algorithm is done in lines 4–8. The matrix represented by NormalizedDependency is multiplied by the vector X_i. Algorithm 1 (MatrixProduct) is used to perform

the multiplication but with a simple modification to multiply by a vector instead of a matrix. The result of the multiplication is crossed with InstanceNormalizedValue in a natural right outer join. From this join we compute a Rank column that adds the rank computed in the multiplication to the intrinsic asset value where each of the components of this sum is adjusted by a damping factor. The projection operator returns only the Row and Rank attributes which are then stored in the next iteration of the vector, namely X_{i+1}. With each iteration the vector will get closer to the actual AssetRank values.

After each iteration, the current vector X_i is compared to the previous vector X_{i-1} using Algorithm 3. When the two vectors are equal, modulo a specified tolerance, the final vector is returned along with a count of the number of iterations required to reach the steady state.

Algorithm 2 SolveSystem: Compute Steady State Iteratively

Input: A relation NormalizedDependency(DependentID, ProviderID, NormalizedWeight), and a damping factor δ.
Output: The count i of the number of iterations required to reach the steady state and the eigenvector corresponding to the eigenvalue 1.
Note: The input relation will be abbreviated as D, MatrixProduct as M, NormalizedValue as NV, and InstanceNormalizedValue as $\overline{IV}$.

1: [Initialization]
2: $X_0 \leftarrow \Pi_{pId,\rho_{Rank/1}}(D)$
3: $i \leftarrow 0$
4: [Main loop]
5: **repeat**
6: $\quad X_{i+1} \leftarrow \Pi_{I.Row,\rho_{Rank/(\delta M.Rank+(1-\delta)NV)}}(M(D, X_i) \bowtie\!\sqsubset \overline{IV})$
7: $\quad i \leftarrow i + 1$
8: **until** CompareVectors(X_i, X_{i-1})
9: [Finish]
10: **return** $i,\ X_i$

5.6 Performance

Performance testing was done to verify the scalability of AssetRank. All timings were taken on the following computer system:

- Pentium IV 3.2 GHz processor with hyperthreading enabled
- Microsoft Windows XP Professional with Service Pack 2
- Microsoft Access 2003 with Service Pack 2
- 2.5 GB RAM

Algorithm 3 CompareVectors: Compare Two Vectors Within A Tolerance

Input: Two vectors $\mathbf{x}$ and $\mathbf{y}$ of equal size.
Output: True if the vectors are within the tolerance, and False otherwise.
Note: The tolerance ϵ is specified in the initialization.

[Initialization]
$\epsilon \leftarrow 0.00001$

[Main loop]
for each element $x \in \mathbf{x}$ and $y \in \mathbf{y}$ **do**
  if $|x - y| \geq \epsilon$ **then**
    return False

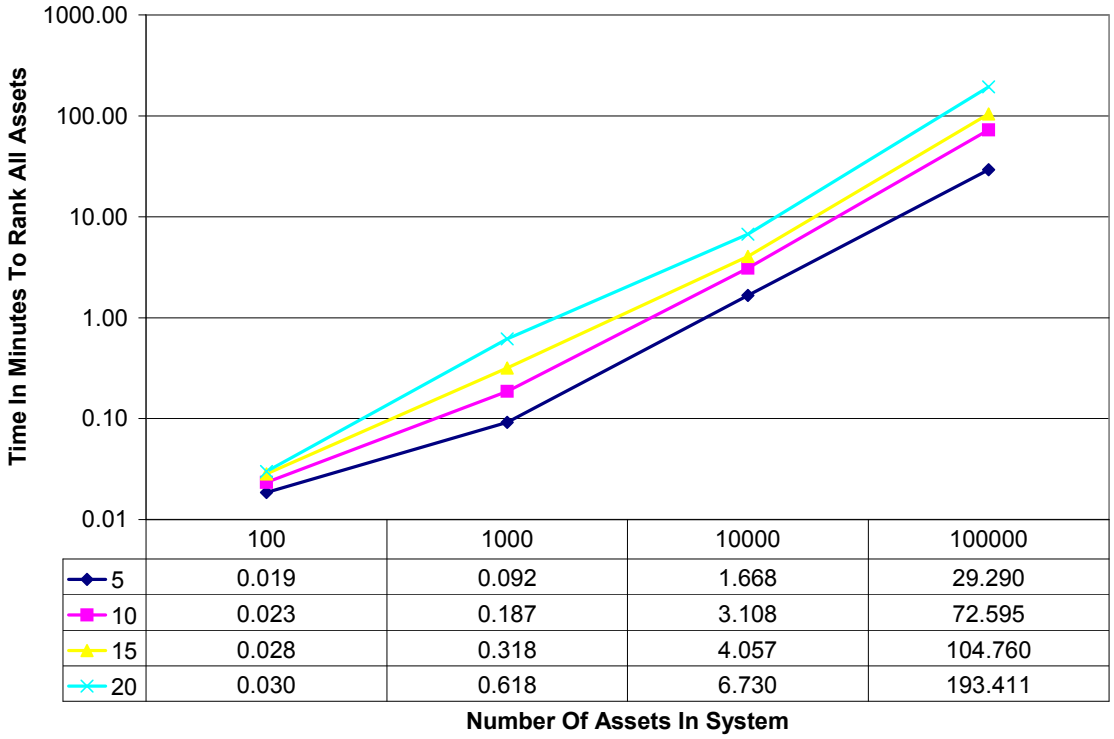
[Finish]
return True

Since Microsoft Access does not have the capability to return process time, clock time was used for the timings and the host was otherwise left idle. We simulated systems by randomly generating dependency matrices. Twenty different system sizes were tested as illustrated in Figure 5. For each system size, ten random systems were generated; hence, two hundred random systems in all were tested. For each system size, the computation times collected across the ten systems were averaged as were the number of iterations required for the Markov chain to reach stability (Table 2). Random dependency matrices generally do not have clusters of interconnected assets so we expect that real asset systems will stabilize faster than the random systems presented here. This conjecture is supported by evidence given in Section 6 and other systems we have tested.

Number Of Assets	Maximum Dependencies Per Asset			
	5	10	15	20
100	35	43	45	45
1,000	36	43	46	47
10,000	38	44	47	48
100,000	38	45	50	50

Table 2: *Iterations Required For Randomly Generated Systems*

The results clearly show that the solution is scalable since even desktop database software can reason on 100,000 assets with up to five dependencies each in under 30 minutes. Large performance gains are expected with a move to a dedicated SQL server with optimized stored procedures. Based upon the data we obtained in [4], the expected time to value the same size of system is 2 minutes.



	100	1000	10000	100000
5	0.019	0.092	1.668	29.290
10	0.023	0.187	3.108	72.595
15	0.028	0.318	4.057	104.760
20	0.030	0.618	6.730	193.411

Figure 5: *Performance Testing Up To 100,000 Assets*

6 Examples

We will work through two examples that illustrate the AssetRank algorithm. The first is a small military system and the second is the set of static dependencies in Microsoft Office 2003.

6.1 Toy Military System

Military organizations, and in fact all large enterprises, depend upon all of the categories given in Figure 3. The following scenario is extremely simplistic. This is done so that it is clear how one change in weighting affects the entire system. Even though the scenario is completely stripped down and only deals with the top few categories, there are still seventeen assets and twenty-eight dependencies.

In this scenario, the Department of National Defence (DND) depends upon two missions — Mission A and Mission B. Mission A depends upon only an officer and Mission B depends upon only a cadet. The officer and cadet depend upon similar types of information

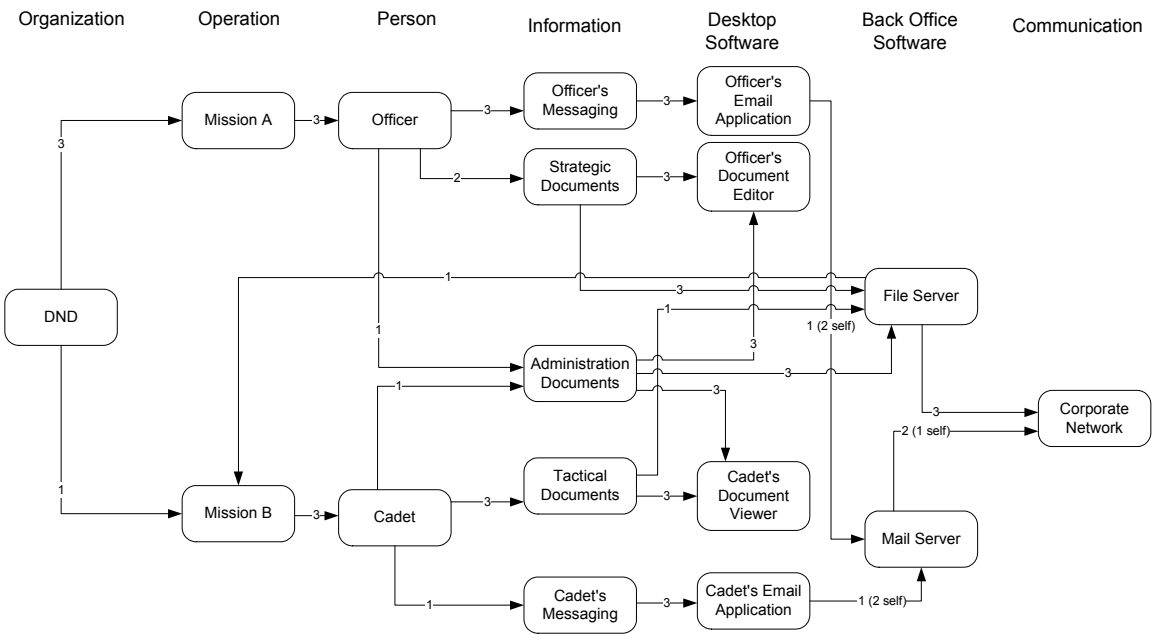


Figure 6: *Toy Military System*

but at different weights. The information depends upon desktop software which in turn depends upon back office software. The scenario is hierarchical except for one backlink which adds a low dependency from the File Server to Mission B. The back office software depends upon the corporate network.

There are several objects that do not have a complete dependency on other objects. That is, they would have some value even if the objects they depend upon were unavailable for some time. For example, the email applications are stated in the figure to have a low dependency upon the mail server. This assumes that the email client can still read emails that have previously been downloaded and can queue replies and new emails until such time as the mail server becomes available again.

We will look at this scenario in more detail. Let a_1 be the cadet's email application, a_2 be the mail server, and a_{n+1} be the asset added to compensate for sink assets. Since a_1 has a low dependence on a_2 the arc $(a_1, a_2, 1)$ is added to the arc set A. It is tempting to represent the fact that a_1 has utility while disconnected from a_2 by adding the arc $(a_1, a_1, 2)$ to A. Once the column is normalized this would have the effect of a_1 having a $1/3$ dependence upon a_2 and a 2/3 dependence upon itself.

However, the effect would be to inappropriately concentrate value in a_1 just as a sink asset would inappropriately concentrate value if it only linked to itself. This partial dependence is resolved in the same manner as it was for the sink asset. Rather than adding a dependence to itself, a dependence to the sink asset compensator is added instead. Hence, the

dependencies of a_1 are given by the set of arcs $\{(a_1, a_2, 1), (a_1, a_{n+1}, 2)\}$.

The data in the Dependency table that represents the system presented in Figure 6 is given in Table 3. The data shown is presented with its labels for human-readability. In the database table the Dependent and Provider attributes are integer unique identifiers.

Dependent	Provider	Weight
Administration Documents	Cadet's Document Viewer	3
Administration Documents	Officer's Document Editor	3
Administration Documents	File Server	3
Cadet	Cadet's Messaging	1
Cadet	Administration Documents	1
Cadet	Tactical Documents	3
Cadet's Email Application	Sink	2
Cadet's Email Application	Mail Server	1
Cadet's Messaging	Cadet's Email Application	3
DND	Mission B	1
DND	Mission A	3
File Server	Corporate Network	3
File Server	Mission B	1
Mail Server	Corporate Network	2
Mail Server	Sink	1
Mission A	Officer	3
Mission B	Cadet	3
Officer	Officer's Messaging	3
Officer	Strategic Documents	2
Officer	Administration Documents	1
Officer's Email Application	Mail Server	1
Officer's Email Application	Sink	2
Officer's Messaging	Officer's Email Application	3
Strategic Documents	File Server	3
Strategic Documents	Officer's Document Editor	3
Tactical Documents	Cadet's Document Viewer	3
Tactical Documents	File Server	1

Table 3: *Toy System Dependency data*

As discussed in Section 5.5.1, the dependency data does not need to take the shape of an actual matrix. Still, humans find it easier to understand the process when presented with the matrix in a visual format. The result of the cross-tab query is too wide to display in its entirety; however, Figure 7 gives a sampling of the result. The assets listed along the header row depend upon the assets listed in the first column with the weight specified in their intersection.

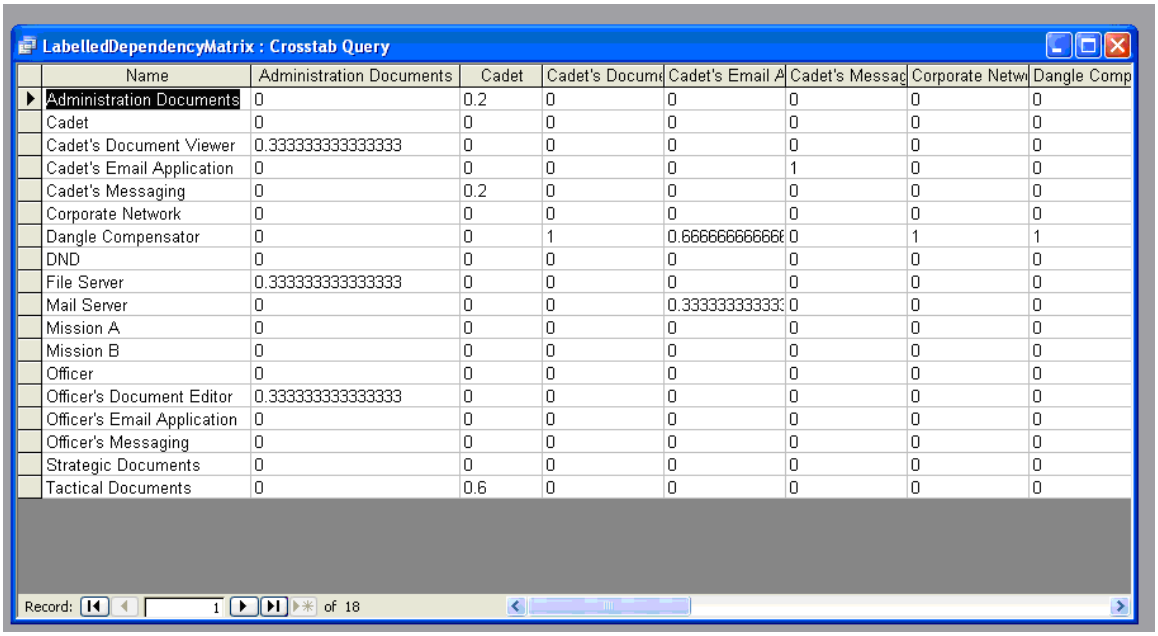


Figure 7: *Results of LabelledDependencyMatrix query*

To calculate the AssetRank of the assets in the system we run the data through Algorithm 2. After 18 iterations the system stabilized under a damping factor of 0.85. The results are presented in Table 4.

One immediately notices that the asset compensating for sink assets accumulates a lot of value. However, its actual value is not important, it only serves to allow the correct relative ranking of the real assets. The sink compensator asset may be ignored or its value may be distributed amongst the real assets.

The AssetRank numbers themselves should not be paid any attention. The purpose of AssetRank is to obtain a relative ranking within a category. The scheme finds importance on a system-wide basis. The administrator cannot obtain any information about the importance of an asset by knowing only its AssetRank value in isolation.

The information in the table is presented with the instances from each category grouped together. This is done because the rankings do not make sense unless similar items are being compared. In this example we see that the cadet is ranked higher than the officer. At first glance this may seem surprising because the organization has placed a high weight on Mission A, which in turn relies upon the officer, while it has placed a low weight on Mission B. With more reflection, one will notice that Mission A depends upon the file server which in turn depends upon Mission B. Thus, some valuation will flow from Mission A to Mission B causing the cadet to rank higher than the officer.

Name	Category	Rank
File Server	Back Office Software	0.404859
Mail Server	Back Office Software	0.359437
Corporate Network	Communication Path	0.611779
Cadet's Document Viewer	Desktop Software	0.443527
Officer's Email Application	Desktop Software	0.407110
Cadet's Email Application	Desktop Software	0.332081
Officer's Document Editor	Desktop Software	0.332048
Tactical Documents	Information	0.342638
Officer's Messaging	Information	0.302482
Administration Documents	Information	0.265040
Strategic Documents	Information	0.251655
Cadet's Messaging	Information	0.214213
Mission B	Operation	0.267908
Mission A	Operation	0.245625
DND	Organization	0.150000
Cadet	Person	0.377722
Officer	Person	0.358781
Sink	Sink Compensator	12.333096

Table 4: *Toy System AssetRanks*

It is important to understand the information that the rankings are able to provide. They do not give an aggregate importance of an asset to the organization, they only give the relative dependence the organization puts on a specific asset in a specific context. Consider the officer and cadet. The system shows how much each person is being depended upon for system availability. In the example scenario, the cadet is being depended upon to a higher degree than the officer. This information allows management to decide if the situation should remain as it is or to change it. Perhaps they will decide that the position is too important to have a cadet there.

If we remove the dependence of the file server on Mission B, then the cadet will be ranked lower as can be seen in Table 5. Notice how removing this one dependency has revalued all of the assets in the system. Not all of them have changed their relative ranking but assets in the operation and person categories have moved rankings significantly. As a point of interest, the new system required only 8 iterations to stabilize.

If an analyst determined that the officer had an intrinsic value that was not being represented accurately in the rankings, he could adjust his intrinsic value. In our next example we have given the officer an intrinsic value of 2 while all other assets have a default value of 1. The dependency from the file server to Mission B has been restored to that shown in Figure 6. The results are shown in Table 6.

Name	Category	Rank
File Server	Back Office Software	0.393411
Mail Server	Back Office Software	0.356443
Corporate Network	Communication Path	0.686384
Cadet's Document Viewer	Desktop Software	0.416228
Officer's Email Application	Desktop Software	0.407110
Officer's Document Editor	Desktop Software	0.328526
Cadet's Email Application	Desktop Software	0.321514
Tactical Documents	Information	0.305343
Officer's Messaging	Information	0.302482
Administration Documents	Information	0.252608
Strategic Documents	Information	0.251655
Cadet's Messaging	Information	0.201781
Mission A	Operation	0.245625
Mission B	Operation	0.181875
DND	Organization	0.150000
Officer	Person	0.358781
Cadet	Person	0.304594
Sink	Sink Compensator	12.535641

Table 5: *Toy System AssetRanks with an altered dependency*

Before giving the officer intrinsic value, the cadet's primary dependence, namely Tactical Documents, was ranked higher in the information category than the officer's primary dependence, Officer's Messaging. After giving the officer high value, Officer's Messaging ranked higher than Tactical Documents. The system dynamically distributed the officer's value to his dependencies.

While we would like to stress test AssetRank on a large military system, the dependencies of a large system are subjectively determined without a concrete model. Without an agreed upon model, it is difficult for a team to develop a valid dependency design. By extension, any asset valuation that is based upon a flawed, incomplete, or inaccurate model will also be inaccurate. Research is needed in this area to design a comprehensive model that will accurately and non-subjectively detail the dependencies of organizations. Even the much simpler problem of determining the dependencies of a computer network is subject to bias. One individual would view the network from a physical perspective while another will list the logical dependencies. The author's view is that logical dependencies will provide the most accurate ranking of assets. The differences between these two example paradigms is quickly seen if one maps the dependencies of the network infrastructure.

Name	Category	Rank
File Server	Back Office Software	0.407030
Mail Server	Back Office Software	0.355238
Corporate Network	Communication Path	0.602889
Officer's Email Application	Desktop Software	0.437018
Cadet's Document Viewer	Desktop Software	0.427470
Officer's Document Editor	Desktop Software	0.337592
Cadet's Email Application	Desktop Software	0.315216
Officer's Messaging	Information	0.346957
Tactical Documents	Information	0.326768
Strategic Documents	Information	0.278673
Administration Documents	Information	0.271943
Cadet's Messaging	Information	0.203659
Mission B	Operation	0.258797
Mission A	Operation	0.232697
DND	Organization	0.142105
Officer	Person	0.482003
Cadet	Person	0.362083
Sink	Sink Compensator	12.211862

Table 6: *Toy System AssetRanks with high officer value*

6.2 File valuation in MS Office 2003

In the interest of rapidly testing AssetRank on a large system we will avoid the dependency paradigm questions and work with the concrete file dependencies of Microsoft Office 2003. This will accomplish two things, it removes the analyst's bias in determining dependencies and gives a much more complicated test in terms of the algorithm's workload.

The static dependencies of Microsoft Office 2003 were found by using the Dependency Walker software tool [9]. Static dependencies are the drivers, libraries, and other files that are loaded every time the application is invoked. Files and libraries that are not loaded every time the application launches were not included because they depend upon which features of the program are used.

There were a total of 136 assets in the system including an Officer. The Officer depends upon the applications EXCEL.EXE, MSACCESS.EXE, OUTLOOK.EXE, POWERPNT.EXE, and WINWORD.EXE. There are a total of 1,239 interdependencies in the system and four sink assets[6]. Using a damping factor of 0.85, AssetRank calculated (in less than a second) the top 20 assets to be those given in Table 7[7]. Thus, the NTDLL.DLL and KERNEL32.DLL libraries are easily the most critical files in Microsoft Office.

[6]The sink assets were DWMAPI.DLL, MSMAPI32.DLL, MSO.DLL, and NTDLL.DLL.

[7]The sink compensator had a value of 72.9 but was removed.

Name	Rank
NTDLL.DLL	12.15
KERNEL32.DLL	7.67
ADVAPI32.DLL	3.75
USER32.DLL	3.07
MSVCRT.DLL	2.75
RPCRT4.DLL	1.90
GDI32.DLL	1.85
OLE32.DLL	1.17
SECUR32.DLL	0.98
WINTRUST.DLL	0.86
NETAPI32.DLL	0.80
OLEAUT32.DLL	0.69
MSIMG32.DLL	0.60
WINSTA.DLL	0.58
SHLWAPI.DLL	0.57
POWRPROF.DLL	0.54
SHELL32.DLL	0.50
VERSION.DLL	0.48
WS2_32.DLL	0.46
SETUPAPI.DLL	0.44

Table 7: *Highest 20 ranked dependencies of Microsoft Office 2003*

With this information an administrator knows which files in the system are critical. These files would also be likely files to be compromised by an informed attacker.

7 Dependency templates

It is not realistic to request that the dependencies of every asset be determined on an individual basis. This would be an overwhelming roadblock to the adoption of the system and would not be maintainable. To be useful in practice, the dependencies of assets should be determined by experts at the class level. The various experts should determine the dependencies in a uniform manner. Methods to accomplish this are the subject of future research.

Figure 8 updates Figure 4 to include the Class and DependencyTemplate relations. An expert will determine, for example, that a researcher has dependencies upon various types of information. Perhaps the same expert will also determine the dependencies that each class of information has but this will be specified as an information dependency, not a research dependency. Similarly, an expert will determine the dependencies of a battle company and another will ascertain the dependencies of particular classes of vehicles. In each case, the expert only determines the direct dependencies and leaves the second and third level de-

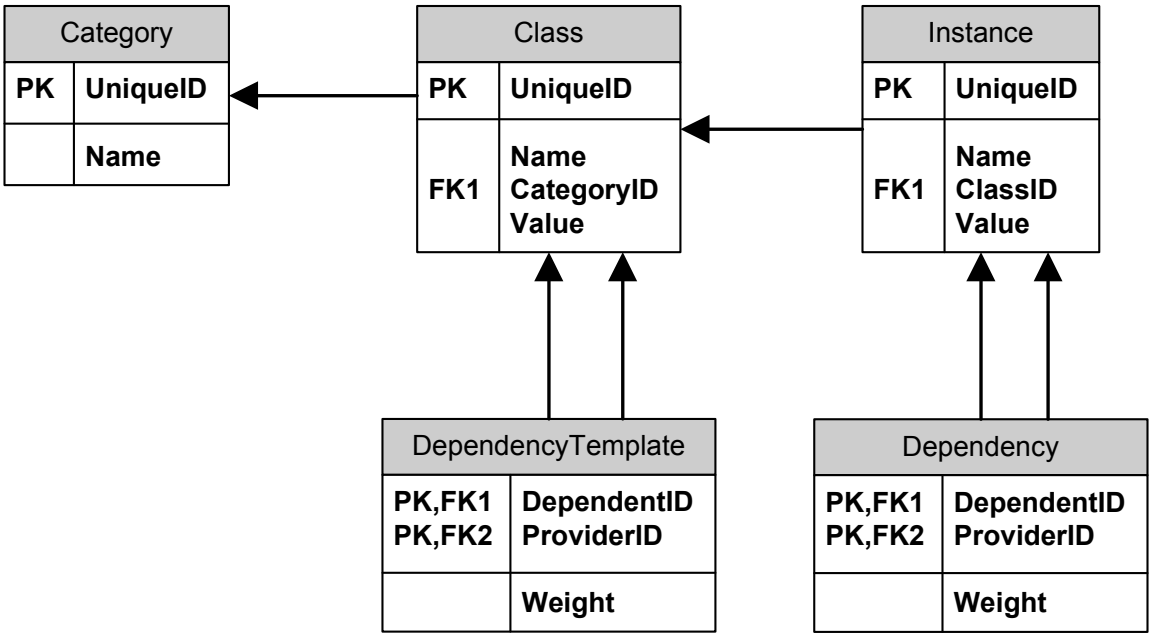


Figure 8: *Entity-Relationship diagram with templates*

pendencies to be specified as direct dependencies in their respective templates. This way, the level of effort is minimized while accuracy and template reuse is maximized.

Implementation details will be worked out in a formal development process; however, the following requirements are proposed:

1. Instances will not copy DependencyTemplate information but will link to it instead. This allows updates to template information to be immediately reflected in the system.

2. The ability to override template data should be included. The system should use template data except where particular dependencies have been overridden.

3. When determining classes and dependencies, do so at an appropriate level of detail. It is easy to become so distracted by the trees that one cannot see the forest.

4. When classes are instantiated, dependencies on some classes will create instances of those classes while in other cases, a single specific asset will be depended upon by all instances. For example, a template with a dependence upon a desktop software asset will create a seperate instance of the asset since each user has a personal copy of the software installed on their PC. Conversely, a dependence upon a back office software asset will not create a separate instance of the asset since the back office software asset is shared.

One way to manage this is to change the DependencyTemplate relation to allow the ProviderID to be a data element either from Class.UniqueID or Instance.UniqueID. If the ProviderID is a Class.UniqueID, then a new instance of the provider will be created for the dependent; otherwise, the dependency will be upon an existing Instance.UniqueID.

8 Future work

Testing on enterprise systems will validate the suitability of using AssetRank to ascertain a ranked list of critical assets. The PageRank algorithm clearly works for Google; AssetRank aims to extend PageRank beyond web pages to all assets. Enterprise testing will reveal the best damping factor, dependency weights, and intrinsic values to use in organizational use.

PageRank is only one of many citation and web page ranking algorithms. We wish to see if other algorithms can be adapted to use a dependency topology and compare their ranking to that generated by AssetRank. The purpose in finding critical assets is to insulate against losing the asset, or minimizing the impact of such a loss. To this end, AssetRank may best be used in combination with an impact assessment tool like that developed by DRDC [10].

AssetRank only tells the relative dependence the organization puts on a specific asset in a specific context. One asset will have a different AssetRank in each of its contexts. The system does not give an aggregate importance of the asset to the organization. Just as Google incorporates the PageRank value into other algorithms to finally give an answer to a search query, the AssetRank value given for each context should be combined together and then incorporated with other vectors to answer the organization's informational needs.

Asset dependencies in a large system will be specified by many experts in diverse disciplines. Research is needed to enumerate the different choices available to specify dependencies. An analysis of the advantages and disadvantages of each method will lead to an informed decision on the method to use in each context. The next challenge after choosing a method is training experts to make uniform choices in weighting and intrinsic value.

The article [7] introduces communities of web pages. It would be interesting to see how this idea could be applied to assets. Research is needed to determine the best way to combine AssetRanks of communities of assets so that communities of equal function but different sizes may be effectively compared. This information could be used to determine the critical communities of the organization.

9 Conclusion

We have presented a method that quickly and dynamically calculates a relative value for all assets in an organization in any context in which dependencies may be specified. The idea is a novel application of the Google PageRank algorithm that extends it to all assets and all contexts. In fact, web pages become a special case of asset ranking in a functionality context. Additionally, our algorithm provides the capability to specify individual dependency weights that one asset has upon another and individual intrinsic values for each asset. An asset category dependency model is proposed and entity-relationsip structures are given to assist in continued development.

References

[1] Burrell, C., Lefebvre, J., and Treurniet, J. (2006), Dynamic Defensive Posture for Computer Network Defence, (DRDC Ottawa TM 2006-250) Defence R&D Canada – Ottawa.

[2] CERT/CC Statistics 1988-2006 (online), http://www.cert.org/stats/cert_stats.html (Access Date: October, 2006).

[3] Stewart, J.M., Tittel, E., and Chapple, M. (2005), CISSP: Certified Information Systems Security Professional Study Guide, Alameda, CA, USA: SYBEX Inc.

[4] Sawilla, R. and Vandenberghe, G. (2006), High Speed Matrix Arithmetic Leveraging Database Technology. Submitted for publication.

[5] Xinming, O., Govindavajhala, S., and Appel, A. (2005), MulVAL: A Logic-based Network Security Analyzer, In *14th USENIX Security Symposium*, pp. 113–128, Baltimore.

[6] Page, L., Brin, S., Motwani, R., and Winograd, T. (1998), The PageRank Citation Ranking: Bringing Order to the Web, Technical Report Stanford Digital Library Technologies Project.

[7] Bianchini, M., Gori, M., and Scarselli, F. (2005), Inside PageRank, *ACM Trans. Inter. Tech.*, 5(1), 92–128.

[8] Silberschatz, A., Korth, H.F., and Sudershan, S. (1998), Database System Concepts, 3rd ed, New York, NY, USA: McGraw-Hill, Inc.

[9] Dependency Walker (online), http://www.dependencywalker.com (Access Date: August, 2006).

[10] Beaudoin, L. (2005), Impact Assessment Tool, (Presentation) Defence R&D Canada – Ottawa.

Distribution list

DRDC Ottawa TM 2006-243

Internal distribution

2 Author

1 Section Display

1 Library

1 Luc Beaudoin

1 Craig Burrell

1 Daniel Charlebois

1 Gregory Van Bavel

Total internal copies: 8

External distribution

1 LCol Chevalier (CO CFNOC)
CFS Leitrim
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 Luc Dandurand
Cyber Defence Futures (CSE)
Edward Drake Building
1500 Bronson Avenue
Ottawa, Ontario K1G 3Z4

1 Col Davies (DMGSP)
Pearkes Building, 10CBN, F006
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 Paul Desmier (DMGOR)
Pearkes Building, 10CBS, S006
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 Director General (DGIMSP)
Pearkes Building, 8th Floor
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 Col Gallant (DIMEI)
Tunney's Pasture, 2, F008
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 Col Jackson (Dir IM Secur)
Tunney's Pasture, 3rd Floor, F38
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 Michel Lessard (DGIMT)
Tunney's Pasture, 4, F007
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 Col Mazzolin (DLCSPM)
LSTL, 4 EK03
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

1 LCol Viens (DIMTPS 2)
Tunney's Pasture, 3rd Floor, A024
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

Total external copies: 10

Total copies: 18

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

   Defence R&D Canada – Ottawa
   3701 Carling Avenue, Ottawa, Ontario, Canada
   K1A 0Z4

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

   UNCLASSIFIED

   (NON-CONTROLLED GOODS)
   DMC A
   REVIEW: GCEC December 2012

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

   Abstracting PageRank To Dynamic Asset Valuation

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

   Sawilla, R.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
December 2006	46	10

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

   Technical Memorandum

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

   Defence R&D Canada – Ottawa
   3701 Carling Avenue, Ottawa, Ontario, Canada K1A 0Z4

9a. PROJECT NO. (The applicable research and development project number under which the document was written. Please specify whether project or grant.)	9b. GRANT OR CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
15BO02	

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Ottawa TM 2006-243	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

   (X) Unlimited distribution
   ( ) Defence departments and defence contractors; further distribution only as approved
   ( ) Defence departments and Canadian defence contractors; further distribution only as approved
   ( ) Government departments and agencies; further distribution only as approved
   ( ) Defence departments; further distribution only as approved
   ( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

We present a method that quickly and dynamically calculates a relative value for all assets in an organization in any context in which dependencies may be specified. The idea is a novel application of the Google PageRank algorithm that extends it to all assets and all contexts. In fact, web pages become a special case of asset ranking in a functionality context. Additionally, our algorithm provides the capability to specify individual dependency weights that one asset has upon another and individual intrinsic values for each asset. An asset category dependency model is proposed and entity-relationship structures are given to assist in continued development.

Our scheme works in general and will provide asset valuation in any context, be it confidentiality, integrity, availability, or even political capital. In this document we combine the idea of asset dependency with an established web page ranking technique. Google's PageRank algorithm is generalized in two key areas, creating an algorithm we call AssetRank. It calculates a relative importance ranking for every asset in the organization from information to tanks to partners. The ranking will assist decision makers by highlighting which assets are the most highly depended upon by the organization.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Dynamic Asset Valuation, Critical Assets, Dependency

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca