University of Crete
Computer Science Department

On Power Laws and the Semantic Web

Yannis Theoharis

Master's Thesis

Heraklion, February 2007

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Σχετικά με τους Κανόνες της Δύναμης και το Σημασιολογικό Ιστό

Εργασία που υποβλήθηκε απο τον

Ιωάννη Φ. Θεοχάρη

ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Ιωάννης Θεοχάρης, Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Βασίλης Χριστοφίδης, Αναπληρωτής καθηγητής, Επόπτης

Παναγιώτης Τσακαλίδης, Αναπληρωτής καθηγητής, Μέλος

Γεώργιος Γεωργακόπουλος, Επίκουρος καθηγητής, Μέλος

Δεκτή:

Παναγιώτης Τραχανιάς, Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Φεβρουάριος 2007

On Power Laws and the Semantic Web

Yannis Theoharis

Master's Thesis

Computer Science Department, University of Crete

Abstract

Semantic Web (as the *WWW* itself) can be seen as a decentralized system that self organizes and evolves, scaling to unforeseen conditions, features which are typical of *complex systems*. A big amount of *Semantic Web* (*SW*) schemas expressed in either *RDFS* [11] or *OWL* [18] has been developed during the last years [43]. In their majority, they are not specified at the same level of detail and hence, only few classes appear as domain/range of many properties, while most appear as domain/range of few or none. Furthermore, they usually form interconnected graphs as a result of a social collaboration process, which involves the *reuse* and *extension* of the classes and properties defined in different schemas. In this setting, it would be interesting to investigate to what extent graph features that emerge in social network analysis, such as *power-law* degree distributions and the *small world* phenomenon, could be used to grasp the morphology of existing *SW* schemas.

The knowledge of these features is essential in several contexts. For instance, it can be exploited for selecting or devising efficient index structures and search algorithms [1] or it can be exploited for ontology visualization [50]. Furthermore, it can be useful for revealing emerging conceptual modeling habits. Finally, it can be used for guiding synthetic SW data generation in order to benchmark *SW* repositories and query languages implementations in a credible manner. For instance, for developing ontology-based repositories that will be able to cope with the expected size of the

Semantic Web in the coming years, we need to be able to create large datasets and test now the scalability of storage, query or update methods.

This Thesis consists of two parts. The former focuses on the investigation of graph features of real SW schemas, while the latter on the generation of synthetic SW schemas that exhibit those features. Concerning the former part, the well-known graph mining techniques cannot be used as such, since SW schemas are graphs enriched with the semantics of RDF/S [11] or OWL [18] specifications. In particular, arcs in these graphs are of different nature, namely, a) arcs representing subsumption relationships among classes, and b) arcs representing relations between classes (e.g. *has_a*) or attributes (e.g. *title*), collectively called properties. The existence of arcs of the former kind implies additional arcs of the latter one, e.g., a class inherits the properties of its ancestors. Hence, for each SW schema we essentially need to study two graphs that have the same set of nodes (i.e., classes or literal types), namely, the *subsumption*, and the *property* graph. Among the results of our experimental analysis, we briefly mention that the total-degree distribution of the *property* graph as well as the class descendants distribution of the *subsumption* graph of real SW schemas follow a *power-law*. Moreover, the *property* graph of the their majority exhibits the *small world* phenomenon.

Concerning the latter part of this Thesis, i.e., the generation of synthetic SW schemas whose *property* and *subsumption* graphs exhibit the features observed in the real SW schemas, the main challenge that was faced was the generation of the subsumption graph given the in- and out-degree sequence of its *transitive closure*. This implies the generation of *transitively closed* graphs given their in- and out-degree sequences, problem that has not been studied before in the literature. We present a reduction of this problem to the *Linear Programming* one.

Supervisor: Vassilis Christophides

Associate Professor

Σχετικά με τους Κανόνες της Δύναμης και το Σημασιολογικό Ιστό

Ιωάννης Θεοχάρης

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Περίληψη

Ο Σημασιολογικός Ιστός (ΣΙ) (όπως και ο παγκόσμιος ιστός) μπορεί να θεωρηθεί σαν ένα αποκεντρωμένο σύστημα που αυτό-οργανώνεται και εξελίσσεται κάτω από απρόβλεπτες συνθήκες, χαρακτηριστικά που αποτελούν τυπικές ιδιότητες των πολύπλοκων συστημάτων. Ένας μεγάλος αριθμός σχημάτων του ΣΙ εκφρασμένων είτε σε RDFS ή σε OWL έχουν αναπτυχθεί τα τελευταία χρόνια. Στην πλειοψηφία τους, δεν έχουν αναλυθεί στον ίδιο βαθμό με αποτέλεσμα λίγες κλάσεις να εμφανίζονται σαν το πεδίο ορισμού/τιμών πολλών ιδιοτήτων, ενώ οι περισσότερες να εμφανίζονται σαν το πεδίο ορισμού/τιμών λίγων ή και καμίας ιδιότητας. Επιπλέον, συνήθως σχηματίζουν διασυνδεδεμένους γράφους σαν το αποτέλεσμα μιας διαδικασίας κοινωνικής συνεργασίας, η οποία περιλαμβάνει την επαναχρησιμοποίηση και επέκταση κλάσεων και ιδιοτήτων ορισμένων σε διαφορετικά σχήματα. Μέσα σ΄ αυτά τα πλαίσια είναι ενδιαφέρουσα η εξακρίβωση του βαθμού στον οποίο γνωρίσματα γράφων που αναδύονται στην ανάλυση των κοινωνικών δικτύων, όπως οι κατανομές κανόνα της δύναμης για τους ολικούς βαθμούς και το φαινόμενο του μικρού κόσμου, μπορούν να χρησιμοποιηθούν για να ¨πιαστεί' η μορφολογία των υπαρχόντων σχημάτων του ΣΙ.

Η γνώση αυτών των γνωρισμάτων είναι ζωτική για διάφορους λόγους. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για την επιλογή ή την επινόηση αποδοτικών δεικτών και αλγόριθμων αναζήτησης, όπως και για την οπτικοποίηση των οντολογιών. Επιπλέον,

μπορεί να χρησιμοποιηθεί για την αποκάλυψη αναδυόμενων συνηθειών εννοιολογικής μοντελοποίησης. Τέλος, μπορεί να χρησιμοποιηθεί για την καθοδήγηση της δημιουργίας συνθετικών δεδομένων του ΣΙ με σκοπό την πειραματική αξιολόγηση επιδόσεων αποθηκών δεδομένων και υλοποιήσεων γλωσσών επερωτήσεων του ΣΙ με αξιόπιστο τρόπο. Για παράδειγμα, για την ανάπτυξη οντολογικο-κεντρικών αποθηκών δεδομένων που θα είναι ικανές να ανταπεξέλθουν στο αναμενόμενο μέγεθος του ΣΙ τα επόμενα χρόνια, χρειάζεται να είμαστε ικανοί να παράγουμε μεγάλα σύνολα δεδομένων και να ελέγξουμε τώρα την κλιμακοσιμότητα των μεθόδων αποθήκευσης, επερώτησης και ενημέρωσης.

Η εργασία αυτή χωρίζεται σε 2 μέρη. Το πρώτο εστιάζει στην εξακρίβωση των γνωρισμάτων των γράφων των σχημάτων του ΣΙ, ενώ το δεύτερο στη δημιουργία συνθετικών σχημάτων που εμφανίζουν αυτά τα γνωρίσματα.

Όσον αφορά το πρώτο μέρος, οι γνωστές τεχνικές εξόρυξης γράφων δε μπορούν να χρησιμοποιηθούν αυτούσιες, αφού τα σχήματα του ΣΙ είναι γράφοι εμπλουτισμένοι με τη σημασιολογία της RDFS ή της OWL. Συγκεκριμένα, οι ακμές σ΄ αυτούς τους γράφους είναι διαφορετικής φύσης, δηλαδή α) ακμές που αναπαριστούν σχέσεις υπαλληλίας μεταξύ κλάσεων και β) ακμές που παριστάνουν σχέσεις ανάμεσα σε κλάσεις ή γνωρίσματα, από κοινού ονομαζόμενες ως ιδιότητες. Η ύπαρξη ακμών του πρώτου είδους συνεπάγεται επιπρόσθετες ακμές του δεύτερου, π.χ. μια κλάση κληρονομεί τις ιδιότητες των προγόνων της. Επομένως, για κάθε σχήμα του ΣΙ είναι ανάγκη να μελετήσουμε δύο γράφους που έχουν το ίδιο σύνολο κόμβων (δηλαδή το σύνολο των κλάσεων και των φιλολογικών τύπων), το γράφο της επαλληλίας και το γράφο των ιδιοτήτων. Ανάμεσα στα αποτελέσματα της πειραματικής μας ανάλυσης εν συντομία αναφέρουμε ότι η κατανομή των ολικών βαθμών του γράφου των ιδιοτήτων όπως και η κατανομή των απογόνων των κλάσεων του γράφου της επαλληλίας των σχημάτων του ΣΙ ακολουθούν έναν κανόνα της δύναμης. Επιπλέον, ο γράφος των ιδιοτήτων της πλειοψηφίας των σχημάτων εμφανίζει το φαινόμενο του μικρού κόσμου.

Όσον αφορά το δεύτερο μέρος της εργασίας αυτής, δηλαδή τη δημιουργία συνθετικών

σχημάτων των οποίων οι γράφοι υπαλληλίας και ιδιοτήτων εμφανίζουν τα γνωρίσματα που παρατηρήθηκαν στα πραγματικά σχήματα του ΣΙ, η μεγαλύτερη πρόκληση που αντιμετωπίσαμε ήταν η δημιουργία του γράφου της επαλληλίας δοσμένων των ακολουθιών των εισερχόμενων και εξερχόμενων βαθμών του μεταβατικού κλεισίματός τους. Αυτό συνεπάγεται τη δημιουργία μεταβατικά κλειστών γράφων δοσμένων των ακολουθιών των εισερχόμενων και εξερχόμενων βαθμών τους, πρόβλημα που δεν έχει μελετηθεί προηγουμένως στη βιβλιογραφία. Στην εργασία αυτή παρουσιάζουμε την αναγωγή του προβλήματος αυτού στο πρόβλημα του Γραμμικού Προγραμματισμού.

Επόπτης Καθηγητής: Βασίλης Χριστοφίδης
Αναπληρωτής Καθηγητής

Στην αδερφή μου, Ειρήνη

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω όλους τους δασκάλους μου από το Δημοτικό έως το Πανεπιστήμιο. Ιδιαιτέρως, τον κ. Θεμιστοκλή Πεπόνη, καθηγητή μαθηματικών των λυκειακών μου χρόνων, και τον κ. Βασίλη Χριστοφίδη επιβλέποντα της πτυχιακής αλλά και της μεταπτυχιακής μου εργασίας. Τον πρώτο για την παρουσίαση των μαθηματικών ώς ζωντανό οργανισμό και για την αποκάλυψη της σχέσης τους με τη φυσική και κοινωνική πραγματικότητα. Τον δεύτερο για την άψογη συνεργασία μας τα τελευταία 3 και πλέον χρόνια, καρπός της οποίας αποτελεί, πέραν πολλών άλλων, και η παρούσα εργασία. Οι γνώσεις και η συνεχής διαθεσιμότητα του για βοήθεια στάθηκαν ιδιαίτερα ευεργετηκές για μένα.

Ιδιαίτερες ευχαριστίες αξίζουν και στους κ. Γιάννη Τζίτζικα και Γιώργο Γεωργακόπουλο. Οι παρατηρήσεις και οι προτάσεις τους απεδείχθησαν παραπάνω από χρήσιμες για την περάτωση της παρούσας εργασίας. Ειδικότερα, θα ήθελα να ευχαριστήσω τον κ. Γιάννη Τζίτζικα για την άψογη συνεργασία μας τα τελευταία 2 χρόνια μέσα από το εργαστήριο πληροφοριακών συστημάτων του ΙΤΕ αλλά και για τις φιλοσοφικές - μεταμαθηματικές συζητήσεις μας.

Θα ήθελα ακόμα να ευχαριστήσω τον κ. Παναγιώτη Τσακαλίδη για την προθυμία του να συμμετάσχει στην εξεταστική επιτροπή της μεταπτυχιακής μου εργασίας.

Ιδιαίτερα θα ήθελα να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστων του Πανεπιστήμιου Κρήτης και το Εργαστήριο Πληροφοριακών Συστημάτων του Ινστιτούτου Πληροφορικής του ΙΤΕ για τις γνώσεις που απέκτησα κατά τη διάρκεια των σπουδών μου στην Κρήτη.

Επίσης θα ήθελα να ευχαριστήσω τους φίλους μου, ειδικά των πρότερων χρόνων, Αντώνη, Δημήτρη, Νίκο, Γιώργο, Μαρίνα, Πανάγο, Αργύρη, αλλά και των ύστερων χρόνων, Γιάννη, Τζώρτζη, Παναγιώτη, Μάρκο, Χριστίνα, Ρόζα. Ιδιαιτέρως, θα ήθελα να ευχαριστήσω την Ιωάννα που ήταν πρόθυμη να απαντήσει στις απορίες μου για τις τεχνικές λεπτομέρειες της μορφοποίησης της παρούσας αναφοράς.

Τελικά και πάνω από όλα θα ήθελα να ευχαριστήσω την αδερφή μου, Ειρήνη, και τους γονείς μου, Φίλιππο και Βέρα, για την υποστήριξη τους όλα αυτά τα χρόνια.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the next evolution step of the Web, termed the Semantic Web [6], vast amounts of information resources (data, documents, programs) will be made available along with various kinds of descriptive information, i.e., metadata. Better knowledge about the meaning, usage, accessibility, validity or quality of web resources will considerably facilitate automated processing of available Web content/services. The Resource Description Framework (*RDF*) [38] enables the creation and exchange of resource metadata as normal Web data. To interpret these metadata within or across user communities, RDF allows the definition of appropriate schema vocabularies (*RDFS*) [11].

Semantic Web (as the *WWW* itself) can be seen as a decentralized system that self organizes and evolves, scaling to unforeseen conditions, features which are typical of *complex systems*. A big amount of *Semantic Web* (*SW*) schemas expressed in either *RDFS* [11] or *OWL* [18] has been developed during the last years [43]. In their majority, they are not specified at the same level of detail and hence, only few classes appear as domain/range of many properties, while most appear as domain/range of few or none. Furthermore, they usually form interconnected graphs as a result of a social collaboration process, which involves the *reuse* and *extension* of the classes and properties defined in different schemas. In this setting, it would be interesting to investigate to what extent graph features that emerge in social network analysis

could be used to grasp the morphology of existing *SW* schemas.

The knowledge of these features is essential in several contexts. For instance, it can be exploited for selecting or devising efficient index structures and search algorithms [1] or it can be exploited for ontology visualization [50]. Furthermore, it can be useful for revealing emerging conceptual modeling habits. Finally, it can be used for guiding synthetic SW data generation in order to benchmark *SW* repositories and query languages implementations in a credible manner. For instance, for developing ontology-based repositories that will be able to cope with the expected size of the Semantic Web in the coming years, we need to be able to create large datasets and test now the scalability of storage, query or update methods.

Generally, *degree sequence*, *diameter* and *average distance* are three main features that characterize graphs. The internet topology [25] as well as *WWW* [37, 5] have been observed to exhibit a *power-law* distribution for their in- and out-degree sequences. Several models of *power-law graphs* evolution have been proposed [5, 9, 37, 36, 2] to capture the way in which such *power-law* distributions arise as graphs evolve (see [3] for a comparison). Furthermore, social network analysis has focused on the investigation of the *small world* phenomenon [53], i.e., graphs with small average distance and many clusters of nodes. However, such graph mining techniques cannot be used as such, since *SW* schemas are graphs enriched with the semantics of *RDF/S* [11] or *OWL* [18] specifications. In particular, arcs in these graphs are of different nature, namely, a) arcs representing subsumption relationships among classes, and b) arcs representing relations between classes (e.g. *has_a*) or attributes (e.g. *title*), collectively called properties. The existence of arcs of the former kind implies additional arcs of the latter one, e.g., a class inherits the properties of its ancestors. Hence, for each *SW* schema we essentially need to study two graphs that have the same set of nodes (i.e., classes or literal types), namely, the *subsumption*, and the *property* graph. It would be interesting to investigate whether the property

graph exhibits the *small world* phenonemenon or a *power-law* degree distribution, while examining the distributions involved in the subsumption graph could provide additional hints about the morphology of SW schemas. Note that, the structure of the property graph is more complex than the graphs studied in the literature, since it involves self-loops (e.g., recursive properties) and multiple arcs (i.e., two classes may be connected by more than one property).

The main contributions of the first part of this thesis are:

i) We analyze the features of the *property* graph for individual SW schemas. In particular, we show that 94.8% of the schemas with a significant number of properties approximate a *power-law* for property total (in- and out-) degree distributions with exponents in $[0.79, 2.18]$ and that 67.2% of them exhibit the *small world* phenomenon as an effect of the previous distributions in conjunction with their trend to form clusters of classes. In contrast to our work, [48] studied only the subsumption graph of schemas, while [26] studied the graph derived by aggregating all schemas (regardless of whether they are interconnected or not) but without distinguishing subsumption relationships from properties.

ii) We study the relationship among the distinct numbers of descendants per class (i.e., subsumed classes). More precisely, we show that the relationship between the $i-th$ biggest number and its corresponding rank (in descending order), i, is approximated by a *power-law* function with exponents in $[0.97, 2.44]$ for the 87.9% of the schemas defining a significant number of classes. Unlike our work, [48] examined only the frequency of the numbers of classes that have the same number of descendants and not their correlation.

iii) We provide the means to sketch an abstract morphology of SW schemas. In particular, we show that the classes with big degrees in the property graph (i.e., classes with many properties) are usually located at the higher levels of the

subsumption graph. Additionally, *SW* schemas have many clusters of connected classes. Moreover, we observed that most schemas have many cyclic paths of average size equal to 21. Finally, class subsumption hierarchies are mostly unbalanced with large branches and many leaves. To the best of our knowledge, no other work has reported similar results.

It should be stressed that, although *SW* schemas is a small subset of the *WWW*, their graph morphology exhibit common features, i.e., *power-law* for their degree distribution (i.e., properties relating classes versus hyperlinks connecting web pages), while both exhibit the *small world* phenomenon. One plausible explanation for this fact is the emergence in both graphs of the preferential attachment [5], which stems from the rough divergence in the significance (for a specific domain of discourse) of classes in *SW* schemas or the popularity of pages in the *WWW*.

The main contributions of the second part of this thesis are:

i) We propose algorithms to generate *SW* schemas given the characteristic exponents of the total-degree distribution of their *property* graph and of the class descendants distribution of their *subsumption* graph. The former guides the generation of the *property* graph, while the latter the generation of the *subsumption* graph. Combinatoric features of real *SW* schemas that correlate the two graphs (that have the same set of nodes) are then exploited to generate a *SW* schema. Specifically, the total-degree of each node is modeled as a *Discrete Real Random Variable* (*DRV*) that follows a *power-law* with the given exponent. The total-degree sequence of the *property* graph is produced by sampling values of that *DRV*. Graph features of real *SW* schemas are then exploited to produce the out- and in-degree sequences. Similarly, we produce the out-degree sequence of the *transitive closure* of the *subsumption* graph using the *power-law* exponent of the class descendants distribution. The corresponding

in-degree sequence is produced as a function of the number of classes according to features observed in real *SW* schemas.

ii) We tackle the problem of the generation of directed graphs without self-loops or multiple edges given either its out-/in-degree sequences, or the out-/in-degree sequences of its transitive closure. The former problem can be solved by the algorithm [42] in $\mathcal{O}(N^3)$ time, where N is the number of graph nodes (i.e., schema classes and literal types). However, the latter one, i.e. the generation of *transitively closed* graphs, has not been addressed in the literature yet. In this thesis we propose its reduction to the *Linear Programming* problem. In particular, we propose a reduction of the problem for trees of $\mathcal{O}(N^3)$ complexity, which yields a tree, whose *transitive closure* realizes the exact out- and in-degree sequences that were given. Moreover, we propose a reduction of the problem for DAGs of $\mathcal{O}(N^5)$ complexity, which yields a DAG whose *transitive closure* has out- and in-degree sequences that approximate the given ones. An exact solution is also possible for DAGs by a reduction to the *Integer Linear Programming* (*ILP*) problem. However, it is of no practical impact, since *ILP* is an *NP-complete* [44] problem.

The remainder of this thesis is organized as follows: Chapter 2 elaborates on the graph features of real *SW* schemas. Chapter 3 focuses on the problem of generating graphs given their degree sequences. Additionally, Chapter 4 provides methods to sample *DRV*s that follow a *power-law* distribution. Chapter 5 presents the generation of *SW* schemas by combining the graph features (see Chapter 2) of real *SW* schemas, the methods to generate graphs given their degree sequences (see Chapter 3) and the methods to sample *DRV*s that follow a *power-law* distribution (see Chapter 4). It also presents the results of the experimental evaluation of the generation of synthetic *SW* schemas. Finally, Chapter 6 summarizes the contributions of this thesis as well

as identifies issues for future research.

Chapter 2

Graph Features of SW Schemas

In this Chapter we study the graph features of real *SW* schemas. Specifically, Section 2.1 provides formal definitions of *SW* schema graphs, as well as definitions related to *power-law* distributions and the *small world* phenomenon. Section 2.2 details the corpus of schemas used in this work and the methods adopted in the experiments performed, and details the results of our analysis. Based on these measurements Section 2.3 sketches the general morphology of *SW* schemas. Finally, Section 2.4 resumes related work.

2.1 Semantic Web Schema Graphs

SW schemas are usually represented as directed labeled graphs, whose nodes are classes or literal types and arcs are properties. These graphs may have self-loops (representing recursive properties) and multiple arcs (when two classes are connected by several properties). In particular, *SW* schemas have two different kinds of arcs: subsumption arcs (*rdfs:subclassOf* or *rdfs:subpropertyOf*), and user defined ones. The former comprises *subsumption* relationships among classes or properties (which are

transitive in nature), while the latter comprises domain-specific attributes or relationships among classes, which are called *properties*. According to the *RDFS* semantics [30], a class inherits all the properties of its (direct and indirect) subsuming classes. As the interpretation of these two arc kinds is different, for each *SW* schema we need to define two graphs: a) the *property* and b) the *subsumption* graph. Both graphs have the same set of nodes (i.e., the union of classes and literal types used in the schema) but they comprise different kinds of arcs.

Definition 1 *The* property graph *of a schema is a directed graph* $\mathcal{G}_p = (\{C \cup L\}, P)$*, where* C *is a set of nodes labeled with a class name,* L *is a set of nodes labeled with a literal type,* P *is a set of arcs of the form* $\langle c_1, p, c_2 \rangle$ *where* $c_1 \in C$*,* $c_2 \in C \cup L$*, and* p *is a property name.*$\diamond$

If $\langle c1, p, c2 \rangle$ is an arc, we shall write $domain(p) = c1$ and $range(p) = c2$.

Definition 2 *The* class subsumption graph *of a schema is a directed graph* $\mathcal{G}_s = (C, P_s)$*, where* C *is a set of nodes labeled with a class name and* P_s *is a binary relation over the elements of* C*.*$\diamond$

We can denote an arc of this kind by $\langle c_1, \textit{rdfs:subclassOf}, c_2 \rangle$ meaning that c_1 is a subclass of c_2. The *property subsumption graph* can be defined in a similar way, where the nodes are the schema properties and the arcs are relationships of the form $\langle p_1, \textit{rdfs:subpropertyOf}, p_2 \rangle$ $(p_1, p_2 \in P)$.

By considering the *RDFS* semantics we can obtain the *closure* of the subsumption and property graph of a schema.

Definition 3 *Let* $\mathcal{G}_s = (C, P_s)$ *be the class subsumption graph of a schema. The* closure of the class subsumption graph*, denoted by* $\mathcal{G}_s^*$*, is the pair* (C, P_s^*) *where* P_s^* *is the reflexive and transitive closure of* P_s*.*$\diamond$

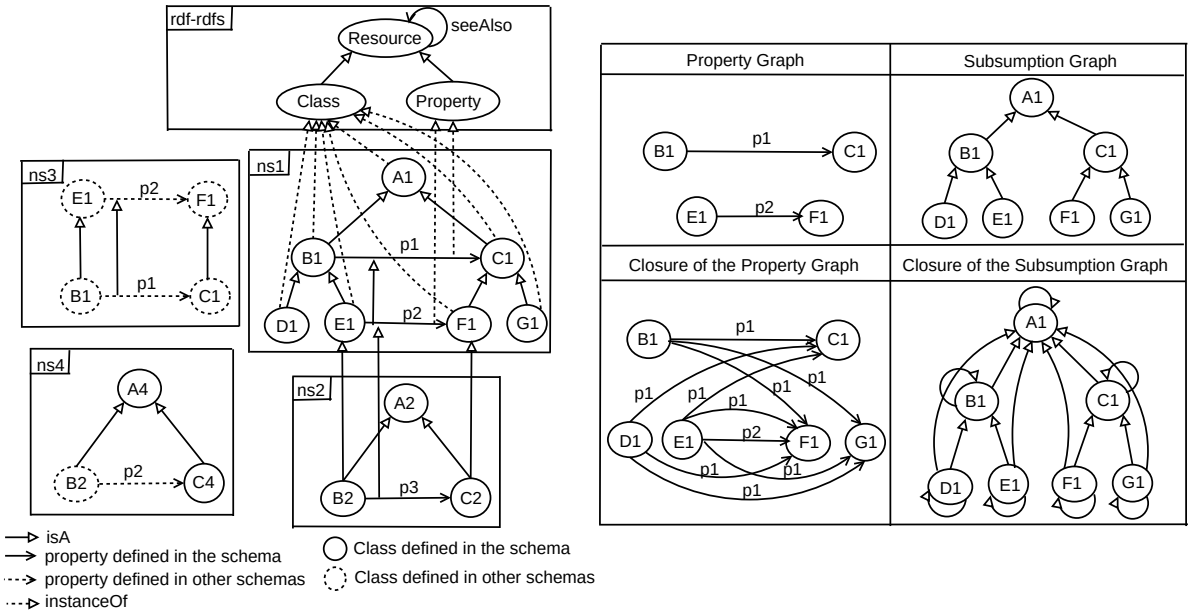

Figure 2.1: *SW* schemas reusability and extension (left) / Property and subsumption graphs of schema ns1 (right)

Definition 4 *Let* $\mathcal{G}_p = (\{C \cup L\}, P)$ *be the property graph of a schema, and let* $\mathcal{G}_s^* = (C, P_s^*)$ *be the closure of its class subsumption graph. The* closure of the property graph *of the schema, denoted by* $\mathcal{G}_p^*$*, is the pair* $(\{C \cup L\}, P^*)$ *where* P^* *comprises all arcs* $\langle c_1, p, c_2\rangle$ *such that there exists* $\langle c_1, c_3\rangle \in P_s^*$*,* $\langle c_2, c_4\rangle \in P_s^*$ *and* $\langle c_3, p, c_4\rangle \in P$*.*$\diamond$

For example, the right part of Figure 2.1 depicts the property, the subsumption graph and their closures of the schema $ns1$ illustrated in the left part of Figure 2.1.

It should be stressed that in order to study the cyclic paths, we need to reduce the *closure of the property* graph to an *undirected unlabeled* one, denoted by $\mathcal{G}_{p_{un}}^*$, which is derived by replacing the set P^* of arcs with the set $P_{un}^* = \{(c_1, c_2) \mid \langle c_1, p, c_2\rangle \in P^*$ *or* $\langle c_2, p, c_1\rangle \in P^*\}$. Additionally, studying the local connectivity of nodes implies the removal of self-loops from the $\mathcal{G}_{p_{un}}^*$, reaching thus way the $\mathcal{G}_{p_{un-}}^*$, whose set of edges is $P_{un-}^* = \{(c_1, c_2) \mid (c_1, c_2) \in P_{un}^*$ *and* $c_1 \neq c_2\}$.

Yannis Theoharis

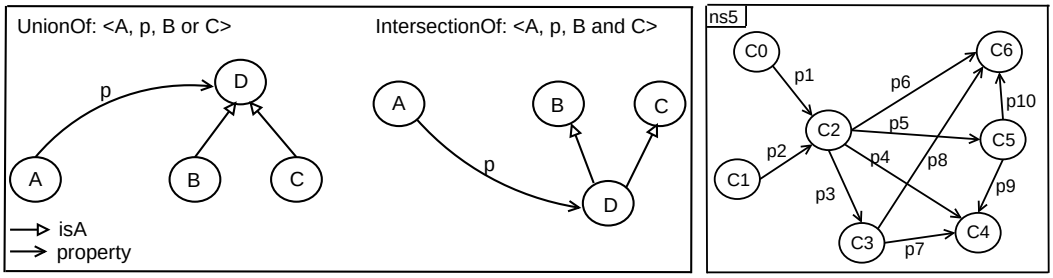

Figure 2.2: Treatment of union and intersection OWL constructs (left) / Connectivity of a *SW* schema (right)

In this work we ignore the logic-style descriptions of schemas expressed in *OWL* [18] and focus only on the core *RDFS* [11] features which are also exploited by *OWL*. In addition, we transformed *OWL* schemas that contain *unionOf* and *intersectionOf* expressions (in class or property definitions), into equivalent *RDFS* schemas (see the Figure 2.2 left). In particular, for *OWL* schemas that contain expressions of the from $A \cup B$, $A \cup B \cup C$, $A \cap B$, $A \cap B \cap C$, we have added the appropriate subsumption relationships of the form $\langle A \cup B$, *rdfs:subclassOf*, $A \cup B \cup C \rangle$, $\langle A \cap B \cap C$, *rdfs:subclassOf*, $A \cap B \rangle$. Finally, we also exploited the *Restriction* feature of *OWL* (*someValuesFrom* / *allValuesFrom*) to identify the domain/range of properties.

SW schemas are subject of social collaboration by reusing and extending existing concept and property definitions. Firstly, *reusability* of schemas is mainly achieved through the definition of properties, i.e., a schema can define as domain/range of one of its properties a class that is defined in another schema. This case is depicted in Figure 2.1 (left) where schema *ns4* defines that property p2, which is defined in *ns1*, has as domain the class B2 (defined in *ns2*) and as range the class C4 (defined in *ns4*). *Extension* of schemas is achieved through the introduction of *subsumption* relationships between classes/properties defined in different schemas. For instance, in Figure 2.1 (left) *ns2* defines classes B2 and C2 and declares that they are subsumed by E1 and F1 respectively, which have been introduced in schema *ns1*. Note that

a schema can declare that two classes are related through the subsumption relation without defining any of these two classes. For example, *ns3* declares that classes B1 and C1, which are defined in *ns1*, are subclasses of E1 and F1 respectively, which are also defined in schema *ns1*. Due to the aforementioned interconnection among schemas, the (property or subsumption) graph of a schema is obtained from the union of its classes and properties with those of the schemas that are reused or extended by it. Finally, another way to reuse schemas defined in different namespaces is through *instantiation*. As we can see from Figure 2.1 (left), all SW schemas are instances of the *rdf-rdfs* meta-schema, i.e., their classes/properties that are instances of the *rdfs:Class*/*rdf:Property* respectively (see Figure 2.1 (left)).

It is worth noticing that all schema classes may not have the same significance. Intuitively speaking, there usually exist few "central" classes (e.g., in Figure 2.2 right class $C2$ is "central") that form the conceptual backbone of the defined schema, and many "peripheral" ones that are used to further detail the analysis of the former. When a "peripheral" class (e.g., $C1$ in Figure 2.2) is added to the schema, it is more likely to be related through properties with "central" classes rather than with other peripheral ones. Moreover, the "central" classes of a schema consist *ideal* articulation points for interconnection with schemas defined in other namespaces, since many properties are located in their neighbourhood (the path between a central concept and the others is small). Both processes of SW schemas development fit in the definition of preferential attachment [5] (also called the "rich-get-richer" phenomenon), which has been proven to yield *power-law* degree distributions.

2.1.1 SW Schema Graph Features

As defined in [53] small world graphs are highly clustered like regular graphs, but have short distances between arbitrary pairs of vertices like random graphs. These

properties are quantified by the *clustering coefficient* and the *average distance* of the graph. Furthermore, small world graphs are characterized by their small *diameter*. In general, small world can emerge from the high variability of degree distributions and the preference for local connectivity (i.e., the trend to form clusters of nodes) [34].

It should be noticed that, since the property graph is sparse, i.e., its number of classes is usually bigger than its number of explicitly given properties (or at least of the same order of magnitude), we consider its closure to examine whether *RDFS* graphs exhibit a *small world* structure. Additionally, self-loops have to be removed, multiple arcs should be replaced by single ones (unlabeled), while the directedness should be ignored. This is because, in the small-world picture, we are interested only in studying static patterns of whether nodes are connected or not (and hence, we consider the $\mathcal{G}^*_{p_{un-}}$). However, since all classes are instances of the *rdfs:Resource* and there exist schema properties defined by the *rdf(s)* meta-schema with domain and range *rdfs:Resource* (e.g., the property *rdfs:seeAlso* of Figure 2.1 (left)), every *SW* schema results in a regular graph (i.e., all nodes are connected with all others). To avoid this effect, we excluded from our computations the classes and properties of the *rdf(s)* namespace (unlike [26]), which is reused by all schemas of our corpus.

We can define the neighbourhood of a class $c_i \in C$, denoted by $n(c_i)$, the set of nodes that are connected with c_i in $\mathcal{G}^*_{p_{un-}}$, i.e. $n(c_i) = \{c_j \mid (c_i, c_j) \in P^*_{un-}\}$. The clustering coefficient CC_i for a class c_i of an undirected simple graph $\mathcal{G}^*_{p_{un-}}$ is the fraction of arcs connecting neighbours of c_i (i.e., classes that are contained in $n(c_i)$) divided by the maximum number of edges that could exist among them [53], specifically, $CC_i = \frac{|\{(c_j,c_k) \mid (c_j,c_k) \in P^*_{un} \text{ and } c_j,c_k \in n(c_i)\}|}{\frac{|n(c_i)|(|n(c_i)|-1)}{2}}$. Figure 2.2 (right) shows an example of a property graph, where $n(c_2) = \{c_0, c_1, c_3, c_4, c_5, c_6\}$, i.e., $|n(c_2)| = 6$. Among these 6 nodes there can be at most $\frac{6*5}{2} = 15$ undirected edges. However, there actually exist only 4 edges, i.e., (c_3, c_4), (c_3, c_6), (c_5, c_4), (c_5, c_6). Hence, the *clustering coefficient* of c_2 is $\frac{4}{15} = 0.26$.

Definition 5 *[53] The clustering coefficient for an undirected simple graph $\mathcal{G}^*_{p_{un-}}$ is the average of the clustering coefficients of its classes, i.e., $CC_{G^*_{p_{un-}}} = \frac{1}{|C|}\sum_{i=1}^{|C|} CC_i$.*$\diamond$

A necessary and sufficient condition to characterize a graph $\mathcal{G}^*_{p_{un-}}$ as a small world, follows.

Definition 6 *[53] Let $\mathcal{G}^*_{p_{un-}}$ be a graph and $\mathcal{G}_{random}$ be a random graph with the same number of nodes and arcs as $\mathcal{G}^*_{p_{un-}}$. Also, let $L_{G^*_{p_{un-}}}$, L_{random} be the average distance of $\mathcal{G}^*_{p_{un-}}$, $\mathcal{G}_{random}$ respectively and $CC_{G^*_{p_{un-}}}$, CC_{random} be the clustering coefficient of $\mathcal{G}^*_{p_{un-}}$, $\mathcal{G}_{random}$ respectively. $\mathcal{G}^*_{p_{un-}}$ is a small world iff, $L_{G^*_{p_{un-}}} \simeq L_{random}$ and $CC_{G^*_{p_{un-}}} \gg CC_{random}$.*$\diamond$

In order to decide whether a graph is a *small world* or not, we need to set the values of L_{random} and CC_{random}. Fortunately, *Erdös* and *Rényi* have proved in [24] that a random graph with N nodes and k arcs per node on average (i.e., $N * k$ arcs totally) has average distance $L_{random} \simeq \frac{\log N}{\log k}$ and clustering coefficient $CC_{random} \simeq \frac{k}{N}$.

2.1.2 SW Schema Graph Distributions

It is well known that the distribution of the *degree sequence* determines partially or totaly some other significant features, such as the average distance and diameter of a graph, as well as the emergence of the aforementioned *small world* phenomenon. Moreover, in our setting the study of the degree distribution can reveal classes of different significance (central or peripheral). Hence, we are interested in investigating the degree distribution of each *RDFS* schema *property* graph. Since it is a directed graph, we need to distinguish between out- and in-degree distributions. The out-degree can be regarded as the distribution of property domains, while the in-degree as the distribution of their ranges. However, from a conceptual modeling viewpoint, the direction

of the properties is not important. One can replace the property $\langle B1, p1, C1\rangle$ of $ns1$ of Figure 2.1 (left), with the property $\langle C1, p3, B1\rangle$ by inverting the direction and changing the property label (see also the *OWL inverse properties*). Thus, it is reasonable to also study the total-degree distribution and try to compare its characteristics with those of the in-/out-degrees. Furthermore, since subsumption plays a significant role in *SW* schemas we will also investigate the out-/in-degree (corresponding to transitive subclasses/superclasses respectively) distributions of the closure of the subsumption graph, $\mathcal{G}_s^*$. Formally, we (mainly) consider the Discrete Random Variables (D.R.V.) for the out-/in-/total-degrees of the *property* graph, $\mathcal{G}_p$, and for the out-degrees of the closure of the *subsumption* graph, $\mathcal{G}_s^*$.

Then, for each D.R.V. we study:

i) its probability density function (*PDF*), i.e., $P(X = x)$

ii) its complementary cumulative probability density function (*CCDF*), i.e., $P(X \geq x)$

iii) the relationship among the values of its range and their rank in decreasing order (*VR*)

The consideration of *VR* beyond the two frequency-based functions is necessary, since it reflects the variability of the values (e.g., of the total degrees), while the *PDF* measures only the frequencies of the values and not their correlation [40]. Note also that the high variability of degrees, which is correlated with the total-degree *VR* (and not *PDF*), is a necessary condition for a graph to be *small world* [34]. Next, we introduce *power-law* functions.

Definition 7 *A power-law (also denoted by $P(\alpha, \beta)$) is a function $f : X \subseteq R_*^+ \rightarrow R_*^+$ of the form: $f(x) = \alpha x^{-\beta}$, where α, β are constants, with $\alpha, \beta \in R_*^+$. Equivalently,* $\log f(x) = \log \alpha - \beta \log x$. $\diamond$

This means that $f(x)$ can be drawn as a line in *log-log* scale. Obviously, the slope of this line equals $-\beta$. Power-law functions are *scale-free*, in the sense that if x is rescaled by multiplying it by a constant, then $f(x)$ would still be proportional to $x^{-\beta}$.

Previous definition excludes the value $\beta = 0.0$, since in this case $f(x)$ could be equal to α for all values of x, i.e. f could be constant (uniform distribution, if f is *PDF* of a D.R.V.). Loosely speaking, uniform distributions can be regarded as a trivial case of *power-law* distributions. Intuitively, the value of β is a measure of the skewness of the distribution, i.e. $\beta = 0$ implies no skewness, while increasing values of it implies increasing skewness. However, it is worth noticing that a distribution can be skewed without being a *power-law* and this is why the skewness of a distribution can not be the only explanation for the emergence of a *power-law*.

2.2 Experimental Results

In this section, we detail our sampling, methodology and the results of our experimental analysis[1]. For each of the three functions of the considered *DRV*s we mainly focus on two aspects: *a*) the fraction of schemas that approximates a *power-law*; and *b*) the evidence or not for some correlation between the characteristic exponent of a *power-law* function and the corresponding number of schema classes/properties.

2.2.1 SW Schemas Corpus

We collected 250 schemas from RDFSuite[2], SchemaWeb[3] and Swoogle[4] schema registries. We should stress that this corpus contains the biggest (in terms of number of classes and properties) schemas published on the *WWW* until May 2006, including

[1] To perform this analysis we extended VRP[4].
[2] athena.ics.forth.gr:9090/RDF/VRP/Examples/
[3] www.schemaweb.info
[4] swoogle.umbc.edu/

the largest schemas of the Swoogle2005 ontology set (with 4000 ontologies). The size of each schema (i.e., the number of classes and properties) is crucial for our analysis since for small schemas the 'noise' would be significant and thus would hinder the deduction of credible conclusions. Hence, we categorized (see Table 2.1) the schemas of our corpus according to their size. One group consists of 83 schemas with more than 100 classes. Note that there are four sets (i.e., $9-16$, $18-26$, $29-36$, $54-55$) of strongly interconnected schemas (i.e., each schema of a set reuses/extends all the other set schemas). A second group (subset of the first one) comprises 58 schemas that have more than 100 properties. The motivation behind this decision is that searching for any kind of distribution in schemas with less than 100 classes (properties respectively) is meaningless as explained earlier[5].

2.2.2 Power-law Investigation Method

In this subsection, we aim at investigating whether the three functions of the considered *DRV*s (see Section 2.1.2) approximate a *power-law* and if they do, to which extent. For this purpose, we rely on a commonly used method (based on the least square errors method), called Linear Regression [45], to fit a line in a set of 2-dimensional points and, thus, to investigate whether the *log-log* plot of a function approximates a line. The accuracy of the approximation is indicated by the correlation coefficient, the absolute value of which (hereafter called *ACC*) always lies in $[0, 1]$. Since, there is no formal definition of a threshold of *ACC* value over which the approximation can be characterized as good, we used 6 scales, namely $0.7, 0.75, 0.8, 0.85, 0.9, 0.95$ and looked at the plots to observe the credibility of the method[6]. By adopting this

[5]Detailed figures depicting the distributions of all corpus schemas are available at athena.ics.forth.gr:9090/RDF/VRP/SWSchemas/

[6]www.tufts.edu/∼gdallal/corr.htm

id	Name	# of C.	# of P.	id	Name	# of C.	# of P.
1	tap	6031	379	53	onto	115	190
2	dcd100	5112	355	54-55	geocoordsyst	113	102
3	not-galen	3135	484	56	moviedatabase	108	394
4	opencycprotege	2966	1317	57	datatypes	101	334
5	cyc	2731	1193	58	cononto_Fuzzy	101	117
6	bsr	2724	1770	59	unspsc84-title	16518	20
7	milo	2480	458	60	unspsc	9810	16
8	sweet_data	2036	205	61	go	7006	25
9-16	sweet_phenomena	1923	190	62	acm-css	1483	16
17	clib-core-office	1352	1786	63	mygrid-reasoned	1012	99
18-26	mds	856	914	64	gams	795	16
27	weap_of_mass_destr	828	263	65	facc	620	16
28	ontology	823	296	66	mygrid-services-lite	592	30
29-36	science	790	301	67	java_ontology	503	28
37	sumo	643	254	68	mygriddomainontology	463	30
38	framenet_1.1_inferred	537	753	69	sweet_hum_act	295	24
39	adw	445	161	70	sweet_mat_thing	295	24
40	p3p-rdf-schema	422	379	71	orel	288	56
41	akt-refont-20020509	362	243	72	math_inter	223	16
42	kimo	341	126	73	kmi_basic	216	17
43	akt_portal_ont	228	208	74	wine_ontology	166	33
44	mgedontology	226	114	75	food_ontology	166	33
45	spacenamespace	224	143	76	substance	138	16
46	resume	217	119	77	gold	129	24
47	phontology	193	148	78	sweto_v2_3	127	85
48	spase_051214	176	224	79	earthrealm	125	16
49	cerif	164	226	80	property	125	16
50	geo_Inf_Met	158	335	81	openmath	107	16
51	copyright_Ontology	155	117	82	pizza_20041007	101	46
52	georelations	148	144	83	sweet_biosphere	100	16

Table 2.1: 83 schemas with $\geq$ 100 classes (58 with $\geq$ 100 properties and 25 with $<$ 100 properties)

approach, we observed that the approximation is: *a*) not good enough to be accepted for $ACC < 0.85$, *b*) quite good, but not to the same degree for all plots for $0.85 \leq ACC < 0.9$, and *c*) credible for $ACC \geq 0.9$. For this reason, we present our results[7] for each distribution for two threshold values of ACC, namely 0.85 and 0.9. We should notice that during our experiments we observed power laws *PDF* exhibiting a characteristic 'heavy tail', i.e., small values have big frequencies. This skewness essentially consists a noise for the linear regression method, which computes a slope (which equals to $-\beta$) smaller in absolute value than the real one (see the *PDF* of schema 61 in Figure 2.3), as has been observed in [12]. In the literature there has

[7]To produce the plots as well as to perform linear regression we used Gnuplot, gnuplot.info

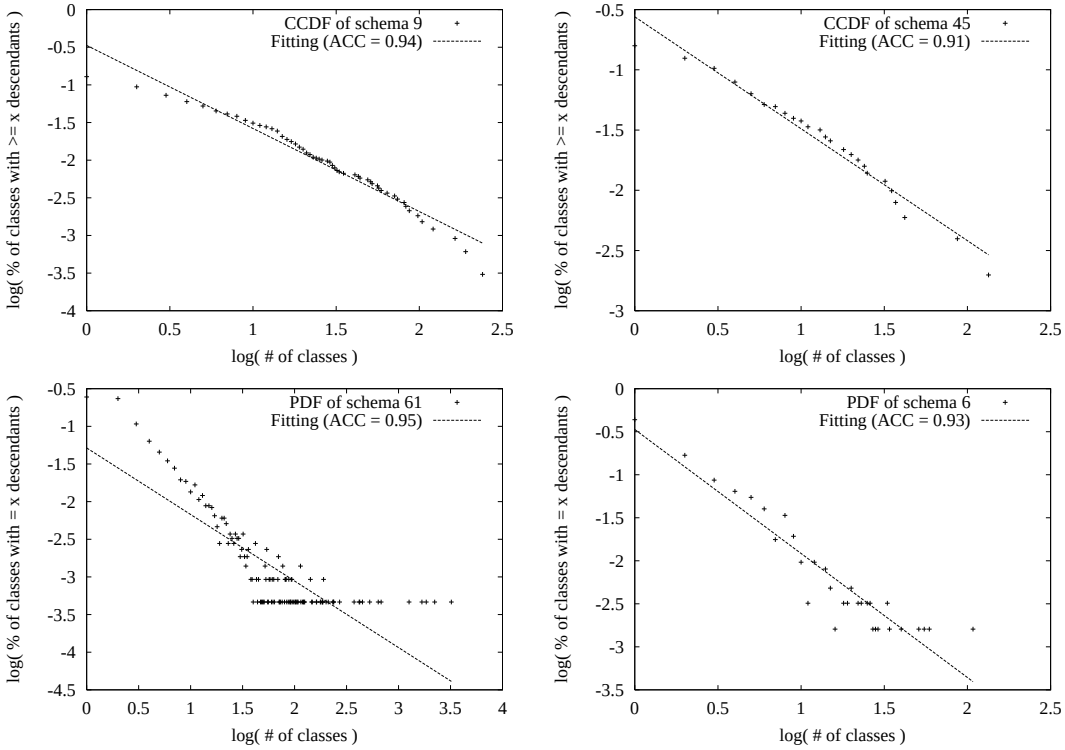

Figure 2.3: Examples of different ACC values for functions *CCDF* and *PDF*

been proposed to bin the data in logarithmic bins, i.e., bins with exponentially increasing widths as we go towards the tail. However, this method introduces a new source of noise, since binning leads to loss of information (all that we retain in a bin is its average). Hence, it is recommended to derive the characteristic exponent of the *PDF* from the corresponding exponent of the *CCDF* [12] which is more resistant to noise. If *PDF* follows a *power-law* with exponent β, i.e., $P(X = x) = \alpha x^{-\beta}$, then *CCDF* follows a *power-law* with exponent $\beta - 1$, i.e., $P(X \geq x) = \gamma x^{-(\beta-1)}$ [12]. Finally, the *VR* function is *one-to-one* and thus no noise incurs in the linear regression method.

To provide a graphical representation of what the value of *ACC* means, Figure 2.3 illustrates the *CCDF* and *PDF* functions of subsumed classes (all figures are plotted in *log-log* scale with log base 10) of schemas $9, 45$ (*CCDF*) and $61, 6$ (*PDF*). We

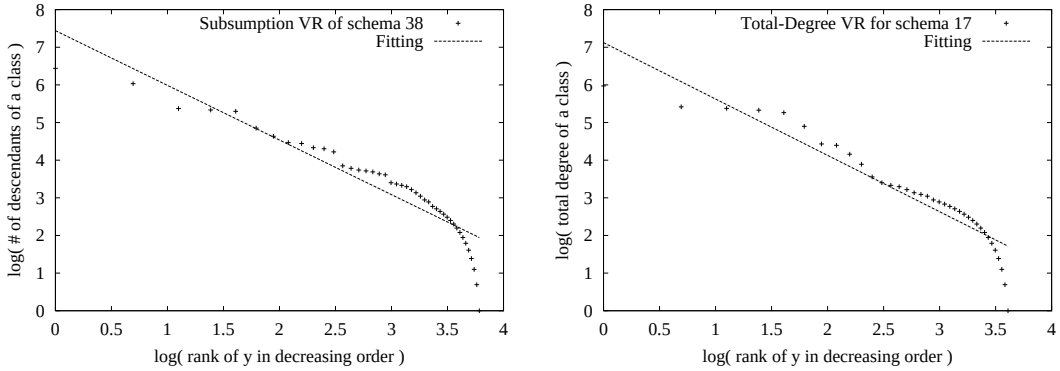

Figure 2.4: Deviations from the *power-law* for class descendants (left) / total-degree (right) *VR* functions

should mention that the *VR* function slightly diverges from a *power-law*, exhibiting a frequent plot of a curve, like the one shown in Figure 2.4 (left), instead of a strict line in *log-log* scale. This kind of divergences from a *power-law* has been also observed in the degree distributions of other graphs, like the WWW and has led to the proposal of the *DGX* [7] distribution, which is a truncated log-normal one. This observation also holds for the degree *VR* function. Another plot observed during our experiments for the *VR* function is that of wings-like plots (see Figure 2.4 (right) which corresponds to the total-degree *VR* function of schema 17). The impact of both divergences in our method is captured by observing the decreasing *ACC* values employed to demonstrate the quality of the *power-law* approximation (and thus we stick on a standard *log-log* method).

2.2.3 Features of the Property Graph

In order to obtain an accurate picture about the domain/range distributions we considered the given *property* graph $\mathcal{G}_p$ (and not its closure). As we will see later on, schema classes present strong differences in their significance. Few central classes are developed in detail (i.e., they appear as domain/range of many properties), in contrast

Domains (out-degrees)				
Distribution	≥ 100 **Properties**		≥ 300 **Properties**	
	$ACC = 0.85$	$ACC = 0.9$	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	51/58 (87.9%)	19/58 (32.7%)	27/30 (90%)	9/30 (30%)
VR	56/58 (96.5%)	34/58 (58.6%)	29/30 (96.6%)	20/30 (66.6%)
Ranges (in-degrees)				
Distribution	≥ 100 **Properties**		≥ 300 **Properties**	
	$ACC = 0.85$	$ACC = 0.9$	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	39/58 (67.2%)	14/58 (24.1%)	23/30 (76.6%)	12/30 (40%)
VR	50/58 (86.2%)	24/58 (41.3%)	27/30 (90%)	22/30 (73.3%)
Total (in- and out-degrees)				
Distribution	≥ 100 **Properties**		≥ 300 **Properties**	
	$ACC = 0.85$	$ACC = 0.9$	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	53/58 (91.3%)	39/58 (67.2%)	27/30 (90%)	25/30 (83.3%)
VR	58/58 (100%)	55/58 (94.8%)	30/30 (100%)	29/30 (96.6%)

Table 2.2: Number of schemas exhibiting a power-law function for domain / range / total degrees

to many peripheral ones. Additionally, central classes appear both as domain/range of many properties. This correlation reflects the fact that for schema developers the direction of the properties is not important.

2.2.3.1 Distribution of Property Domains (out-degrees)

Table 2.3 (left) depicts the characteristics of the distribution of property domains for schemas $1 - 58$. For each schema, its min, max, mean, standard deviation and C.O.V. (i.e., st.dev./mean) measure is shown. Table 2.2 (upper) shows the portion of schemas that define more than 100/300 properties and also approximate a *power-law* for each considered function of *property domains* and for two threshold values of *ACC*, namely 0.85 and 0.9. The main conclusion that can be drawn is that the *VR* functions of property domains of most schemas approximate a *power-law*. However, we can not infer any relationship between the number of properties of a schema that approximates a *power-law* and the exponent of it (the corresponding

		domains					ranges				
id	**# of P.**	**Min**	**Max**	**Mean**	**St.dev.**	**C.O.V.**	**Min**	**Max**	**Mean**	**St.dev.**	**C.O.V.**
1	379	0	313	18.95	67.53	3.56	0	22	1.11	1.24	1.1
2	379	0	45	18.03	13.59	0.75	0	4	1.23	0.6	0.48
3	484	0	17	2.93	2.55	0.87	0	4	1.11	0.39	0.35
4	1317	0	65	3.97	7.64	1.92	0	85	4.09	9.4	2.29
5	1193	0	63	4.07	7.79	1.91	0	77	4.1	9.18	2.23
6	1770	0	6	1.18	0.42	0.36	0	552	67.92	131.59	1.93
7	458	0	19	3.0	3.05	1.01	0	17	2.7	2.76	1.02
8	205	0	15	3.2	3.23	1.01	0	10	1.36	0.97	0.71
9-16	190	0	15	3.14	3.26	1.03	0	10	1.37	0.97	0.71
17	1786	0	225	6.41	23.81	3.71	0	197	3.98	14.18	3.55
18-26	914	0	11	2.06	1.85	0.89	0	32	1.94	3.12	1.6
27	263	0	19	3.52	3.31	0.93	0	17	3.03	3.15	1.03
28	296	0	15	2.69	2.82	1.04	0	12	2.31	2.53	1.09
29-36	301	0	7	1.76	1.28	0.73	0	30	1.79	3.38	1.87
37	254	0	19	3.66	3.38	0.92	0	17	3.16	3.25	1.02
38	753	0	6	2.66	1.69	0.63	0	4	2.4	1.01	0.42
39	161	0	8	1.59	1.3	0.82	0	31	6.6	9.7	1.45
40	379	0	313	21.82	72.87	3.33	0	42	1.1	1.67	1.5
41	243	0	10	2.51	2.11	0.84	0	38	2.27	4.28	1.87
42	126	0	12	2.69	2.48	0.92	0	34	2.74	5.35	1.95
43	208	0	21	3.14	3.84	1.22	0	8	1.92	1.66	0.86
44	114	0	6	2.15	1.34	0.62	0	4	2.4	1.01	0.42
45	143	0	16	5.15	5.3	1.02	0	19	5.45	6.0	1.1
46	119	0	12	2.37	2.44	1.02	0	16	1.84	2.64	1.43
47	148	0	29	8.78	8.79	1.0	0	19	3.4	3.95	1.16
48	224	0	124	4.55	16.8	3.68	0	133	9.61	30.14	3.13
49	226	0	12	3.81	3.28	0.86	0	4	1.4	0.79	0.57
50	335	0	21	4.45	3.74	0.84	0	10	1.63	1.33	0.81
51	117	0	9	2.79	1.95	0.7	0	7	2.16	1.59	0.73
52	144	0	11	3.56	2.78	0.78	0	8	2.0	1.71	0.85
53	190	0	38	4.54	7.84	1.72	0	30	3.48	6.17	1.77
54-55	102	0	11	3.76	2.83	0.75	0	4	2.09	1.08	0.51
56	394	0	70	6.45	12.8	1.98	0	286	12.28	52.67	4.28
57	334	0	30	4.54	5.27	1.15	0	20	3.17	4.75	1.28
58	117	0	6	1.77	1.14	0.64	0	4	1.44	0.83	0.57

Table 2.3: Distribution of property domain/range

figures are not illustrated in this paper). For *PDF* the range of exponents lies between $[1.82, 2.99]$, except for schema 2 with exponent 1.5, while for *VR* it lies between $[0.83, 2.22]$.

2.2.3.2 Distribution of Property Ranges (in-degrees)

Table 2.3 (right) depicts the basic characteristics of the distribution of property ranges for schemas $1 - 58$. Table 2.2 (middle) shows the portion of the schemas that define more than 100/300 properties and also approximate a *power-law* for each

Distribution	PDF	VR
ranges	12/39	21/39
domains	15/39	26/39
total	33/39	39/39

Table 2.4: Number of small world schemas that approximate ($ACC \geq 0.9$) a power-law for property domain/range

function of *property ranges*. We conclude that as the number of properties increases, the portion of schemas that approximate a *power-law* also increases. The range of *PDF* exponents for the majority of the schemas is $[2.12, 2.71]$. However, we can not figure out any relationship between the exponent and the number of properties. Finally, for *VR* the corresponding range of its exponents is $[0.91, 2.23]$. One noteworthy observation is that all schemas that approximate a *power-law* for *property ranges VR* function, also approximate a *power-law* for *property domains VR* function.

2.2.3.3 Distribution of Total Degrees

We should stress that there is a strong correlation between the number of properties that have as domain a class and the number of properties that have as range the same class. This correlation was revealed by the analysis of the total-degree (in- and out-degree) functions, the results of which are shown in Table 2.2 (bottom). Note that the percentages for total degrees are greater than those corresponding to both in- (ranges) and out-degrees (domains). So if we consider the properties as undirected, then their distribution better approximates a *power-law*. The range of the exponents in this case lie in $[1.65, 3.05]$ for *PDF* and in $[0.79, 2.18]$ for *VR*.

2.2.3.4 Small World Schema Graphs

As has been mentioned in Section 2.1.1, to examine if a *SW* schema graph exhibits the *small world*, we consider the $\mathcal{G}^*_{p_{un-}}$. Additionally, to decide whether such a graph

is a small world or not, we need to further detail the necessary and sufficient condition presented in Section 2.1.1, i.e., $L_{G^*_{p_{un-}}} \simeq L_{random}$ and $CC_{G^*_{p_{un-}}} \gg CC_{random}$. For the former, we consider that it is true if $L_{G^*_{p_{un-}}} < 2 * L_{random}$. This decision is motivated by the fact that the average distance of small world graphs should not exceed by an order of magnitude the average distance of a random graph with the same number of classes and properties. Moreover, since we observed that SW schemas (treated as undirected simple graphs) have small average distances, we further restrict the upper bound into $2 * L_{random}$. For the latter (i.e., $CC_{G^*_{p_{un-}}}$), we consider the initial assumption of the paper [53] that introduced this condition using as a threshold a multiplicative of CC_{random}. For example, in [53] the smallest threshold considered in the examples presented there was 5. In our experiments, we increase it up to 8 (i.e., $CC_{G^*_{p_{un-}}} > 8 * CC_{random}$), since we observe a transition point there.

39 (i.e., the schemas with id $1-5$, $7-26$, $28-36$, 40, 46, 48, 51, 53) out of the 58 schemas that contain more than 100 classes and more that 100 properties, are small world graphs. All of them, approximate a *power-law* for total-degree VR function and 33 of them also approximate a *power-law* for the corresponding PDF ($ACC \geq 0.9$). The approximation of a *power-law* from the total-degree VR function of a schema implies the high variability of total-degrees which is necessary for a graph to be *small world*. The preference for local connectivity (i.e., the second necessary condition for a graph to be a *small world*), although common, is exhibited by less schemas than those exhibiting a *power-law* total degree VR function. Furthermore, as it has been already mentioned in Section 2.1.2, the PDF does not provide any information about the relationship among values (i.e., total-degrees in this case). The difference between total-degree VR and PDF is even more clearly reflected for property *domains/ranges* (Table 2.4). Also, it is worth mentioning that even the 19 schemas that do not exhibit the small world phenomenon have small average distance (i.e. $L_{G^*_{p_{un-}}} \simeq L_{random}$ holds), but not a sufficiently big clustering coefficient (i.e., $CC_{G^*_{p_{un-}}} \gg CC_{random}$

Distr.	Min	Max	Mean	St.dev.	COV
self-loops	0.0	0.382	0.126	0.124	0.984
multiple arcs	0.0	0.626	0.177	0.194	1.09

Table 2.5: Distribution of self-loops/multiple arcs

Distr.	Min	Max	Mean	St.dev.	COV
# of cycles	0	65204	2315.51	9057.05	3.91
size of cycles	1	96	21.02	22.71	1.08

Table 2.6: Distribution of cyclic paths

does not hold). Specifically, $CC_{G^*_{p_{un-}}}$ of the schemas $1-58$ lies in $[0.1, 0.82]$ (0.1 corresponds to schema 28 while 0.82 to schema 40), while $\frac{CC_{G^*_{p_{un-}}}}{CC_{random}}$ lies in $[0.58, 179.66]$ (0.58 corresponds to schema 45 while 179.66 to schema 3).

2.2.3.5 Cyclic Paths

Another worth mentioning finding of our experiments concerns the percentage of multiple arcs and self-loops in the property graph $\mathcal{G}_p$. As we can see in Table 2.5, the effect of self-loops is significant, i.e., 12.6% in the average case and its maximum reaches the 38.2% for schema 37. The average percentage of multiple arcs is slightly bigger, i.e., 17.7%, but its maximum roughly diverges from that of self-loops, reaching the 62.6% for schema 56.

Finally, we also measure the number and the length of schema cycles (see Table 2.6). With the term 'cycles' we mean a path (of length ≥ 1) from a class to itself in the *undirected closure of the property* graph $\mathcal{G}^*_{p_{un}}$. We should notice the big difference between the *C.O.V.* measure of the 2 distributions. The former is much more skewed than the latter, i.e., most schemas have few or no cycles, while few schemas have many cycles (up to 65,204). On the other hand, the corresponding deviation in the cycle length distribution is relatively lower (i.e., 21.02 in the average case).

2.2.4 Features of the Subsumption Graph

Since subsumption is a transitive relationship, we consider the closure of the class subsumption graph, $\mathcal{G}_s^*$. We mainly focus on the out-degree (i.e., class descendants) functions. However, to better grasp the morphology of the class hierarchies in the presence of multiple subsumers (i.e., tree vs DAG-shaped hierarchies) we also study the in-degree (i.e., class ancestors) functions. Furthermore, to discover dominating structural patterns of these hierarchies, we analyzed the *DRV* of class descendants that are leaves. In addition to the depth-first view of the hierarchies provided by these three *DRV*s, we finally employ a fourth one for analyzing the number of classes located at each hierarchy level and thus, reveal a breadth-first view of the subsumption hierarchies.

From our experiments we observed that class hierarchies are usually unbalanced, i.e., some branches are very deep, while others are shallow. Thus, few classes located highly in the closure of the subsumption graph have big out-degree (i.e., many descendants), in contrary, most of those located at lower levels have small out-degree. In particular, large branches expose few leaves at the medium levels, while most of them are located at the maximum branch depth (see Figure 2.5). In conjunction with the list-like structure [52] of subsumption relationships, we can sketch the general morphology of class hierarchies as depicted in Figure 2.5.

2.2.4.1 Distribution of Class Descendants (out-degrees)

Table 2.7 (left) presents the basic characteristics of the class descendants distribution for schemas with id $1 - 58$. Table 2.8 shows the portion of schemas that approximate a *power-law* for the three functions. Our results are presented for two groups of schemas: one with ≥ 100, and another with ≥ 200 classes and for two

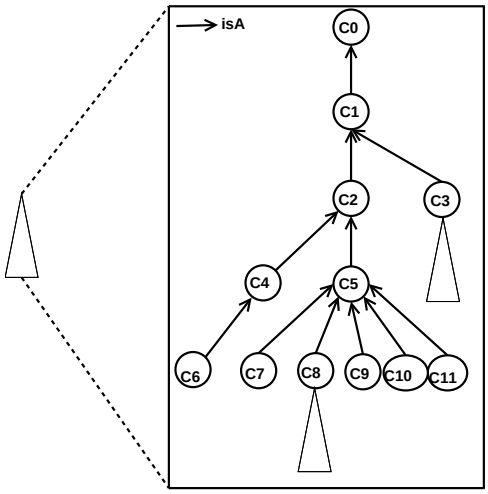

Figure 2.5: Structural pattern of class hierarchies

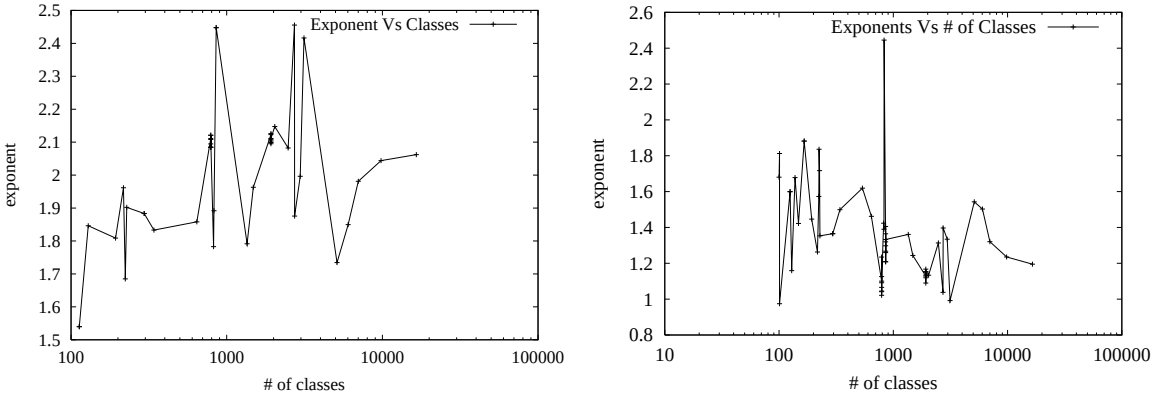

Figure 2.6: Exponents vs number of classes for *subclass FV* (left) / *VR* (right)

threshold values of ACC, namely 0.85 and 0.9. We can easily observe that the emergence of a *power-law* is almost total for VR function, which better approximates a *power-law* than the other two functions for all threshold values of ACC (confirming that it is not sensible to noise, see Section 2.2.2). Also, by comparing the two columns ('$\geq$ 100 classes' and '$\geq$ 200 classes') of Table 2.8, we can observe that, as the number of classes increases, the portion of schemas that approximate a *power-law* for some or all of the 3 functions, also increases. Moreover, as the number of classes increases, the results obtained for $ACC = 0.85$ converge to those obtained for $ACC = 0.9$ (column

			descendants				ancestors			
id	**# of Classes**	**Depth**	**Max**	**Mean**	**St.dev.**	**C.O.V.**	**Max**	**Mean**	**St.dev.**	**C.O.V.**
1	6031	11	5455	28.25	232.53	8.22	17	6.94	2.53	0.36
2	5112	4	4227	101.37	425.48	4.19	9	4.7	0.82	0.17
3	3135	6	71	4.1	6.85	1.67	8	1.6	1.00	0.6
4	2966	14	2214	17.62	105.46	5.98	64	15.22	10.98	0.72
5	2731	3	1829	17.71	95.8	5.4	61	13.94	11.35	0.81
6	2724	5	108	4.14	7.44	1.79	5	1.94	0.94	0.48
7	2480	15	626	8.24	31.59	3.83	18	3.79	2.72	0.71
8	2036	9	240	9.72	21.48	2.2	12	3.29	1.91	0.58
9-16	1923	9	241	9.94	22.07	2.21	12	3.36	1.92	0.57
17	1352	15	737	19.09	60.87	3.18	28	11.49	5.64	0.49
18-26	856	4	44	3.82	4.96	1.29	15	2.18	2.28	1.04
27	828	15	626	14.95	50.24	3.35	18	6.89	2.84	0.41
28	823	12	474	22.32	58.63	2.62	19	7.38	2.96	0.40
29-36	790	9	86	7.62	13.06	1.71	10	3.80	2.01	0.52
37	643	13	626	15.96	54.92	3.44	18	7.29	2.69	0.37
38	537	5	74	4.49	10.92	2.43	12	1.9	1.97	1.03
39	445	5	319	18.48	53.42	2.89	7	3.42	1.43	0.41
40	422	4	318	49	98.6	2.01	7	4.42	1.72	0.39
41	362	9	151	5.92	15.19	2.56	22	6.22	5.77	0.92
42	341	8	262	12.15	33.05	2.72	11	4.78	1.82	0.38
43	228	10	167	10.07	22.71	2.25	15	5.48	3.65	0.66
44	226	7	212	19.42	42.6	2.19	11	5.3	2.57	0.48
45	224	7	161	22.08	43.13	1.95	9	3.43	1.56	0.45
46	217	3	47	6.87	9.21	1.33	5	1.9	1.11	0.58
47	193	12	188	15.49	33.18	2.14	12	7.29	2.69	0.36
48	176	4	54	5.59	8.64	1.54	13	4.08	2.98	0.73
49	164	3	29	5.78	6.27	1.08	3	1.67	0.7	0.41
50	158	3	22	3.4	4.4	1.29	3	1.43	0.7	0.49
51	155	6	28	4.17	5.94	1.42	11	2.68	3.08	1.14
52	148	4	49	11.93	14.84	1.24	4	1.89	0.8	0.42
53	115	4	23	4.39	5.97	1.35	7	2.05	1.84	0.89
54-55	113	3	49	20.5	19.9	0.97	3	1.67	0.57	0.34
56	108	3	29	15.66	10.15	0.64	2	1.89	0.3	0.16
57	101	5	16	4.07	3.65	0.89	5	2.14	0.96	0.45
58	101	5	11	3.99	2.96	0.74	6	2.17	1.37	0.63

Table 2.7: Distribution of descendants/ancestors

'$\geq$ 200 classes'), i.e., the functions better approximate a *power-law*.

In order to investigate whether there is a correlation between the number of classes of a schema that exhibits a *power-law* and its characteristic exponent (i.e., the value of β), we plotted Figure 2.6 (left) depicting the *subclass FV* function ($ACC \geq 0.9$): x axis represents the number of schema classes and is plotted in log scale, while y axis represents the corresponding *power-law* exponents. We can easily observe a trend of increasing exponents with respect to the number of classes. The range of exponents lies in $[1.54, 2.47]$. Figure 2.6 (right) illustrates the *subclass VR* function whose

Distribution	≥ 100 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	75/83 (90.3%)	50/83 (60.2%)
VR	82/83 (98.7%)	73/83 (87.9%)
Distribution	≥ 200 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	60/61 (98.3%)	48/61 (78.6%)
VR	61/61 (100%)	59/61 (96.7%)

Table 2.8: Number of schemas exhibiting a power-law distribution for class descendants

Distribution	≥ 100 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	23/83 (27.7%)	17/83 (20.4%)
Distribution	≥ 200 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	22/61 (36%)	16/61 (26.2%)

Table 2.9: Number of schemas exhibiting a power-law distribution for class ancestors

exponents lie in the range $[0.97, 2.44]$, while their mean value is 1.4 approximately. Although this figure reveals a rough fluctuation of exponents, we can also infer a trend of decreasing exponents with respect to the number of schema classes.

2.2.4.2 Distribution of Class Ancestors (in-degrees)

Table 2.7 (right) illustrates the basic characteristics of the class ancestors distribution for schemas with id $1 - 58$. Due to the multiple subsumption relationships, the maximum number of superclasses of a class surpasses the maximum depth of many schemas (see Table 2.7 right). Table 2.9 details the portion of schemas that approximate a *power-law* for *PDF* and *CCDF* functions. After computing additional features of the subsumption graph, such as the number of direct ancestors (explicitly given in $\mathcal{G}_s$) per class, we observed that the schemas approximating a *power-law* for class ancestors employ extensively multiple subsumption relationships. For instance,

Distr.	Min	Max	Mean	St.dev.	COV
% of leaf classes	49.88%	96.37%	74.41%	0.1	0.14

Table 2.10: Distribution of percentages of leaf classes

Distribution	≥ 100 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	75/83 (90.3%)	48/83 (68.5%)
VR	78/83 (93.9%)	73/83 (87.9%)
Distribution	≥ 200 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	59/61 (96.7%)	44/61 (72.1%)
VR	60/61 (98.3%)	59/61 (96.7%)

Table 2.11: Number of schemas exhibiting a power-law function for leaf descendants

schemas $18 - 26$, which approximate a *power-law* ($ACC \geq 0.9$), the 18.9% of their classes have more than one proper subsumer (up to 8). Another characteristic example is schema 4, in which the 79.1% of the classes have more than one parent (up to 9). However, the majority of the schemas do not approximate a *power-law* for class ancestor functions. This fact reveals that the effect of multiple subsuming classes is not so important. The range of the corresponding *PDF* exponents lies in $[1.98, 3.73]$.

It should be stressed that the *VR* of class ancestors does not approximate a *power-law* by definition. This is due to the fact that all (or almost all) hierarchy levels contain leaf classes. Hence, almost every integer value in $[1, max]$ (i.e., up to the maximum number of class ancestors), represents a distinct number of ancestors per class. Hence, the plot of *VR* consists a line in *linear-linear* (and not in *log-log* as required for *power-law* approximation) scale.

2.2.4.3 Distribution of Leaf Descendants

We observed that on average 75% of the classes of each schema are leaves (see Table 2.10). Given that in most schemas subsumption hierarchies are deep (see

Distribution	≥ 100 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	74/83 (89.1%)	38/83 (45.7%)
VR	73/83 (87.9%)	35/83 (42.1%)
Distribution	≥ 200 **Classes**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	55/61 (90%)	35/61 (57.3%)
VR	58/61 (95%)	33/61 (54%)

Table 2.12: Number of schemas exhibiting a power-law distribution for class levels

Table 2.7), if the fanout (i.e., the number of direct descendants) was the same for all classes, the percentage of leaves would be close to 50% (i.e., as in complete trees). However, from our experiments we observed that the fanout is small for intermediate level classes and big for the classes that are close to the leaves (see Figure 2.5).

Consequently, the *DRV* of leaf descendants has as range a set of values slightly smaller than those of the corresponding *DRV* for class descendants, while bigger values exhibit slightly bigger frequencies. As a result, the percentages of schemas approximating a *power-law* for the considered functions (see Table 2.11) coincide with those of class descendants (see Table 2.8) with comparable exponents, i.e., $[1.38, 2.56]$ for *PDF* and $[1.00, 2.45]$ for *VR*.

2.2.4.4 Distribution of Class Levels

The *PDF* distribution is almost *uniform* (exponents lie in $[-0.09, 0.12]$). Specifically, in most cases *PDF* is a one-to-one function (or has only one pair with the same range values), i.e., a discrete number of classes is placed at each level. Hence, the percentages for *PDF* presented in Table 2.12 essentially demonstrate the approximation of the *uniform* and not of the *power-law* distribution. On the other hand, the *VR* of 42.1% of schemas approximate a *power-law* with exponents in $[1.04, 2.38]$. Moreover, the corresponding exponent of the *VR* distribution increases as long as the number

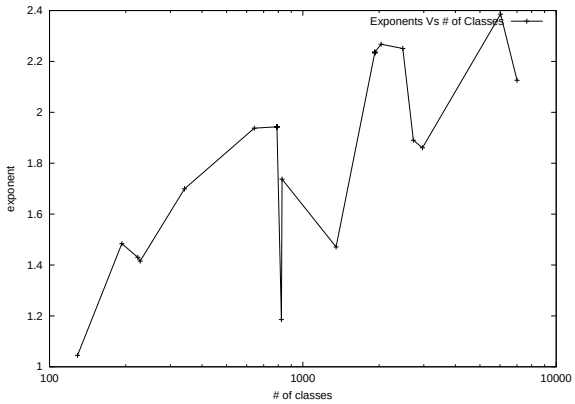

Figure 2.7: Exponents vs number of classes for *VR* class level distribution

Distribution	≥ 100 **Properties**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	15/58 (25.8%)	5/58 (8.6%)
VR	17/58 (29.3%)	8/58 (13.7%)
Distribution	≥ 300 **Properties**	
	$ACC = 0.85$	$ACC = 0.9$
PDF & CCDF	8/30 (26.6%)	5/30 (16.6%)
VR	8/30 (26.6%)	7/30 (23.3%)

Table 2.13: Number of schemas exhibiting a power-law distribution for property descendants

of classes increases (see Figure 2.7). The plot of Figure 2.7 is approximated by the line $y = 0.00017 * x + 1.36$.

Intuitively, the aforementioned results reflect the fact that usually deeper levels contain a bigger number of classes. The biggest number of classes is observed between the middle and the leaf levels. This is due to the fact that only few branches reach the maximum depth of the hierarchies, while most of them end few levels higher.

2.2.5 Subproperty Hierarchies

Although property subsumption does not play a significant role as class subsumption, we also studied the *closure of the subsumption* graph of properties. Table 2.13

depicts the percentage of schemas exhibiting a *power-law* for property descendants functions. The observed low percentages reflect the fact that subsumption relationship for properties is not widely employed by existing *SW* schemas. The range of exponents lies in $[1.63, 2.63]$, except for schema 56 with exponent 0.09 for *PDF* and in $[0.96, 1.55]$ for *VR*.

2.2.6 Combinatoric Features

An interesting finding of our experiments is that most properties have as domain, classes which are located highly in the class hierarchy, i.e., somewhere between the root and the middle level. This was revealed by computing for each schema the correlation coefficient (in most cases in $[-1, -0.5]$) between the depth in the *subsumption* graph, $\mathcal{G}_s$, of each class with its corresponding out-degree in the *property* graph, $\mathcal{G}_p$. It seems that the specification of a class in *SW* schemas is used more for classification purposes rather than for refining classes with additional properties. The same trend was observed for range classes, although the dominance of classes located higher over those located lower in the subsumption graph is not so important as for property domains.

2.3 Towards a Morphology of SW Schemas

Figure 2.8 illustrates, the abstract morphology of the graphs employed by the SW schemas of our corpus. The upper left part of the figure depicts the distribution of classes to the various subsumption hierarchy levels. The level of a class c equals 0 if c is the root, otherwise it equals $p + 1$ where p is the level of its parent class (if c has more than one parents, then p is the maximum of their levels). The upper right part of the figure shows the distribution of properties according to their level, by

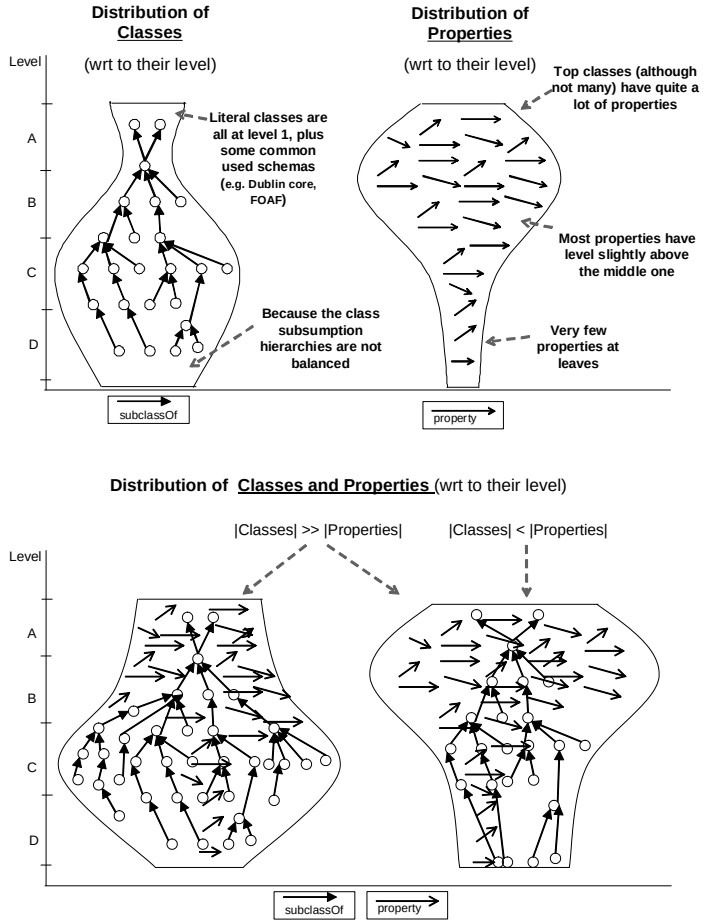

Figure 2.8: Distributions of classes and properties w.r.t. their level in the subsumption hierarchy

considering as level of a property the mean of the levels of the classes that it connects (i.e., $level(p) = (level(from(p)) + level(to(p)))/2$).

The drawings of Figure 2.8 (which resemble vases and amphoraes) were derived by first splitting the range of levels [0...maxDepth] into 4 equally sized intervals (named A, B, C, D), and then counting the classes/properties that are located at each level interval.

At the upper left part of the figure we can observe that the minimum number of classes does not occur at level A (as one would expect) but between A and B. This

is due to the fact that level A comprises all the literals, as well as, the classes of commonly reused schemas (like the Dublin Core or the FOAF schema) with shallow class hierarchies. The maximum number of classes is not placed at level D (as one would expect) but at level C (see also Section 2.2.4.4). This is due to the fact that class hierarchies are mostly unbalanced.

At the upper right part of figure we can see that the maximum number of properties are placed at level B (see also Section 2.2.6) and the minimum at level D. Notice that the number of properties at level A is bigger than the corresponding number of classes, since top level classes, although few, usually expose several properties.

The bottom part of the figure illustrates the total number of elements (classes and properties) located at each level interval. In particular, the bottom right part of the figure captures schemas that have more properties than classes. For those schemas, the maximum number of properties is bigger than the maximum number of classes for a specific level interval and as a consequence the bottom left part of the figure converges to that of properties (upper right part).

On the other hand, schemas with much more classes than properties are divided into two groups with respect to the level interval, denoted by M, at which the sum of classes and properties reaches its maximum. The former comprises schemas for which M converges to the level interval where the maximum number of classes occurs (resulting in the bottom left part of the figure). The latter comprises schemas for which M converges to the level interval where the maximum number of properties occurs (resulting in the bottom right part of the figure). M actually depends on the morphology of the subsumption hierarchy of each schema, i.e., its depth as well as the percentage of its classes placed at lower levels.

2.4 Related Work

In this Section we position the contributions of this Chapter with respect to prior research in the field. The results presented in this paper go beyond the statistical analysis presented in [41], which provides only min/avg/max values for depth and size of *RDFS* class (and property) subsumption hierarchies. Moreover, authors in [52] analyzed the expressiveness of the *OWL* fragments employed by a large corpus of *SW* schemas (we employee the same corpus of schemas but analyze only the big ones) and discovered that only few *OWL* schemas are *OWL* Full, while most of them are *OWL* Lite or DL. Furthermore, [48] classified the ontologies of the DAML ontology library[8] into three clusters, the taxonomic (i.e., ontologies with few properties and a large number of classes), the logic-style (i.e., ontologies with a high number of axioms per class) and the database-like one (i.e., ontologies of medium size containing on average 65 classes and 25 properties). They observed that the distribution of the DAML restrictions (e.g., property cardinalities) follows a *power-law* for each of the 3 clusters. They also observed that the cumulative class descendants distribution follows a *power-law* for the taxonomic cluster, while approximates a *power-law* for the other two. However, they did not study the property graph. Moreover, according to the analysis presented in [26], the cumulative total-degree distribution of the graph obtained by aggregating all ontologies of the DAML library follows a *power-law*. However, authors did not distinguish between subsumption and user defined properties, something which consists a decisive factor for the quality of the analysis method and the interpretation of the presented results. Unlike that study, in our work we treat and analyze the graphs of each *SW* schema *separately* (i.e., the graphs obtained from the union of its classes/properties and only those of other schemas that it reuses

[8] www.daml.org/ontologies

or/and extends). This is important for two main reasons: *a*) we need to know individual schema graph features in order to generate realistic, synthetic *SW* schemas, and *b*) the fact that the graph aggregating all schemas (regardless of whether they are interconnected or not) exhibits some features does not necessary imply their emergence for each individual one. This implication requires to study whether the graph is *scale-free*[9] (not addressed in [26]), i.e., whether it exhibits the same features irrespective of the employed scaling factor. On the contrary, from our experiments we found a variety of schemas approximating a *power-law* for different distributions and with different characteristic exponents. In the same direction, a recent work [32] analyzed the graph structure of two *OWL* ontologies (one of them is schema 37 while the other is small) focusing on measures like the diameter, the density of the graphs as well as on different notions of centrality (i.e., degree, betweenness, eigenvector centrality). They showed the usefulness of these measures for the identification of important (or dummy) concepts, of clusters of concepts and of the core conceptual backbone of an ontology. In this study both classes and properties have been considered as graph nodes (i.e., a bipartite graph). Although useful for evaluating ontologies according to various quality criteria, these metrics are not sufficient for generating synthetic *SW* schemas, which is the main focus of our work.

Furthermore, authors in [19, 28] collected a voluminous set of online FOAF[10] documents and analyzed the resulting RDF instance graph. They also observed a *power-law* for both in- and out-degree distribution (data level), while authors in [46] analyzed a slice of a specific folksonomy and found that the network composed of the tags from the folksonomy exhibits both of *small world* and *power-law* degree distribution. Our work focuses on *SW* schemas rather than their instances. Additionally, authors in [20] focused on aspects such as the provenance, the age and the size of

[9]It is well known that the *power-law* distribution of total-degrees is a necessary but not a sufficient condition for a graph to be *scale-free* [40].

[10]www.foaf-project.org/

SW documents (i.e., schema instances). They found that the *CCDF* of instances per class follows a *power-law*, i.e., few classes are populated by many instances while most classes by few instances or none at all. These results provide a complementary view regarding the significance of classes, i.e., not only (as in our work) determined by the number of properties for which they appear as a domain/range, but also by the corresponding number of instances. However, a study of the involved instance graphs is not presented in [20]. Finally, authors in [47] analyzed subsets of the instance graphs of the Gene Ontology (schema 61), which also exhibit *power-law* degree distributions and the *small world* phenomenon.

Chapter 3

Graph Generation Using Linear Programming

In this Chapter we elaborate on the problem of generating synthetic graphs. In particular, we are interested in the synthetic generation of the two graphs forming a SW schema, namely G_p and G_s. We assume that the in/out-degree sequences of G_p and the in/out-degree sequences of the *transitive closure* of G_s are given.

We propose a *Linear Programming* (LP) reduction of the problem of generating directed graphs (without self-loops and multiple edges) given the out-and in-degree sequence of it or of its *transitive closure*. This allows us to generate our graphs by exploiting LP algorithms, such as *Simplex* [17], that have *expected polynomial* time complexity [10], while their optimization has been extensively studied during the last 6 decades.

Previous works on generating graphs given their degree sequences can be categorized with respect to the nature of the graphs that they generate (e.g. general graph or DAG or tree) and the consideration of transitive edges or not. We position the algorithms presented in this Chapter according to these axes in Table 3.1.

The remainder of this Chapter is organized as follows: Section 3.1 introduces the LP problem in the abstract framework of optimization problems. Section 3.2 provides a model for graphs that allows to capture G_p and G_s in terms of variables and

	Non Transitive Edges		Transitive Edges	
	Previous Work	**Contribution**	**Previous Work**	**Contribution**
Undirected Graph	[27, 29] $\mathcal{O}(NlogN)$ [8] $\Omega(N^3)$	-	-	-
Directed Graph	[42] $\mathcal{O}(N^3)$	Section 3.2.1 $\mathcal{O}(N^3)$	-	Section 3.2.2.2 $\mathcal{O}(N^5)$
DAG	-	-	-	Section 3.2.2.1 $\mathcal{O}(N^5)$
Directed Tree	-	-	-	Section 3.2.2.3 $\mathcal{O}(N^3)$

Table 3.1: Graphs and Generation Algorithms

constraints of an *LP*. Specifically, Section 3.2.1 copes with the problem of generating graphs with non transitive edges. Generating graphs with transitive edges is the subject of Sections 3.2.2.1 (for DAGs), 3.2.2.2 (for graphs) and 3.2.2.3 (for Trees). Finally, Section 3.3 subsumes related work and compare our algorithms to them.

3.1 Optimization Problems and LP

Many problems of both practical and theoretical importance concern themselves with the choice of a "best" configuration or set of parameters to achieve some goal. Over the past few decades a hierarchy of such problems has emerged, together with a corresponding collection of techniques for their solution. At one end of this hierarchy is the general *nonlinear programming* (*NLP*) problem:

Find x *to*

Minimize $f(x)$

Subject to $g_i(x) \geq 0, \quad i = 1, ..., m$

$h_j(x) = 0, \quad j = 1, ..., p$

where f, g_i, h_j are general functions of the parameter $x \in R^n$. The techniques for

solving such problems are almost always iterative in nature, and their convergence is studied using the mathematics of real analysis.

When f is convex, g_i concave, and h_j linear, we have what is called a *convex programming problem*. This problem has the most convenient property that local optimality implies global optimality. We also have conditions for optimality that are sufficient, the Kuhn-Tucker conditions [44].

To take the next big step, when f and all the g_i and h_j are linear, we arrive at the *linear programming (LP)* problem. Several striking changes occur when we restrict attention to this class of problems. First, any problem in this class reduces to the selection of a solution from among a *finite* set of possible solutions. The problem is what we can call *combinatorial*. The finite set of candidate solutions is the set of vertices of the convex polytope defined by the linear constraints.

The widely used *simplex* algorithm of G.B. Dantzig [17] finds an optimal solution to a linear programming problem in a finite number of steps. This algorithm is based on the idea of improving the cost by moving from vertex to vertex of the polytope. Thirty years of refinement has led to forms of the *simplex* algorithm that are generally regarded as very efficient. It is also true, however, that there are specially devised problems on which the simplex algorithm takes a disagreeably exponential number of steps.

Although, a *polynomial* algorithm for *LP* has been invented, i.e. the *ellipsoid algorithm* [35], in practice *simplex* is still considered to be the most efficient algorithm for *LP*. The general form of an *LP* problem follows.

Definition 8 *[44] Given an $m \times n$ integer matrix A with rows $\vec{a_i^T}$, let M be the set of row indices corresponding to equality constraints, and let $\overline{M}$ be those corresponding to inequality constraints. Similarly, let $\vec{x} \in R^n$ and let N be the column indices corresponding to constrained variables and $\overline{N}$ those corresponding to unconstrained variables. Then an instance of the* general LP *is defined by*

$$
\begin{array}{ll}
min\ \vec{c^T}\vec{x} & \\
\vec{a_i^T}\vec{x} = b_i & i \in M \\
\vec{a_i^T}\vec{x} \geq b_i & i \in \overline{M} \\
x_j \geq 0 & j \in N \\
x_j \in R & j \in \overline{N}
\end{array}
$$

where $\vec{b}$ is an m-vector of integers and $\vec{c}$ an n-vector of integers.

The variables of the general form of the *LP* problem are real ones. If instead, we restrict them to take only integer values, then we reach the *Integer Linear Programming (ILP)* problem, which is *NP-complete* [44].

3.2 Modeling Graphs using LP

Let $G : (V, E)$ be the graph that we want to generate, where V is the set of its nodes and E the set of edges, i.e., the set of ordered pairs $\langle u, v\rangle$ $u, v \in V$. We will consider a variable $x_{u,v}$ for a *candidate* edge from u to v. The value $x_{u,v} = 0$ means that the edge $\langle u, v\rangle$ does not exist in G, while $x_{u,v} = 1$ means that it exists. Whenever, $0 < x_{u,v} < 1$ we will consider $x_{u,v}$ as the probability that the edge $\langle u, v\rangle$ exists in G. We should notice that if we did not allow non-integer values for each $x_{u,v}$, then our problem would no longer be an *LP* but an *ILP* instance, which is intractable as noted in Section 3.1.

Additionally, we will not consider any unconstrained variable, i.e., $\overline{N} \equiv \emptyset$, because the variables represent edges and thus are bounded in the range $[0, 1]$. Moreover, we will consider $c = \vec{0}$, (i.e., we will leave empty the *objective function*). This is due to the fact that we will exclusively express the conditions for the given sequences to be simultaneously realizable in terms of constraints of the *LP* instance. The features of the graph that we want to generate, will affect the number of the candidate edges

(i.e., the value of n in Definition 8), as well as the number (i.e., the value of m in Definition 8) and the type (i.e., equality or inequality) of the constraints of the LP instance. Specifically, the influencing factors are: a) whether G is a general graph or a DAG or a tree and b) where the set of its edges, i.e. E, should be transitively closed or not.

3.2.1 Non Transitive Edges

In this section we elaborate on the problem of generating a graph G that simultaneously realizes the given in-/out-degree sequences. In the rest of this Chapter, we will denote with $Dout/Din$ the out-/in-degree (respectively) sequence of G. Finally, we will denote with $Dout(u)/Din(u)$ the out-/in-degree (respectively) of node u in G.

We consider as candidate edges all the possible edges of a *directed graph* without *self-loops* and *multiple edges*. In such a graph every node can be connected with each other once, resulting in a set of $N(N-1)$ candidate edges.

After having computed the set of candidate edges (i.e., the variables of the LP instance) we turn our attention to the definition of constraints which guarantee that G realizes the given sequences. It is evident that the sum of the values of all candidate out-/in-going edges from/to a node of G should be equal to its out-/in-degree respectively:

$$\forall u \in V, \sum_{(u,v)\in E} x_{u,v} = Dout(u) \qquad and \qquad \forall u \in V, \sum_{(v,u)\in E} x_{v,u} = Din(u) \tag{3.1}$$

Considering additionally the constraints that bound each candidate edge between 0 and 1, we reach the reduction of our problem to the LP problem of the form *LP1*.

Definition 9 (LP1)
Let G be a graph and let E be the set of its candidate edges. The generation of G given its out- and in-degree sequences can be reduced to an LP *instance of the form:*

$$
\begin{aligned}
& min\ 0^{\vec{T}}\vec{x} \\
& \textstyle\sum_{(v,u)\in E} x_{v,u} = Dout(v),\ \forall v \in V \\
& \textstyle\sum_{(u,v)\in E} x_{u,v} = Din(v),\ \ \forall v \in V \\
& 0 \leq x_{u,v} \leq 1, \qquad\quad \forall \langle u,v\rangle \in E
\end{aligned}
$$

The number of constraints of *LP1* is $2 * N$, since for each node $u \in V$ we form 2 constraints (one corresponding to each in- and one corresponding to each out-degree). Hence, the expected time complexity of *simplex* to solve an *LP1* instance is $\mathcal{O}(N^3)$. We should stress that although we allowed the edges variables $x_{u,v}$ to take non-integer values, every solution of an *LP1* instance is integral (all the variables $x_{u,v} \in Z$ and more specifically $x_{u,v} \in \{0,1\}$). To prove this fact we will first show (see Theorem 2) that the matrix A of every *LP1* instance is *totaly unimodular (TUM)*, i.e. every square submatrix of A has determinant equal to 0 or ± 1 by exploiting Theorem 1 and then we will exploit Theorem 3 to conclude that every solution of an *LP1* instance is integral.

Theorem 1 *[44] An integer matrix A with $a_{ij} = 0, \pm 1$ is TUM if no more than two nonzero entries appear in any column, and if the rows of A can be partitioned into two sets I_1 and I_2 such that:*

- *If a column has two entries of the same sign, their rows are in different sets;*
- *If a column has two entries of different signs, their rows are in the same set.*

Theorem 2 *Every LP1 instance can be described as $\{\vec{x} : A\vec{x} = \vec{b}, 0 \leq \vec{x} \leq 1\}$ where A is a TUM $m \times n$ matrix.*

Proof. *It is evident from the type of constraints of LP1 that each element a_{ij} of A is either 0 or 1. Moreover, A can be split into two $\frac{m}{2} \times n$ matrices C, D, i.e, $A = \binom{C}{D}$, where C corresponds to the first $\frac{m}{2}$ rows of A (i.e., those corresponding to the out-degree constraints) and D corresponds to the next $\frac{m}{2}$ rows of A (i.e., those corresponding to the in-degree constraints). Each variable $x_{i,j}$ appears in one and only out-degree constraint, namely the constraint that corresponds to the $Dout(i)$. Hence, each column of C contains one and only nonzero element (i.e. equal to 1). Similarly,*

each variable $x_{i,j}$ appears in one and only in-degree constraint, namely the constraint that corresponds to the $Din(j)$. Hence, each column of D contains one and only nonzero element. Hence, every column of A (which corresponds to a specific variable $x_{i,j}$) has 2 nonzero elements (i.e., $+1$) of the same sign. Moreover, the rows of A can be partitioned into two sets I_1 and I_2 such that if a column has two entries of the same sign, their rows are in different sets. It is evident that I_1 coincides with the set of rows of C and I_2 with the set of rows of D. Since all the conditions of Theorem 1 are satisfied, we conclude that the matrix A of every LP1 instance is TUM.

Theorem 3 *[31, 51] If A is a TUM matrix, then all the vertices of the polytope $\{\vec{x} : A\vec{x} = \vec{b},\ \vec{x} \geq 0\}$ are integer for any integer vector $\vec{b}$.*

Simplex algorithm seeks for the optimum in the vertices of the polytope defined by an *LP* instance. Since, every vertex of the polytope defined by an *LP1* instance is integral (as a corollary of Theorems 2, 3), every solution of it is integral. Theorem 3 assumes that $\vec{x} \geq 0$. The additional range constraints $\vec{x} \leq 1$ of every *LP1* instance restricts the range of the coordinates of $\vec{x}$ in $\{0, 1\}$.

3.2.2 Transitive Edges

In this paragraph we elaborate on the problem of generating a *transitively open* DAG G, whose transitive closure, denoted by G^*, simultaneously realizes the given in-/out-degree sequences. Hereafter, we will denote with $Dout/Din$ the out-/in-degree (respectively) sequence of G^* (and not G, as in Section 3.2.1). We first study the problem in its generality and then under the assumption that G to is also a *tree*.

3.2.2.1 Directed Acyclic Graphs

We divide our method into two steps. The former is to generate G^* while the latter to compute the *transitive reduction* on G^* in order to obtain G. Figure 3.1 shows a *transitively open* DAG whose transitive closure simultaneously realizes the

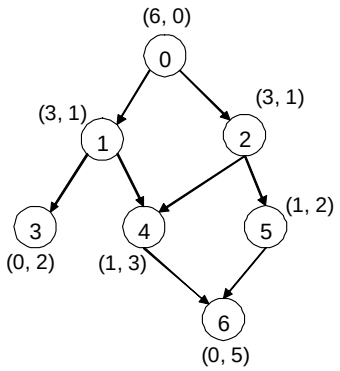

Figure 3.1: An example of a DAG

following sequences, $Dout = \langle 6, 3, 3, 0, 1, 1, 0 \rangle$, $Din = \langle 0, 1, 1, 2, 3, 2, 5 \rangle$. For each node u, the pair $(Dout(u), Din(u))$ is drawn. As one can observe in Figure 3.1, each node has more transitive descendants than any of each descendant (direct or transitive). Similarly, it has less transitive ancestors than any of each descendants (direct or transitive). The reader's intuition is validated by the Lemma 1.

Lemma 1 *An edge $\langle u, v \rangle \in V \times V$ can be an edge of a transitively closed DAG $G^* : (V, E^*)$ only if, $Dout(u) > Dout(v)$ and $Din(u) < Din(v)$.*

Proof. *Let $\langle u, v \rangle \in V \times V$ be an edge of G^* and let $desc(u)$ be the set of the transitive descendants of each node u in V, i.e. $desc(u) = \{v \in V \mid \langle u, v \rangle \in E^*\}$. Then, $\forall w \in desc(v)$ it also holds that $w \in desc(u)$. Equivalently, $desc(v) \subseteq desc(u)$. Moreover, $v \in desc(u)$, while $v \notin desc(v)$. Hence, $desc(v) \subset desc(u)$, i.e. $Dout(u) > Dout(v)$. Similarly, let $anc(u)$ be the set of the transitive ancestors of each node u in G^*, i.e. $anc(u) = \{v \in V \mid \langle v, u \rangle \in E^*\}$. Then, $\forall w \in anc(u)$ it also holds that $w \in anc(v)$. Equivalently, $anc(u) \subseteq anc(v)$. Moreover, $u \in anc(v)$, while $u \notin anc(u)$. Hence, $anc(u) \subset anc(v)$, i.e., $Din(u) < Din(v)$.*

We should notice that the strict inequalities in the formulae of the above lemma, guarantee that no cyclic path can be obtained by the considered set of candidate edges. Otherwise, all the nodes of the cyclic path would have exactly the same out- and in-degree, since G^* is transitively closed, something that contradicts to the fact that the edges of the cyclic path are candidate.

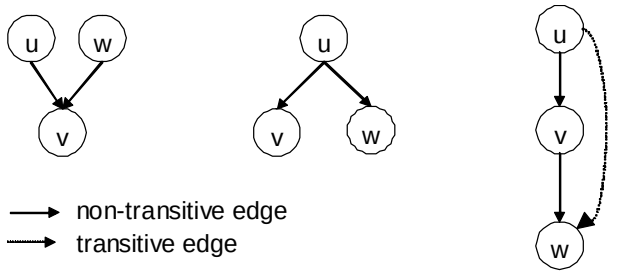

Figure 3.2: Three possible connections of three nodes

After having computed the set of candidate edges (i.e., the variables of the LP instance) we turn our attention to the definition of constraints which guarantee that G^*: a) realizes the given sequences and b) is transitively closed. Beginning with the former, the condition presented in Section 3.2.1 for the out- and in degrees of nodes holds for every graph and thus it also holds for G^*. Thus, for each node we consider two linear equality constraints that express our goal that G^* will have the given degree sequences.

However, this is not sufficient for our problem. Many DAGs may simultaneously realize the same pair of sequences. Among them, we seek for those (specifically one of those) that are also transitively closed (recall we aim to generate G^*). As a consequence, we need to model the transitivity of the edges as linear constraints for the LP problem that we build. Figure 3.2 provides us the intuition behind the type of linear constraints we seek.

Whenever two edges of the form $\langle u, v\rangle$, $\langle v, w\rangle$ exist in the generated graph, we should enforce the edge $\langle u, w\rangle$ also to exist. The fact that $\langle u, v\rangle$, $\langle v, w\rangle$ exist implies that $x_{u,v} + x_{v,w} = 2$. Note also that it always (irrespective of whether $\langle u, v\rangle$, $\langle v, w\rangle$ exist or not) holds that $x_{u,v} + x_{v,w} \leq 2$ (since $0 < x_{u,v} < 1, \forall\langle u, v\rangle \in E$). To enforce that $\langle v, w\rangle$ exists whenever the other two edges exist, we write $x_{u,v} + x_{v,w} - x_{u,w} \leq 1$.

Lemma 2 *Let $G^* : (V, E^*)$ be a transitively closed DAG. For every triple $\langle u, v, w\rangle \in V \times V \times V$, such that $\langle u, v\rangle$, $\langle v, w\rangle \in E^*$, it holds that: $x_{u,v} + x_{v,w} - x_{u,w} \leq 1$.*

Proof. *Whenever either $x_{u,v}$ or $x_{v,w}$ does not exist, the above inequality obviously holds since it is of the form $a - b \leq 1$, where $a, b \in [0, 1]$ (two such examples are shown in the left and the center part of Figure 3.2). We focus on the case that both $x_{u,v}$ and $x_{v,w}$ exist (see as an example the right part of Figure 3.2). Then, $x_{u,w}$ also exists because G^* is transitively closed. Hence, $x_{u,v} + x_{v,w} - x_{u,w} = 1 + 1 - 1 \leq 1$, which obviously holds as equality.*

We consider one such constraint for every triple $\langle u, v, w\rangle$ such that both $\langle u, v\rangle$, $\langle v, w\rangle$ are candidate edges. Adding the range constraints that bound every variable in $[0, 1]$, we have completed the reduction of the generation of G^* to the *LP* problem of the form *LP2*.

Definition 10 (LP2)
Let G^* be a transitively closed DAG and E^* be the set of its candidate edges. Then, the generation of G^* given its degree sequences, Dout, Din, can be reduced to an LP *instance of the form:*

$$\begin{aligned} \min\ & 0^{\vec{T}}\vec{x} \\ \textstyle\sum_{(u,v)\in E^*} x_{u,v} &= Dout(u), && \forall u \in V \\ \textstyle\sum_{(v,u)\in E^*} x_{v,u} &= Din(u), && \forall u \in V \\ x_{u,v} + x_{v,w} - x_{u,w} &\leq 1, && \forall\langle u, v\rangle,\ \langle v, w\rangle \in E^* \\ 0 \leq x_{u,v} &\leq 1, && \forall\langle u, v\rangle \in E^* \end{aligned}$$

Let us now see an example of the proposed reduction for the generation of a *transitively closed* graph given its sequences. Note that there is no reason to express constraints of the form $\sum_{(v,u)\in E^*} x_{v,u} = 0$ because, in this case, we know a priori that $\forall(v, u) \in E^*,\ x_{v,u} = 0$. Similarly, we do not express the constraint $\sum_{(root,u)\in E^*} x_{root,u} = Dout(root)$, since in a transitively closed graph, the root is connected to every other node of the graph, i.e., $\forall(root, u) \in E^*,\ x_{root,u} = 1$. Moreover, we replace $x_{root,u}$ with 1 in every constraint that it appears.

Example 1 *The LP2 instance that corresponds to the sequences* $Dout = \langle 6, 3, 3, 0, 1, 1, 0\rangle$, $Din = \langle 0, 1, 1, 2, 3, 2, 5\rangle$ *follows:*

$$\begin{gathered}
\min 0^{\vec{T}}\vec{x} \\
+x_{1,3} + x_{1,4} + x_{1,5} + x_{1,6} = 3 \\
+x_{2,3} + x_{2,4} + x_{2,5} + x_{2,6} = 3 \\
+x_{4,6} = 1 \\
+x_{5,6} = 1 \\
+x_{1,3} + x_{2,3} = 1 \\
+x_{1,4} + x_{2,4} = 2 \\
+x_{1,5} + x_{2,5} = 1 \\
+x_{1,6} + x_{2,6} + x_{4,6} + x_{5,6} = 4 \\
+x_{1,4} + x_{4,6} - x_{1,6} \leq 1 \\
+x_{1,5} + x_{5,6} - x_{1,6} \leq 1 \\
+x_{2,4} + x_{4,6} - x_{2,6} \leq 1 \\
+x_{2,5} + x_{5,6} - x_{2,6} \leq 1 \\
0 \leq x_{1,3} \leq 1 \\
0 \leq x_{1,4} \leq 1 \\
0 \leq x_{1,5} \leq 1 \\
0 \leq x_{1,6} \leq 1 \\
0 \leq x_{2,3} \leq 1 \\
0 \leq x_{2,4} \leq 1 \\
0 \leq x_{2,5} \leq 1 \\
0 \leq x_{2,6} \leq 1 \\
0 \leq x_{4,6} \leq 1 \\
0 \leq x_{5,6} \leq 1
\end{gathered}$$

The solution has the following 1*-coordinates* $(x_{1,4}, x_{1,5}, x_{1,6}, x_{2,3}, x_{2,4}, x_{2,6}, x_{4,6}, x_{5,6})$ *and all the other variables are zero. Note that* $x_{0,1}$, $x_{0,2}$, $x_{0,3}$, $x_{0,4}$, $x_{0,5}$, $x_{0,6}$ *have been initially precomputed to be* 1*. The transitive reduction of* G^* *is the DAG obtained from the DAG shown in Figure 3.1, if we swap nodes* 1 *and* 2*, which have the same out- and in-degree, i.e. (3,1).*

The *LP2* Instance Solution. Beginning with the case that the problem is *infeasible*, we can conclude that the constraints of the problem are not *simultaneously satisfiable*, i.e., there does not exist a *transitively closed* DAG that simultaneously

realizes the given sequences. However, if the *LP2* instance is *feasible*, there is no guarantee that the solution is integral, i.e., that all its variables are integers. We should devise a method for obtaining the set of edges of the generated graph.

The fact that the *LP2* instance is feasible provides us a crucial observation: there exists at least one *transitively closed* DAG that simultaneously realizes the given sequences. In the style of [55], we could adopt a method for selecting, according to certain criteria, a non-integer variable and to generate two subproblems by setting this variable to 0 and 1, respectively. These two subproblems are solved recursively, repeating the above process for each subproblem. However, the recursion depth, which determines the number of *LP* subproblems to be solved, can be big. The formulation of sufficient criteria (if exist) to select among the non-integer variables of the solution of the current subproblem the one (and to decide its value, i.e. 0 or 1) which faster guides to an integral solution remains an open problem. Hence, we rely on heuristics leading to "good approximations", i.e., the degree sequences of the generated graph approximate the given ones. For instance we can consider a threshold value $T \in [0, 1]$, and consider that an edge $\langle u, v\rangle$ exists iff $x_{u,v} \geq T$. In Chapter 5 we will explore issues concerning the value of T that yields the best approximations.

Complexity. To study the time complexity for solving the *LP* instance presented above, we have first to bound the number of its variables and the number of its constraints. The former is evident to be $\mathcal{O}(N^2)$ (however, note that in practice it can be much less than $N(N-1)$, depending on the given sequences), while for the latter note that the degree constraints are $2*N$ (N for out-degrees and N for in-degrees) and the transitivity constraints are $\mathcal{O}(N^3)$, since all possible triples u, v, w, such that both $\langle u, v\rangle$, $\langle v, w\rangle$ are candidate edges, are less (not all edges are candidate) than N^3. We conclude that *simplex* has expected time complexity $\mathcal{O}(N^5)$. However, the constants of the $\mathcal{O}$ definitions are quite small in practice for our framework. For instance, as has

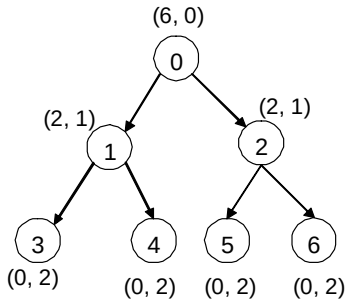

Figure 3.3: An Example of a Tree

been observed in the Section 2.2.4 approximately 75% of the classes of a SW schema are leaves (i.e., their out-degree is 0), and thus the number of variables is much less than $N(N-1)$. Using the same observation we can also decrease the number of constraints by a constant factor.

3.2.2.2 Graphs

Although in this work we do not need to generate any *transitively closed* graph (G_s^* is a DAG or a tree), we note that if we modify appropriately the set of candidate edges to allow cycles, then we can reduce the problem of the generation of transitively closed graphs to an LP2 instance. Specifficaly, we need to replace the necessary and sufficient condition for an edge $\langle x_{u,v} \rangle$ to be candidate, i.e. $Dout(u) > Dout(v)$ and $Din(u) < Din(v)$, with $Dout(u) \geq Dout(v)$ and $Din(u) \leq Din(v)$, since it is evident that all the nodes of a cyclic path of a transitively closed graph have the same out- and in-degree. However, notice that the transitive reduction of a graph that contain cycles is not unique.

3.2.2.3 Trees

In this subsection we restrict our attention to the generation of trees. In this case, it is possible to generate G immediately from the degree sequences of G^*, without

generating firstly G^*. Below, we make clear the idea, beginning with an intuitive example.

Figure 3.3 depicts a tree whose transitive closure simultaneously realizes the following sequences, $Dout = \langle 6, 2, 2, 0, 0, 0, 0 \rangle$, $Din = \langle 0, 1, 1, 2, 2, 2, 2 \rangle$. For each node u, the pair $(Dout(u), Din(u))$ is drawn. Since, every tree is also a DAG, for each edge $\langle u, v \rangle$ it holds that $Dout(u) > Dout(v)$ and $Din(u) < Din(v)$. Additionally, as the reader can observe from the tree drawn in Figure 3.3, for each edge $\langle u, v \rangle$ it holds that $Din(u) = Din(v) - 1$. More formally, we prove the following lemma.

Lemma 3 *An edge $\langle u, v \rangle \in V \times V$ can be an edge of a Tree $G : (V, E)$ only if, $Dout(u) > Dout(v)$ and $Din(u) = Din(v) - 1$.*

Proof. *Let $\langle u, v \rangle \in V \times V$ be an edge of G and let $desc(u)$ be the set of the transitive descendants of each node $u \in V$, i.e. $desc(u) = \{v \in V \mid \langle u, v \rangle \in E^*\}$, where E^* stands for the set of edges of G^*, i.e., the transitive closure of E. Then, $\forall w \in desc(v)$ it also holds that $w \in desc(u)$. Equivalently, $desc(v) \subseteq desc(u)$. Moreover, $v \in desc(u)$, while $v \notin desc(v)$. Hence, $desc(v) \subset desc(u)$, i.e. $Dout(u) > Dout(v)$. Similarly, let $anc(u)$ be the set of the transitive ancestors of each node u in G^*, i.e. $anc(u) = \{v \in V \mid \langle v, u \rangle \in E^*\}$. Since G is a tree there is a unique path, denoted by $p = \langle root, ..., v \rangle$, from the root node to node v and v has one and only parent node. Thus, if $\langle u, v \rangle$ is an edge of G, then u is the immediate predecessor of v in p. Hence, $anc(v) \setminus anc(u) = u$, i.e., $Din(u) = Din(v) - 1$.*

Lemma 3 actually splits V into partitions of nodes which can not be interconnected. Specifically, a node located at the $i - th$ level of the tree can be connected only with nodes located at the $(i+1) - th$ level, since $Din(u)$ coincides with the level at which u is located. Moreover, $\forall u \in V_i$ there should be a set $S \subseteq V_{i+1}$ such that $\sum_{v \in S} Dout(v) + |S| = Dout(u)$. For instance, consider the node v_0 of the tree drawn in Figure 3.3. It holds that $Dout(v_1) + Dout(v_2) + 2 = 2 + 2 + 2 = 6 = Dout(v_0)$. Based additionally on Lemma 3, as well as on the fact that for every tree each node (except for the root) has one and only parent we formulate the problem of generating G in terms of an LP instance as follows.

Definition 11 (LP3) *Let $V_i = \{v \in V \mid Din(v) = i\}$ and d be the maximum value of i. V_i corresponds to the set of nodes located to the $i-th$ level of the tree G and d to its depth. Then the problem of generating G given the sequences of its transitive closure is reduced to the LP instance of the form :*

$$\begin{gathered}
\min 0^{\vec{T}}\vec{x} \\
\sum_{u\in V_{i+1}} (Dout(u)+1)x_{v,u} = Dout(v), \quad \forall i \in [0, d-1],\ \forall v \in V_i \ s.t.\ Dout(v) \neq 0 \\
\sum_{u\in V_{i-1}} x_{u,v} = 1, \qquad \forall i \in [1, d],\ \forall v \in V_i \\
0 \leq x_{u,v} \leq 1, \qquad \forall \langle u, v\rangle \in E
\end{gathered}$$

Example 2 *The* LP3 *instance that corresponds to the sequences* $Dout = \langle 6, 2, 2, 0, 0, 0, 0\rangle$, $Din = \langle 0, 1, 1, 2, 2, 2, 2\rangle$ *follows:*

$$\begin{gathered}
\min\ 0^{\vec{T}}\vec{x} \\
+3x_{0,2} + 3x_{0,1} = 6 \\
+1x_{2,4} + 1x_{2,6} + 1x_{2,3} + 1x_{2,5} = 2 \\
+1x_{1,4} + 1x_{1,6} + 1x_{1,3} + 1x_{1,5} = 2 \\
+x_{0,2} = 1 \\
+x_{0,1} = 1 \\
+x_{2,4} + x_{1,4} = 1 \\
+x_{2,6} + x_{1,6} = 1 \\
+x_{2,3} + x_{1,3} = 1 \\
+x_{2,5} + x_{1,5} = 1 \\
0 \leq x_{0,2} \leq 1 \\
0 \leq x_{0,1} \leq 1 \\
0 \leq x_{2,4} \leq 1 \\
0 \leq x_{2,6} \leq 1 \\
0 \leq x_{2,3} \leq 1 \\
0 \leq x_{2,5} \leq 1 \\
0 \leq x_{1,4} \leq 1 \\
0 \leq x_{1,6} \leq 1 \\
0 \leq x_{1,3} \leq 1 \\
0 \leq x_{1,5} \leq 1
\end{gathered}$$

The solution has the following 1*-coordinates* $(x_{0,1}, x_{0,2}, x_{1,4}, x_{1,6}, x_{2,3}, x_{2,5})$ *and all the other variables are zero. The corresponding tree is obtained from the tree shown in Figure 3.3, if we swap nodes* 3 *and* 6*, which have the same out- and in-degrees, i.e.* $(0, 2)$

The *LP3* Instance Solution. Beginning with the case that the *LP3* instance is infeasible, we can conclude that there does not exist a tree whose transitive closure simultaneously realizes the given sequences. Surprisingly enough, we observed that whenever the *LP3* instance is feasible, its solution is integral, i.e., all the variables take integer values (0 or 1). This fact was not expected, since the matrix A that corresponds to the constraints of the *LP3* problem is not *totally unimodular (TUM)* in the general case (e.g., consider the *LP3* instance expressed in Example 2. Fortunately, Theorem 4 guarantees the integrality of LP3 instance solutions,

Theorem 4 *Every solution, $\vec{x^*}$ of an LP3 instance is integral. Specifically, it holds that $\vec{x^*} \in \{0,1\}^n$.*

Proof. *Every LP3 instance can be described as $\{\vec{x} : A\vec{x} = \vec{b},\ 0 \leq \vec{x} \leq 1\}$, where A is an $m \times n$ matrix that can be split into two matrices C, D, i.e ., $A = \binom{C}{D}$, where C corresponds to the first m_1 rows of A (i.e., those corresponding to the out-degree constraints) and D corresponds to the next $m-m_1$ rows of A (i.e., those corresponding to the in-degree constraints), and $\vec{b}$ can similarly be split in two vectors $\vec{b_1}$, $\vec{b_2}$. Then the LP3 instance can be described as $\{\vec{x} : C\vec{x} = \vec{b_1},\ D\vec{x} = \vec{b_2},\ 0 \leq \vec{x} \leq 1\}$. Each element of D is either 0 or 1. Additionally, each column of D has only one nonzero entry, because each variable $x_{i,j}$ appears in one and only in-degree constraint. Hence, D is* TUM *according to Theorem 1. According to Theorem 3 each solution $\vec{x^*}$ of the system $D\vec{x} = \vec{b_2},\ \vec{x} \geq 0$ is integral. Thus each solution of the system $D\vec{x} = \vec{b_2},\ 0 \leq \vec{x} \leq 1$ is contained in $\{0,1\}^n$. Additionally, each solution of an LP3 instance should also be solution of the system $D\vec{x} = \vec{b_2},\ 0 \leq \vec{x} \leq 1$ and thus each solution of an LP3 instance is contained in $\{0,1\}^n$.*

It is worth noticing that integral solution does not necessary implies uniqueness of solution (even if it is defined as uniqueness up to isomorphism). For instance, consider the two non-isomorphic trees, depicted in Figure 3.4, that simultaneously realize the same pair of sequences. The solution of the *LP3* instance produced for that case will be the left (in Figure 3.4) tree.

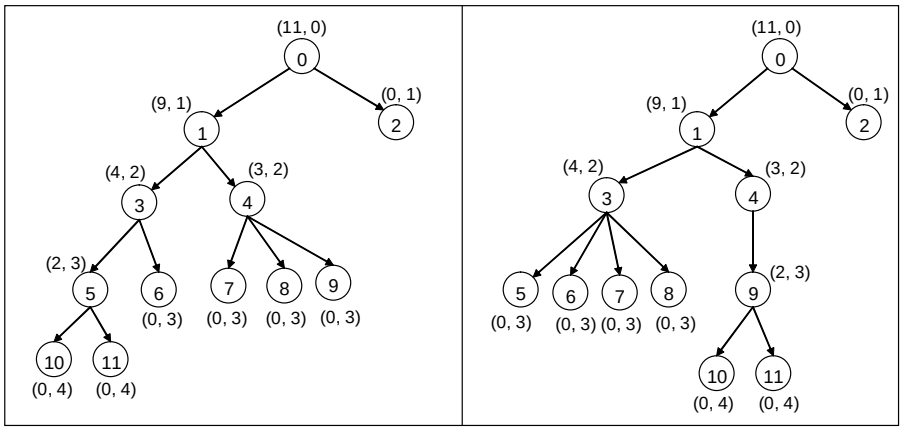

Figure 3.4: Two non-isomorphic trees whose transitive closures realize the same sequences

Complexity. The number of variables, i.e., the number of the candidate edges is $\mathcal{O}(N^2)$ (actually much less, since $\forall\langle u, v\rangle \in E \rightarrow Din(u) = Din(v) - 1$). The number of constraints is $\mathcal{O}(N)$, since for each node $v \in V$ we consider one constraint for its in-degree and possibly (only if $Dout(v) \neq 0$) another for its out-degree. Hence, the *expected* time complexity for *Simplex* to solve this LP instance is $\mathcal{O}(N^3)$.

3.3 Related Work

The problem of generating a graph given its degree sequence(s) dates back to [27, 29]. These works consider a given non-ascending sequence of non-negative integers and provide the necessary and sufficient condition for that sequence to be realizable (i.e., to exist a graph satisfying the given degree sequence). Authors considered only simple (undirected, without self-loops and multiple edges) graphs.

Definition 12 *[15, 23] Let n denote the number of nodes of the graph we wish to generate. Let u_i, $1 \leq i \leq n$ denote nodes and $d_1 \geq d_2 \geq ... \geq d_n$ intended degrees of these nodes. The necessary and sufficient condition for a degree sequence to be realizable is: $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} min\{k, d_i\}$*

Yannis Theoharis

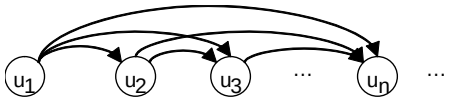

Figure 3.5: Algorithm [27, 29] Generating Undirected Graphs

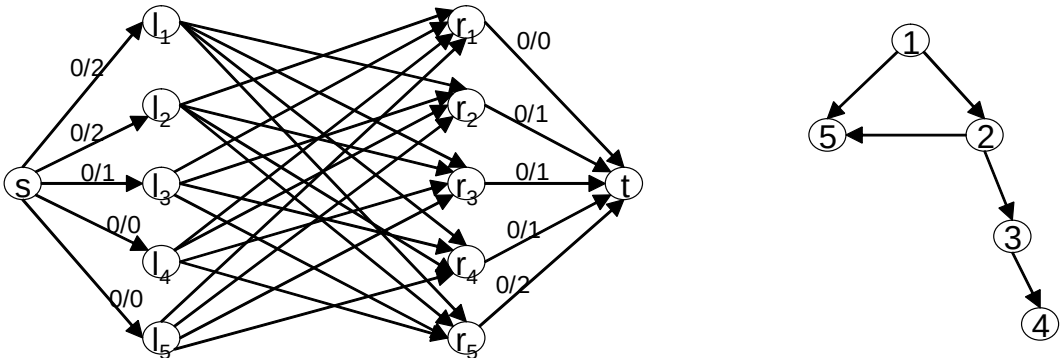

Figure 3.6: Algorithm [42] Generating Directed Graphs (left) and its output (right)

The algorithm is iterative and maintains the residual degrees of vertices. In each iteration it picks an arbitrary vertex u and adds edges from u to d_u vertices of highest residual degree, where d_u is the residual degree of u (see Figure 3.5). The residual degrees of the latter d_u vertices are updated appropriately. By connecting with d_u highest degree vertices the algorithm ensures that the necessary and sufficient condition holds for the residual problem instance. However, this algorithm is not suitable for synthetic SW schema generation. Specifically, for the generation of G_s we need to consider the transitivity of the edges. Moreover, both G_s and G_p are directed graphs (we consider two sequences).

Moreover, [42] proposed a reduction of the problem of generating *directed* graphs that satisfy simultaneously a given degree sequence for in-degrees and for out-degrees. The proposed method follows:

Let $d_{in} = (d_{in,1}, d_{in,2}, ..., d_{in,n})$ and $d_{out} = (d_{out,1}, d_{out,2}, ..., d_{out,n})$ be sequences of integers (in no particular sorted order), with $\sum_{i=1}^{n} d_{in,i} = \sum_{i=1}^{n} d_{out,i}$. We wish to construct a directed graph on n nodes, such that node u_i has $d_{in,i}$ incoming edges

and $d_{out,i}$ outgoing edges, $1 \leq i \leq n$. Consider the following graph. There is a source s, a sink t, a set of nodes $L = \{l_1, ..., l_n\}$ and a set of nodes $R = \{r_1, ..., r_n\}$. There is a link of capacity 1 directed from each l_i to each r_j, for $1 \leq i, j \leq n$ and $i \neq j$. There is a link of capacity $d_{out,i}$ directed from s to each l_i, for $1 \leq i, j \leq n$. Finally, there is a link of capacity $d_i n, i$ directed from each r_i to t, for $1 \leq i, j \leq n$. We may now consider integral maximum flows from s to t. If there is such a flow of value $\sum_{i=1}^{n} d_{in,i} = \sum_{i=1}^{n} d_{out,i}$, then the corresponding degree sequences are simultaneously realizable and the flow gives a directed graph that satisfies, simultaneously, in-degrees $d_{in,i}$ and out-degrees $d_{out,i}$.

Figure 3.6 illustrates an example for this algorithm, where $d_{in} = \langle 0, 1, 1, 1, 2 \rangle$ and $d_{out} = \langle 2, 2, 1, 0, 0 \rangle$. The maximum integral flow of the graph of Figure 3.6 (left), is $maxFlow = 5 = \sum_{d \in d_{in}} d = \sum_{d \in d_{out}} d$. Hence, d_{in}, d_{out} are simultaneously realizable and the graph of the maximum flow, which is drawn in Figure 3.6 (right), satisfies them.

This algorithm can be used for the generation of G_p, but not of that of G_s since: a) the transitivity of the edges needs to be considered (we consider as given the sequences of the transitive closure of G_s) and b) G_s is a DAG (i.e., no cycles are allowed). The reduction of the problem considers $2 * N$ nodes and N^2 edges, where N is the length of the given sequences (the number of nodes of the graph we want to generate). Several algorithms solving the Max-Flow problem have been proposed. They differ in their time complexity: a) the algorithm of [16] costs $\mathcal{O}(|V|^3)$, b) that of [21] costs $\mathcal{O}(|V|^2|E|)$ and c) that of [22] costs $\mathcal{O}(|V||E|^2)$. Since in our case asymptotically $|V| < |E|$ (since $\exists N_0$ *s.t.* $\forall N > N_0$, it holds that $2 * N < N^2$), we consider the first algorithm and its complexity as $\mathcal{O}(N^3)$.

Furthermore, authors in [8] studied the number of *simple undirected graphs* that realize a given sequence and proposed non-deterministic algorithms whose output is one them. Specifically, they focused on how each one of the graphs that realize the

given sequence is generated from their algorithms with as uniform distribution as possible. Although theoretically interesting, that work cannot be used in our case for the same reasons that the works [27, 29] can not.

Finally, in the bibliography there exists works, such as [13], studying the problem of generating graphs given an *expected* degree sequence, i.e., the generated graph is not guaranteed to realize the exact given sequences. These models are stochastic and are based on the convergence of the graph sequence to the given one, as long as the size of the graph increases. Since the graphs we want to generate is of relatively small size (i.e., their nodes number is mostly in 100-5,000), these models cannot be exploited (not to mention their inability to capture transitive edges).

To the best of our knowledge, in this Chapter we proposed the first algorithm to generate a synthetic DAG given the in/out-degree sequences of its *transitive closure*. In particular, we rely on a *Linear Programming (*LP*)* reduction, which allows us to generate *SW* graphs in polynomial time (using algorithms such as *Simplex* [10]). Adopting alternative *LP* formulations of graphs, we tackled both the problems of: a) generating a graph that simultaneously realizes the given sequences and b) generating a graph whose *transitive closure* simultaneously realizes the given sequences.

Chapter 4

Sampling Discrete Random Variables With A Power Law Distribution

In this chapter we elaborate on the problem of sampling *Discrete Random Variables (DRVs)* that follow a *power-law* distribution. This is needed to generate the in- and out-degree sequences of G_s^* and G_p by exploiting features of real SW schemas (Chapter 2). We will present two ways for the sampling. The former (Section 4.2) implements well known ideas and is independent of the nature of the PDF function. The latter (Section 4.3) is far from trivial and applies only when the VR function is a *power-law*. The former is useful in the case that the DRV is bounded by a maximum value, while the latter in the case that the sum of the bag of sampled values is predefined. The effectiveness of each method will be experimentally evaluated in Sections 4.2.1 and 4.3.1 respectively. The inherent relationship between the two methods is the result of the relation between PDF and VR functions, presented in Section 4.1.

4.1 The Relation between PDF and VR Functions

As noticed in [14] there exists a correlation between the *PDF* and the *VR* function of a *DRV* X. Let us, assume that the *VR* of X is a *power-law* with characteristic exponent $\frac{1}{b-1}$ for $i_0 \leq i \leq n + i_0$, i.e., $x_i = ci^{-\frac{1}{b-1}}$. The constant c is determined by the average value of X and i_0 depends on the maximum value of X. Then the *PDF* of X is a *power-law* with characteristic exponent b, i.e., $f(x) = c'x^{-b}$.

The appearance of i_0 has the meaning that if we ignore the first i_0 values of X, then the *PDF* function of X is also a *power-law*. In the sequel, we exploit the correlation between *PDF* and *VR* functions ignoring the observation about i_0 but keeping in mind that the consideration of the first values may decrease the *ACC* value of the *VR* plot. Specifically, we will examine whether $\gamma = \frac{1}{b-1}$ holds, where γ/b stands for the characteristic *power-law* exponent of the *VR*/*PDF* respectively. Consequently, we will examine whether $\gamma = \frac{1}{\delta}$, where δ is the characteristic exponent of the *CCDF*.

4.2 Sampling According to the PDF

Let X be a *DRV* and f its *PDF*. We consider that the range of X is a given finite set of successive integers in the range $[1, m]$, where m stands for the maximum allowed value for X. In order to sample a number N of values of this *DRV*, we adopt the following process.

We construct the plot (see Figure 4.1) of f (we plotted a *power-law* function, although the method is the same for any function). Let S_i be the space defined by the x-axis, the function f and the parallel to y-axis lines $x = i$ and $x = i+1$. We consider the surface of S_i, denoted by E_i, as the probability of $X = i$, i.e., $P(X = i) = E_i$, $\forall i \in [1, m]$. It is evident that $E_i = \int_i^{i+1} f(x)dx$. We then draw a new plot and define S'_i to be the space that is defined by the x-axis, the parallel to it line $y = 1$ and the parallel to y-axis lines $x = \sum_{j=1}^{i-1} E_j$ and $x = \sum_{j=1}^{i} E_j$. The final step is to sample N

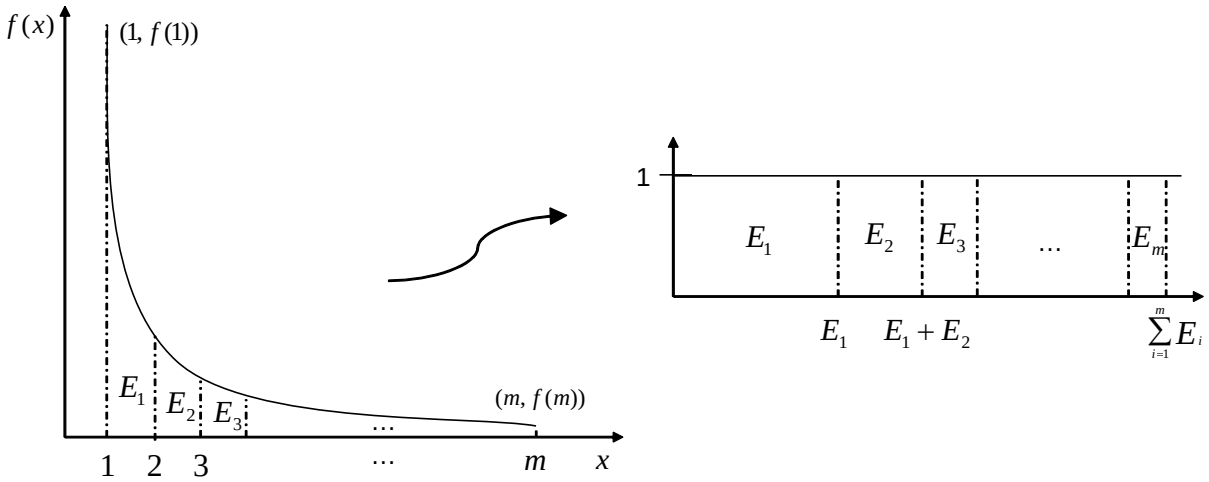

Figure 4.1: Sampling *DRV*s According to *PDF* function

pseudo-random real numbers in the range $[0, \sum_{i=1}^{m} E_i]$. For each sampled number k, we compute the i such that $k \in S'_i$, i.e., $\sum_{j=1}^{j=i-1} E_j \leq k < \sum_{j=1}^{j=i} E_j$. We then consider that the value i has been sampled.

In order to apply the above process in our case, we need to clarify some issues which depend on the specific function f. Firstly, since $f(x) = e^a x^{-b}$, $E_i = \int_i^{i+1} e^a x^{-b} dx = \frac{e^a}{1-b}[x^{1-b}]_i^{i+1}$. The characteristic exponent of the *power-law*, i.e. b, is assumed (for this method to apply) to be bigger than the unity and thus $1 - b$ can appear in the denominator. Moreover, the value e^a that appears in the formula of E_i is actually the $f(1)$, i.e., the probability of the most frequent value of X. Fortunately, the sampling is independent of e^a, because this quantity appears in E_i, $\forall i \in [1, m]$. What really matters for each i, is not the quantity E_i, but its relation with E_j, $\forall j \in [1, m]$, with $j \neq i$ and note that $\frac{E_i}{E_j} = \frac{[x^{1-b}]_i^{i+1}}{[x^{1-b}]_j^{j+1}}$, i.e., the sampling is depended only on b.

4.2.1 Experimental Evaluation

Since the sampling of *DRV*s that follow a *power-law* depends on the sampling of pseudo-random numbers, we need to examine the plots of the distribution that the

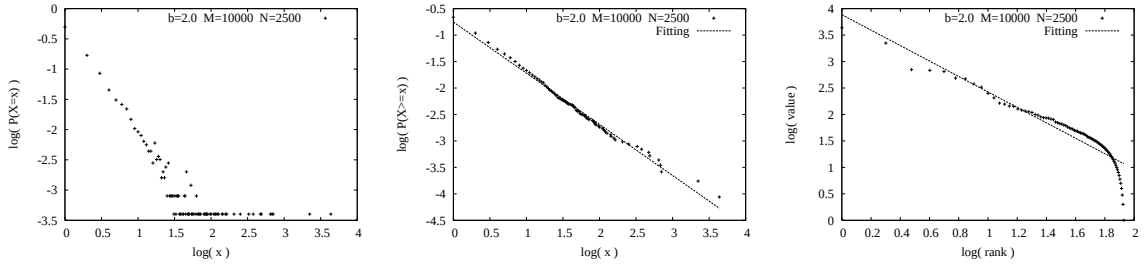

Figure 4.2: Sampling with $(b, M, N) = (2, 10000, 2500)$

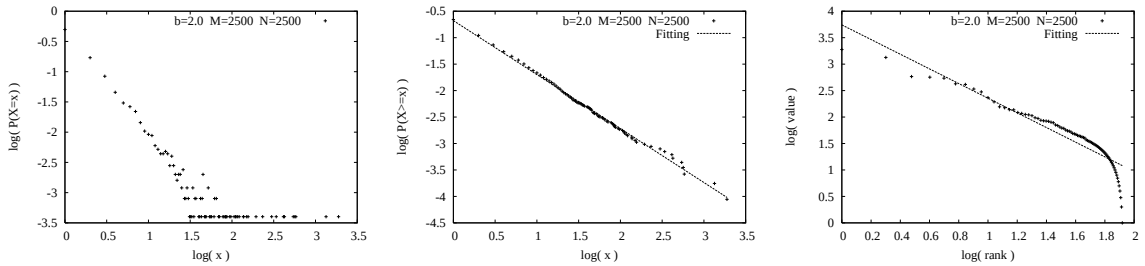

Figure 4.3: Sampling with $(b, M, N) = (2, 2500, 2500)$

generated values obey with respect to the given ones. Specifically, we consider as input of the above sampling method: a) the characteristic exponent, denoted by b, of the *power-law* that corresponds to the *PDF* function of X, b) the maximum value, denoted by M, that X is allowed to take and c) the number of values, denoted by N, that we want to sample.

The output of the sampling is a bag of values, which can be modeled using another *DRV*, denoted by Y. We can plot the *PDF* as well as *CCDF* and *VR* of Y in $log-log$ scale. Then we apply *Linear Regression* (as presented in Chapter 2) in order to understand to what extent the distribution of Y approximates (in terms of *ACC* values) a *power-law* and the characteristic exponent of it. Recall that we will use *CCDF* (and not *PDF*) of X, Y to compare their distributions due to the reasons explained in Chapter 2. Hence, if we give as input a characteristic exponent b for the *PDF* of X, the corresponding exponent of its *CCDF* is $b - 1$.

Figure 4.2, depicts the sampling obtained for $(b, M, N) = (2, 10000, 2500)$. The

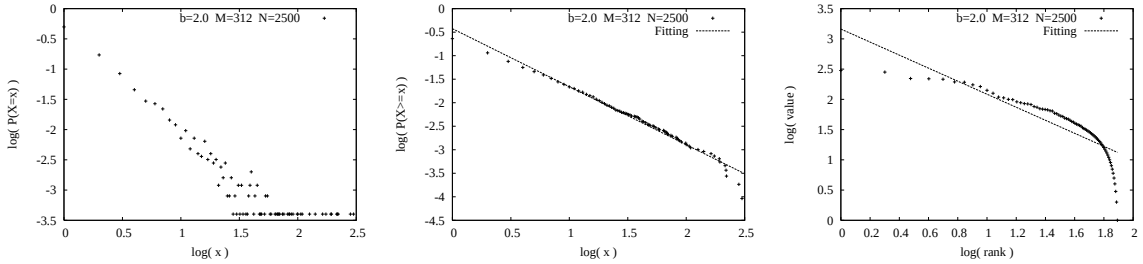

Figure 4.4: Sampling with $(b, M, N) = (2, 312, 2500)$

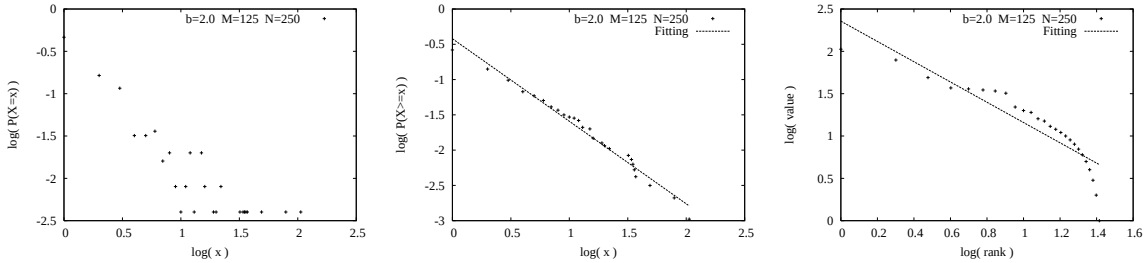

Figure 4.5: Sampling with $(b, M, N) = (2, 125, 250)$

CCDF exponent of X is $2 - 1 = 1$. Moreover, the *VR* exponent of X is $\frac{1}{2-1} = 1$. The *CCDF power-law* exponent of Y is 0.96 (very close to that of X), while the approximation of Y to a *power-law* is good (ACC=0.939). The plot corresponding to the *VR* of Y also approximates (ACC=0.967) a *power-law* with exponent 1.45. This actually diverges from the *VR power-law* exponent of X (which is 1). Moreover, from the plot of *VR* we can easily observe that the deviation of Y from a *power-law* exhibits a similar behaviour to that reported for *SW* schemas (see Chapter 2) and other real data sets [7].

We now keep b and N constant and decrease the value of M. Figure 4.3, shows the sampling obtained for $(b, M, N) = (2, 2500, 2500)$. The *CCDF* exponent of Y increases (compared to Figure 4.2) up to 1.01 (almost the same as the *CCDF* exponent of X). The same also holds for the *ACC* value, which increases up to 0.942. For the corresponding *VR* plot we can observe a kind of deviation from a *power-law* similar to that of Figure 4.2. The *ACC* value remains the same (i.e., 0.967) while the

	***CCDF* of Y**		***VR* of Y**	
(b, M, N)	**Exponent**	**ACC**	**Exponent**	**ACC**
$(2.0, 10000, 2500)$	0.96	0.939	1.45	0.967
$(2.0, 2500, 2500)$	1.01	0.942	1.38	0.967
$(2.0, 312, 2500)$	1.23	0.954	1.07	0.967
$(2.0, 125, 250)$	1.16	0.923	1.19	0.944

Table 4.1: Sampling DRVs based on their PDF function

characteristic VR exponent decreases down to 1.38.

We further decrease the value of M keeping constant the values of b and N. Figure 4.4, shows the sampling obtained for $(b, M, N) = (2, 312, 2500)$. The $CCDF$ exponent of Y increases (compared to that of the $CCDF$ of Figure 4.3) up to 1.23 (and thus diverges from the corresponding exponent of X which is 1). The ACC value also increases up to 0.954. Concerning the plot of the VR, the ACC value remains the same 0.967, while the computed characteristic exponent decreases down to 1.07 (and thus converges to the corresponding VR exponent of X).

We finally examine the case where both M, and N are small while b remains the same. Figure 4.5, shows the sampling obtained for $(b, M, N) = (2, 125, 250)$. The $CCDF$ exponent of Y is 1.16, while $ACC = 0.923$. Concerning the plot of the VR, the $ACC = 0.944$ while the computed characteristic exponent is 1.19. Table 4.1 summarizes the experiments of this Section (since $b = 2.0$, the $CCDF$ exponent of X is 1.0 and its VR exponent is 1.0). As one can observe, the $CCDF$ exponent of Y and the corresponding ACC value increases as long as M increases. On the other hand, the VR exponent of Y decreases (and the corresponding ACC value remains constant) as long as M increases.

4.3 Sampling According to the VR

Let X be a *DRV* and f its *VR* function. We consider that the range of X is a given finite set of successive integers in the range $[1, N]$, where N stands for the number of values of X that we want to sample. Let S be the bag of values we want to sample. Then, we consider that the quantity $\sum_{i=1}^{N} S(i)$, i.e., the *sum* of the values that will be sampled, is given. Since, f is a *power-law*, we can adopt the following process to construct S.

The $i-th$ biggest value, denoted by x_i, of the range of X is given by the formula: $x_i = e^a i^{-b}$. Specifically, the relation between two values of X, i.e., x_i and x_j, is $\frac{x_i}{x_j} = \frac{i^{-b}}{j^{-b}}$. Hence, for the constant factor e^a we can choose any constant number. For normalization purposes we will choose the quantity $\frac{sum}{\sum_{i=1}^{N} i^{-b}}$. This way the sum of the sampled values is guaranteed to be equal to $\sum_{i=1}^{N} S(i)$.

Definition 13 *Let X be a* DRV *that follows a* power-law *with* VR *exponent b. We can sample a bag S of N values of X of given* sum *as follows:*
$\forall i \in [1, N]$, $f(i) = \lfloor \frac{sum}{\sum_{i=1}^{N} i^{-b}} i^{-b} \rfloor$, *where $\lfloor x \rfloor$, stands for the integral part of x.*

The intuition behind the above definition is that $\lfloor f(i) \rfloor$ and $\lfloor f(i+1) \rfloor$ are different integers if i is small, but the probability that they are the same increases as long as i increases. Another way to grasp this idea is to observe that $\lfloor f(i) \rfloor - \lfloor f(i+1) \rfloor$ decreases as long as i increases. Hence, the number m of successive i's for which $\lfloor f(i) \rfloor - \lfloor f(i+m) \rfloor \leq 1$ increases as long as i increases. An indicative example follows.

Example 3 *Let X be a* DRV *that follows a* power-law *with* VR *exponent $b = 1$ and consider we want to sample* 100 *values of sum* 1000. *Bellow we write each sampled value, k, preceding by the range of i such that $\lfloor f(i) \rfloor = k$. (1, 192), (2, 96), (3, 64), (4, 48), (5, 38), (6, 32), (7, 27), (8, 24), (9, 21), (10, 19), (11, 17), (12, 16), (13, 14), (14, 13), (15-16, 12), (17, 11), (18, 10), (19-20, 9), (21-23, 8), (24-26, 7), (27-31, 6), (32-37, 5), (38-47, 4), (48-63, 3), (64-95, 2), (96-100, 1).*

Yannis Theoharis

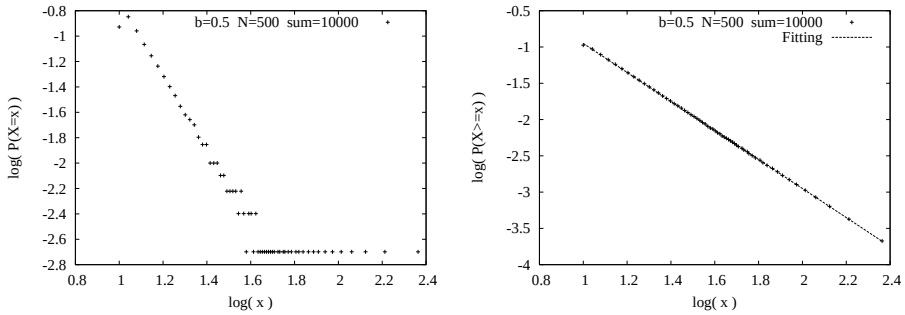

Figure 4.6: Sampling with $(b, N, sum) = (0.5, 500, 10000)$

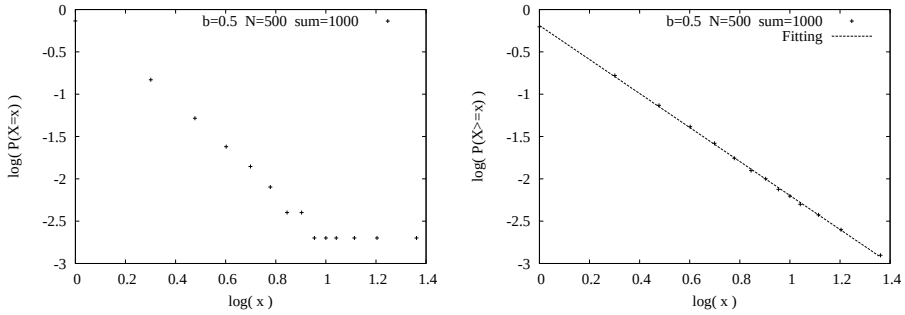

Figure 4.7: Sampling with $(b, N, sum) = (0.5, 500, 1000)$

We should notice that the consideration of $\frac{sum}{\sum_{i=1}^{N} i^{-b}}$ as the constant factor of the *VR power-law* function would normalize the x_i's, if they were real numbers. Since, we consider them to be integers, the final sum of the sampled values, i.e., $\sum_{i=1}^{N} S(i)$, is slightly different from the given sum.

4.3.1 Experimental Evaluation

The bag S of sampled values can be modeled by a *DRV*, denoted by Y. In order to contrast the *PDF* and *CCDF* functions of Y to those of X, we considered two experiments that show the effect of b values and of the sum of the values that will be sampled, denoted by sum.

Figure 4.6 shows the sampling obtained for $(b, N, sum) = (0.5, 500, 10000)$. In that

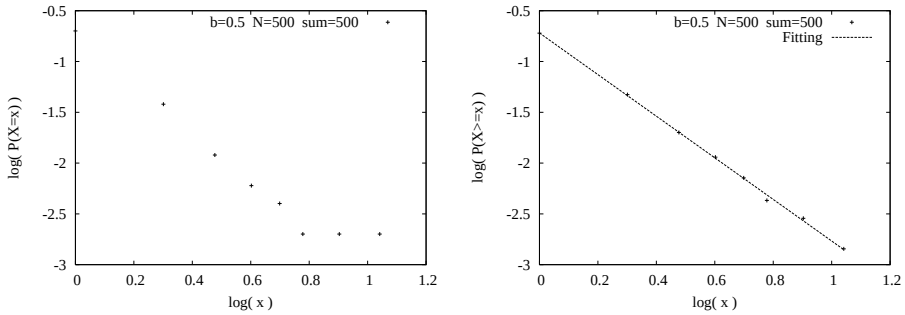

Figure 4.8: Sampling with $(b, N, sum) = (0.5, 500, 500)$

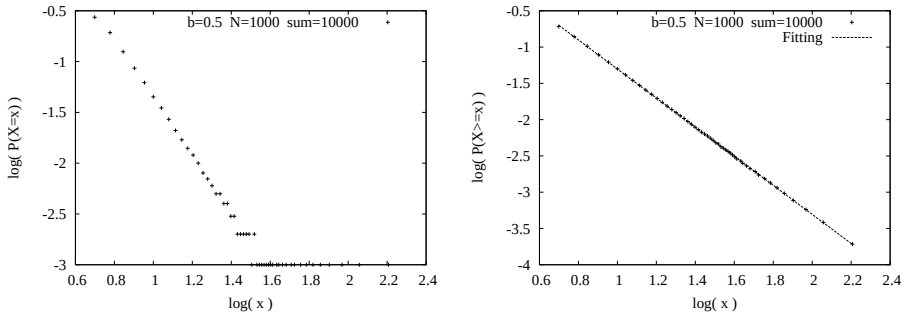

Figure 4.9: Sampling with $(b, N, sum) = (0.5, 1000, 10000)$

case the *CCDF* exponent of X is $\frac{1}{\frac{1}{2}} = 2$. The *CCDF* exponent of Y is 2.0, i.e., same as for X, while the approximation of *CCDF* to a *power-law* is excellent ($ACC = 0.98$). Decreasing the *sum* down to $1,000$ (Figure 4.7) results in the same *CCDF* exponent, i.e., 2.0, but the *ACC* value decreases down to 0.92. To complete this group of experiments we present the sampling obtained for $(b, N, sum) = (0.5, 500, 500)$ in Figure 4.8, for which the *CCDF* exponent is 2.04 while the *ACC* value decreases down to 0.88. The main conclusion that can be drawn is that *CCDF* of Y coincides with that of X. However, the approximation of *CCDF* to a *power-law* (*ACC* value) decreases as long as *sum* increases.

To realize the effect of b values, we keep constant N and *sum* and we vary b. Figure 4.9 shows the sampling obtained for $(b, N, sum) = (0.5, 1000, 10000)$. In that case the *CCDF* exponent of X is $\frac{1}{\frac{1}{2}} = 2$. As one can see in Figure 4.9, the exponent of the

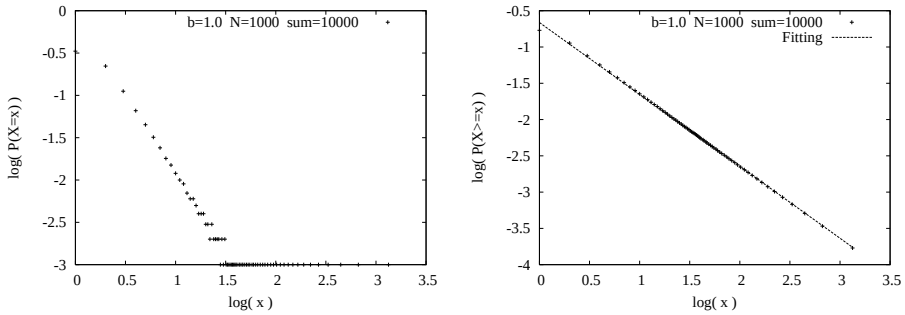

Figure 4.10: Sampling with $(b, N, sum) = (1.0, 1000, 10000)$

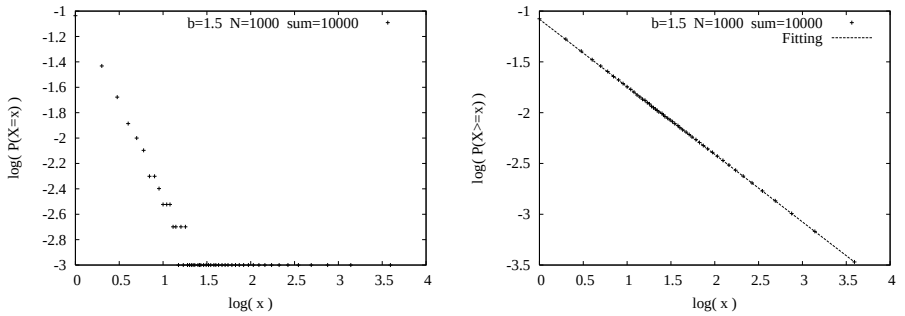

Figure 4.11: Sampling with $(b, N, sum) = (1.5, 1000, 10000)$

CCDF of Y is 2.0 while the *ACC* value is 0.97. Figure 4.10 illustrates the distributions of the corresponding samplings obtained when b is increased up to 1.0 (and thus the *CCDF* exponent of X is $\frac{1}{1} = 1$). The *CCDF* exponent of Y decreases down to 0.99 and the *ACC* value also decreases down to 0.94. We further increase b up to 1.5 (and thus the *CCDF* exponent of X is $\frac{1}{1.5} = 0.66$) and the distributions corresponding to the obtained sampling are depicted in Figure 4.11. The *CCDF* exponent of Y decreases down to 0.66 and the *ACC* value also decreases down to 0.91. We complete this group of experiments by increasing b up to 2.0 and thus the *CCDF* exponent of X is $\frac{1}{2}$ (Figure 4.12). The *CCDF* exponent decreases down to 0.49 and the respective *ACC* value also decreases down to 0.88. The main conclusion that can be drawn from these experiments (see Table 4.2) is that *CCDF* of Y coincides with that of X. However, the approximation of *CCDF* to a *power-law* (*ACC* value) decreases as long

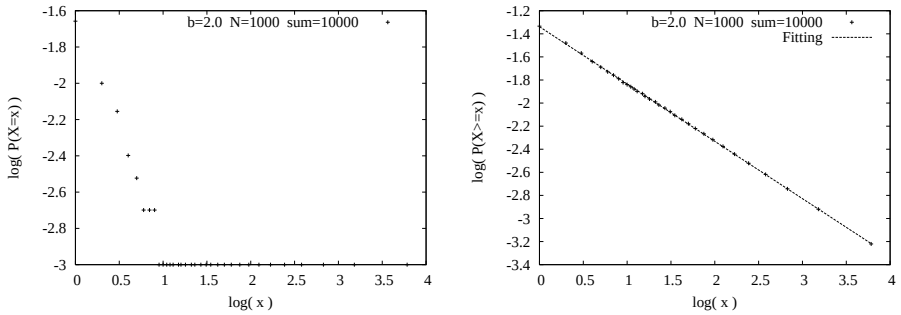

Figure 4.12: Sampling with $(b, N, sum) = (2.0, 1000, 10000)$

as b increases.

	***CCDF* of X**	***CCDF* of Y**	
(b, N, Sum)	**Exponent**	**Exponent**	**ACC**
$(0.5, 500, 10000)$	2.0	2.0	0.98
$(0.5, 500, 1000)$	2.0	2.0	0.92
$(0.5, 500, 500)$	2.0	2.04	0.88
$(0.5, 1000, 10000)$	2.0	2.0	0.97
$(1.0, 1000, 10000)$	1.0	0.99	0.94
$(1.5, 1000, 10000)$	0.66	0.66	0.91
$(2.0, 1000, 10000)$	0.5	0.49	0.88

Table 4.2: Sampling DRVs based on their VR function

4.4 The Usefulness of Both Sampling Methods

The existence of the two ways presented in Sections 4.2, 4.3, is important for both theoretical and practical reasons. By a theoretical viewpoint, it provides the means to realize the correlation between the PDF and the VR function of a DRV that follows a *power-law*. Moreover the fact that selecting appropriate values for the parameters of both methods we can produce deviations from *power-law*s that appear to SW schemas, provides a hint for a more elaborated interpretation of the conditions

of the evolution of *SW* schemas that result in the emergence of *power-law*s for various kinds of distributions. This is part of our future work.

By a practical viewpoint, we notice that in order to produce the degree sequences of G_s^*, G_p, the standard sampling method (Section 4.2) based on the *PDF* function is not sufficient. For instance, in order to generate the in-degree sequence of G_s^* we can exploit the observation (see Section 2.2.4.4) that the *VR* function of *Level of Classes* distribution is a *power-law*. However, its *PDF* approximates the uniform distribution (it is rare for two different levels to have the same number of classes located at them). Using the sampling method that is based on the *VR* function we can sample a bag of values, which guarantee that the *VR* is the given *power-law* while the respective *PDF* is a constant function (i.e., uniform distribution). Such issues will become more clear in Chapter 5.

Chapter 5

Synthetic SW Schema Generation

In this Chapter we elaborate on the generation of synthetic SW schemas based on the findings of Chapter 2. To this end, we consider as given the number of schema classes, denoted by N_c, and of properties, denoted by N_p. Then, we essentially need to generate the subsumption graph, $G_s = (V, E_s)$ and the property graph $G_p = (V, E_p)$, where V is a set of nodes corresponding to classes with $|V| = N_c$, $E_s \subseteq V \times V$ a set of edges representing subsumption relationships and $E_p \subseteq V \times V$ a set of edges representing relationships between classes, with $|E_p| = N_p$.

The remainder of this Chapter is organized as follows: Section 5.1 focuses on the generation of G_p, while Section 5.2 of G_s. Then Section 5.3 aims at combining G_p and G_s to generate a synthetic SW schema. Finally Section 5.4 presents the results of the experimental evaluation of the generation of synthetic SW schemas.

5.1 Generating The Property Graph

In order to generate G_p we additionally consider as given the characteristic *power-law* exponent, denoted by b, of the *VR* function of the *total-degree* distribution. Table 5.1 sketches the algorithm that generates G_p. Bellow, we explain in detail each of its steps.

Algorithm $GenerateG_p$
Input. b: the VR exponent of the total-degree distribution, N_c/N_p: the number of schema classes/properties respectively, p_0: percentage of classes that are neither domain or range of any property.
(1) $D := VRSampling(b, \lfloor(1-p_0)N_c\rfloor, 2N_p)$; $D_{in} := \emptyset$; $D_{out} := \emptyset$; (2) for $i = 1$ to N_c do $D_{in}[i] = D_{out}[i] = \frac{1}{2} * D[i]$; (3) Chose randomly nodes and attach on them a set S of $\lfloor 0.126 * N_p \rfloor$ self-loops (modify D_{in} and D_{out}); (4) Chose randomly pairs of nodes and attach on them a set M of $\lfloor 0.177 * N_p \rfloor$ multiple edges (modify D_{in} and D_{out}); (5) $E :=$ the set of edges of the solution of the *LP1* instance corresponding to D_{in} and D_{out}; (6) $E_p = E \uplus S \uplus M$;

Table 5.1: Algorithm $GenerateG_p$

Using the sampling method based on VR function that was presented in Section 4.3 we can generate the *total-degree* sequence, denoted by D, of the G_p based on b (step 1). D is the vector sum of Din and $Dout$, i.e. $\forall v \in V$, $D(v) = Dout(v) + Din(v)$. We choose to generate D instead of $Dout$ and Din, because the percentage of real SW schemas that approximate a *power-law* for the *total-degree* distribution is bigger than the corresponding percentages for *out-* and *in-degree* distributions (see Section 2.2.3). Moreover, there exists a strong correlation between the out- and in-degrees of nodes. Note that we choose to use the sampling method based on the VR and not on the PDF function of the total-degree distribution (see Section 4.3). This is due to the fact that the *sum* of the sampled values is determined by N_p, i.e., the number of schema properties. Specifically, since each edge $\langle u, v\rangle$ is counted twice in D (i.e., once in the $D(u)$ and once in the $D(v)$), it holds that $\sum_{v\in V} D(v) = 2N_p$. We should also

stress that there exist classes in real SW schemas that are neither domain nor range of any property, i.e., their total-degree is 0. Although the percentage, denoted by p_0, of such classes can be a parameter of the generator, we report that in average case 50% of the classes of each schema have total degree 0. D can be generated using the aforementioned sampling method with parameters $(b, N, sum) = (b, \lfloor(1 - p_0)N_c\rfloor, 2N_p)$.

As a next step (2), we need a method that splits D into $Dout$ and Din. This method should exhibit the following characteristics :

- $\sum_{v \in V} D(v) = \sum_{v \in V} Dout(v) + \sum_{v \in V} Din(v)$,
- $\sum_{v \in V} Dout(v) = \sum_{v \in V} Din(v)$, because there is no graph that simultaneously realizes $Dout$ and Din if $\sum_{v \in V} Dout(v) \neq \sum_{v \in V} Din(v)$, since every edge is counted once in $Dout$ and once in Din,
- nodes of high out-degree should also be of high in-degrees and similarly nodes of low out-degree should also be of low in-degree (see Section 2.2.3.3).

A simple formula that conforms to all the above features is $\forall v \in V,\ Dout(v) = Din(v) = \frac{1}{2}D(v)$ (we can tackle odd values of $D(v)$ by considering its value as $\lfloor D(v)\rfloor$ for one of the two sequences and as $\lceil D(v)\rceil$ for the other).

In Section 3.2.1 we proposed the reduction of the generation of a directed graph without self-loops or multiple edges to an *LP1* instance. However, considerable percentages of edges of real SW are self loops, i.e., 12.6%, or multiple edges 17.7% (see Section 2.2.3). Hence, we will choose nodes (whose both out- and in-degree are bigger than 1) to assign them self-loops and pairs of nodes (whose both out- and in-degree are bigger than 2) to assign them multiple edges (steps 3 and 4). Since we do not have the means to distinguish nodes according to the number of their self-loops or multiple edges, we will distribute them uniformly. Let S be the set of the chosen self-loops and M the set of the chosen multiple edges. This process will modify the $Dout$ and Din. Specifically, for each self-loop $\langle u, u\rangle$, it will be

$Dout(u) := Dout(u) - 1$ and $Din(u) := Din(u) - 1$. Similarly for each multiple edge $\langle u, v\rangle$, it will be $Dout(u) := Dout(u) - 1$ and $Din(v) := Din(v) - 1$.

The modified *Dout* and *Din* can now be used for the generation of a directed graph without self-loops and multiple edges (step 5) according to the method presented in Section 3.2.1.

Let E be the set of edges obtained from the solution of the corresponding *LP1* instance. We will add to the generated graph the self-loops and multiple edges chosen in steps (3) and (4), i.e., we will consider as the set of graph edges, denoted by E_p, the bag-union $E \uplus S \uplus M$ (step 6).

5.1.1 Attributes

In addition, we need to consider edges which have as destination *Literals* (e.g., String, Integer), i.e., attributes of classes. After generating G_p we can add to the set of its nodes V, the Literal types, as specified in *XML schema*[1]. Then we connect them to the pre-existent nodes of G_p under the condition that the total-degree sequence of G_p remains the same. This constraint can be satisfied by replacing a number k of edges of the form $\langle u, v\rangle$, where u, v correspond to classes, such that $Dout(v) = 0$ (nodes representing literal types should have out-degree zero) and $Din(v) = 1$, with edges of the form $\langle u, w\rangle$, where w is a node that corresponds to a *Literal* type. The number k of the attributes can be given as input (e.g., as a percentage of N_p).

5.2 Generating The Subsumption Graph

In order to generate G_s we additionally consider as given the characteristic exponent, denoted by b, of the *power-law* of the *PDF* function of the *class descendants* distribution. Moreover, we will consider as given the depth, denoted by d, of G_s as

[1] *http://www.w3.org/XML/Schema*

Algorithm $GenerateG_s$
Input. b: the PDF exponent of the class descendants distribution, N_c: the number of schema classes.
(1) $Dout := PDFSampling(b, N_c - 1, 0.25 * N_c)$; (2) if $N_c - 1$ is not in $Dout$ $D_{out} := D_{out} \cup \{N_c - 1\}$; (3) for $i = 1$ to $\lfloor 0.75 * N_c \rfloor - 1$ do $D_{out} := D_{out} \uplus \{0\}$; (4) $\gamma := 0.0017 * N_c + 1.36$; (5) $S := VRSampling(\gamma, d, N_c)$ (Order S in descending order); (6) $k = \lfloor \frac{3}{4} * d \rfloor$; (7) $L :=$ the set of G_s^* levels ordered according to $f(l) = \lvert k - l \rvert$; (8) for $i = 1$ to d do for $j = 1$ to $S[i]$ do $Din.append(L[i])$; (9) Assure that $\sum_{v \in V} Dout(v) = \sum_{v \in V} Din(v)$; (10) Order $Dout$ in descending and Din in ascending order; (11) $E_s^* :=$ the set of edges of the solution of the $LP2$ (or $LP3$ if G_s is considered to be a tree) instance corresponding to D_{in} and D_{out}; (12) $E_s :=$ the transitive reduction of E_s^*;

Table 5.2: Algorithm $GenerateG_s$

well as the information whether G_s should be a DAG or a tree. Table 5.2 sketches the algorithm that generates G_s. Bellow, we explain in detail each of its steps.

5.2.1 Generating Dout of G_s^*

Using the sampling method based on *PDF* function that was presented in Section 4.2, we can generate the *out-degree* sequence of G_s^*. Specifically, the biggest allowed value is $N_c - 1$, since the *root* node (that corresponds to the root of the class hierarchy) has $N_c - 1$ descendants. Furthermore, on average the 75% of classes of real *SW* are leaves (see Section 2.2.4), i.e. their out-degree is 0. Thus we choose the

following parameters $(b, M, N) = (b, N_c - 1, 0.25 * N_c)$ for the sampling (step 1). Note that we choose the sampling method based on the *PDF* instead of the *VR* function, because we do not know the sum of the sampled values (needed as input for the *VR*-based method) and even if we knew it, we could not guarantee that the value $N_c - 1$ would be contained in the bag of sampled values (e.g. the given sum may have not allowed it).

Note that the value $N_c - 1$ will not certainly be contained in the bag of sampled values. This is due to the fact that $P(X = x)$ decreases as long as x increases. Hence, we need to add the value $N_c - 1$ in the bag (step 2). In order to obtain a sequence of length N_c we add $0.75 * N_c - 1$ times (corresponding to the 0.75% of leaf classes) the value 0 (step 3).

5.2.2 Generating Din of G_s^*

The generation of *Din* of G_s^* is not as easy as that of *Dout*. Firstly, we can not use the *class ancestors* distribution, because real *SW* schemas do not follow a *power-law* for it (see Section 2.2.4). Instead, we can exploit the findings about the *level of classes* distribution. Its *VR* for real *SW* schemas approximates a *power-law*. Moreover, the characteristic *VR* exponent, denoted by γ, approximately depends linearly (see Section 2.2.4) on the number of nodes (classes). Hence, we can produce a sequence of values, which correspond to numbers of nodes that are located at a specific level, by sampling according to the *VR* function of the *level of classes* distribution with parameters $(b, N, sum) = (\gamma, d, N_c)$, since we want to distribute N_c nodes to d levels (steps 4 and 5). Note that since N_c is orders of magnitude bigger than d the sampled values comprise a set (i.e., each value presents only once), denoted by S.

However, we still do not know to which level a specific value of the sampled set corresponds. We exploit the finding that the level, denoted by k, at which the maximum number of nodes are located is approximately $0.75 * depth$ for real

SW schemas (see Section 2.3). Let x_i be the $i-th$ biggest value of S. We will define an one-to-one function m that maps each i to one and only level, denoted by l, of G_s^*. We order levels according to their distance to the most populated level, i.e., $L = \langle k, k+1, k-1, k+2, k-2, ... \rangle$ (step 7). L has $d-k$ elements obtained from the sum $k+m$ and $k-1$ elements obtained from the difference $k-n$. Then, we consider $m(i) = L(i)$. For instance, consider that $d = 4$, then $m(1) = 3$, $m(2) = 4$, $m(3) = 2$, $m(4) = 1$. The generation of Din directly follows, since the level at which a node is located coincides with its in-degree (step 8).

5.2.3 Assuring that Dout and Din can be Simultaneously Realizable

A final step of $Dout$ and Din modification is needed in order to achieve that $\sum_{v \in V} Dout(v) = \sum_{v \in V} Din(v)$ (step 9). This can be achieved by modifying the frequencies of 0 and 1 in $Dout$. We should stress that it may not be possible to achieve $\sum_{v \in V} Dout(v) = \sum_{v \in V} Din(v)$ for every given depth d. The range of the allowed values for d is depended on N_c, b and the percentage of d of the most populated level (i.e., 0.75, since we observed that $k = 0.75$ for real SW schemas). The exact computation of this range is left as future work.

Note that we could ignore this step and repeating the process of generating the two sequences from the beginning, until $\sum_{v \in V} Dout(v) = \sum_{v \in V} Din(v)$ holds. However, it is not reasonable to spend time for that, not to mention that it is not guaranteed that we will ever reach a state that $\sum_{v \in V} Dout(v) = \sum_{v \in V} Din(v)$ holds (e.g. the seed of the pseudo-random number generation needed for the generation of Dout may prevent such a case). Finally, we need to order $Dout$ in descending and Din in ascending order (see Lemma 1).

In order to generate G_s, the modified $Dout$ and Din sequences are given (step 11) as input to the method presented in Section 3.2.2.1 if G_s should be a DAG (reduction

to an *LP2* instance) or to that presented in Section 3.2.2.3 if G_s should be a tree reduction to an *LP3* instance). The set of edges, denoted by E_s^*, obtained from the solution of either *LP2* or *LP3* instance corresponds to G_s^*. Finally, we compute the transitive reduction of E_s^* to obtain the set of edges of G_s, denoted by E_s (step 12).

5.3 Combining The Property and The Subsumption Graph

At this step we consider that $G_s : (V_s, E_s)$ and $G_p : (V_p, E_p)$ are generated. Since the two graphs have the same set of nodes we should define an one-to-one function $h : V_s \rightarrow V_p$ that maps each node of G_s to one and only node of G_p. To this end, we exploit the combinatoric finding that was presented in Sections 2.2.6, 2.3, i.e., that nodes with high out-degree in the G_p are located highly in the G_s. Specifically, let k be the level of G_s^* at which the source nodes (corresponding to domains of properties) of most edges of G_p are located. We order the nodes of V_p in descending out-degree order and we reach a list P. Also, let V_i be the set of nodes of G_s^* located at level i, i.e., $V_i = \{v \in V_s \mid Din(v) = i\}$. Then, we map each node of the first $|V_k|$ nodes of P to one and only node of V_k. Similarly, we map the next $|V_{k+1}|$ nodes of P to one and only node of V_{k+1} and the next $|V_{k-1}|$ nodes of P to one and only node of V_{k-1}. This process continues until we map the nodes of V_0 and of V_d to nodes of V_p.

5.4 Experimental Evaluation

In this Section we experimentally evaluate the synthetic *SW* schema generator presented in this thesis on two axis, namely, the effectiveness and the efficiency. Concerning the former we present a detailed example of the generation of a synthetic *SW* schema based on the methods presented in previous Chapters. On the course of

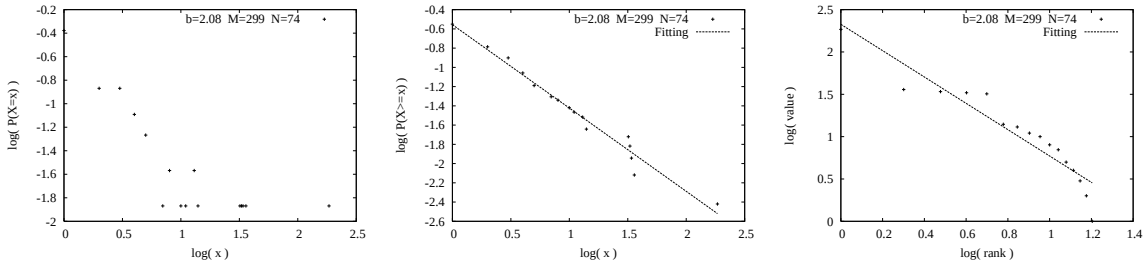

Figure 5.1: Sampling with $(b, M, N) = (2.08, 299, 0.25 * 300)$ to generate the out-degree sequence of G_s^*

this process we will comment on details that have not been covered yet. Concerning the latter we focus on time and memory requirements of the reduction of the problem of generating G_s and G_p to the *LP* problem. Moreover, we measure the number of variables and the number of constraints of the produced *LP* instances.

5.4.1 Effectiveness

We consider that we want to generate a schema of 300 classes and of 1000 properties. Moreover we consider that the *power-law* exponent of the total-degree *VR* function is 0.5, and that of the class descendants *PDF* is 2.08. Finally, we consider that G_s is a DAG and its depth is 4.

5.4.1.1 Generating The Subsumption Graph

In order to generate the *Dout* of G_s^* we use the sampling method (see Section 4.2) based on the *PDF* of the class descendants distribution with parameters $(b, M, N) = (2.08, 299, 0.25{*}300)$. Figure 5.1 shows the *PDF* and *CCDF* functions that correspond to the bag of sampled values. We explicitly add the value 299 once and the value 0 $0.75 * 300 = 225$ times. We reach the final *Dout*. The sum of *Dout* is 828.

We now generate the in-degree sequence of G_s^*. We consider the *VR* function of the

level of classes distribution with exponent $\gamma = 0.00017 * 300 + 1.36 = 1.411$ (see Section 2.2.4.4). To find the number of nodes located at the 4 levels of G_s we use the sampling method based on the *VR* function with parameters $(b, N, sum) = (1.411, 4, 299)$. We choose the *sum* to be 299 and not 300, because we consider one node (the root) with in-degree equal to 0. We obtain the sequence $L = \langle 173, 65, 36, 24\rangle$, i.e., the most populated level has 173 nodes, the second most populated has 65, etc. We observe that $\sum_{x \in L} x = 298$ and not 299 (see Section 4.3). We need to add one more node to a specific level. We choose to increase the nodes of the most populated level. Hence, we reach the sequence $L = \langle 174, 65, 36, 24\rangle$. As a next step, we map numbers of classes to specific levels, i.e., we compute the function m. The most populated level is $0.75 * 4 = 3$. Hence, $m(1) = 3$, $m(2) = 4$, $m(3) = 2$, $m(4) = 1$, i.e., the third level has 174 nodes, the fourth has 65, the second 36 and the first 24. Since, the level at which a node is located coincides with its in-degree in G_s^* and thus we have generated the *Din*. However, $\sum_{d \in Din} d = 820 \neq 828 = \sum_{d \in Dout} d$. To achieve that $\sum_{d \in Din} d = \sum_{d \in Dout} d$ we will modify *Dout*. Since $\sum_{d \in Din} d < \sum_{d \in Dout} d$ we consider $\sum_{d \in Dout} d - \sum_{d \in Din} d$ nodes of out-degree equal to 1 as nodes with out-degree equal to 0. Finally, we order *Dout* in descending and *Din* in ascending order (see Lemma 1).

The produced *Dout* and *Din* are given as input of the method presented in 3.2.2.1. The *LP2* instance aims at generating G_s^*. However, the solution is not integral. There exist coordinates (corresponding to edges of G_s^*) that are real numbers in $(0, 1)$. To reach a final set of edges we consider a threshold value $T \in [0, 1]$: an edge $\langle u, v\rangle$ exists iff $x_{u,v} \geq T$. The use of such heuristic for deciding whether non-integer edges exist or not may violate transitivity of G_s^*, i.e., G_s^* may not be a *transitively closed* graph, since some transitive edges may miss. For instance, consider that $T = 0.6$ and in the *LP* solution $x_{u,v} = 0.7$, $x_{v,w} = 0.8$ and $x_{u,w} = 0.5$. Then, $x_{u,v} + x_{v,w} - x_{u,w} = 1 \leq 1$, as required for the edges of a *transitively closed* DAG (see Lemma 2). However, to make the solution integral, we will consider that $x_{u,v} = 1$, $x_{v,w} = 1$ and $x_{u,w} = 0$. By

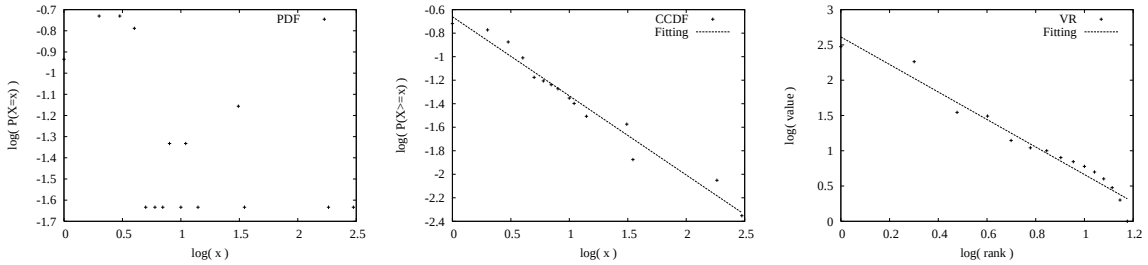

Figure 5.2: The functions corresponding to the out-degree sequence of the generated G_s^*

transitive reduction of G^* we obtain G. We can then compute the approximation (or distance) of the degree sequences of the *transitive closure* of G with the given ones. Concerning the value of T, we should notice that a big threshold value (i.e., close to 1), e.g., $T = 0.9$, decreases the probability of missing transitive edges, but ignores many edges and thus the generated *Dout* and *Din* roughly diverge from the given ones. On the other hand, a small threshold value (i.e., close to 0), e.g., $T = 0.1$, results in bigger number of edges than in the given sequences. After experiments we found that the value $T = 0.6$ leads to best approximations with respect to the given sequences *Dout* and *Din*. This is due to the fact that the missing transitive edges from the produced G_s^* are minimized for $T = 0.6$, while the number of generated edges converges to those of the given sequences. The *PDF*, *CCDF* and *VR* functions of the transitive closure of the generated G_s are shown in Figure 5.2.

5.4.1.2 Generating The Property Graph

In order to generate the total-degree sequence D we use the sampling method based on the *VR* function of the *total-degree* distribution with parameters $(b, N, sum) = (0.5, 150, 2000)$. Figure 5.3 shows the *PDF* and *CCDF* functions that correspond to the bag of sampled values. As a next step, we split D into *Dout* and *Din*. After, choosing nodes to assign on them $0.126 * 1000 = 126$ self-loops and pairs of nodes

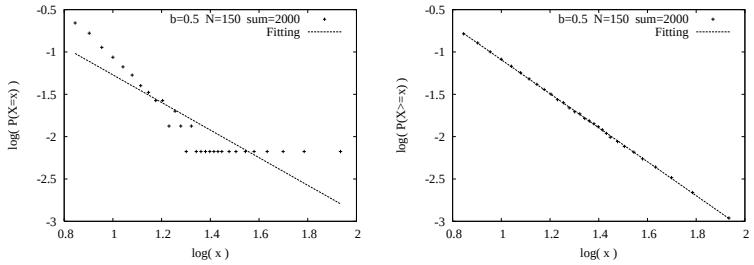

Figure 5.3: The *PDF* and *CCDF* function corresponding to the total-degree sequence of the generated G_p

to assign $0.177 * 1000 = 177$ multiple edges, we use the modified *Dout* and *Din* for the generation of a directed graph without self-loops and multiple edges according to the method presented in Section 3.2.1. It should be noticed that the solution of the *LP1* instance is always integral (see Section 3.2.1) and thus the generation of G_p is finished.

5.4.2 Efficiency

In this Section we report the time and memory requirements of the LP instances produced for the generation of synthetic *SW* schemas of various size. To solve the LP instances we used the academic software Soplex [54]. The measurements reported below are strongly depended on the efficiency of Soplex. Improved measurements can be obtained by using more sophisticated *simplex* implementations, such as the commercial CPLEX [33]. We also measure the number of variables and the number of constraints of the produced *LP* instances. These measurements show the complexity of the reduction of our problem to the *LP* instances and are independent of the specific software implementation of *simplex*, which was used to solve them. Experiments were carried out on a PC with a Pentium IV 3.2GHz processor and 512MB of main memory (1GB of Virtual Memory), over Suse Linux (v9.1).

Figure 5.4 shows the number of variables and constraints of the LP1 instances

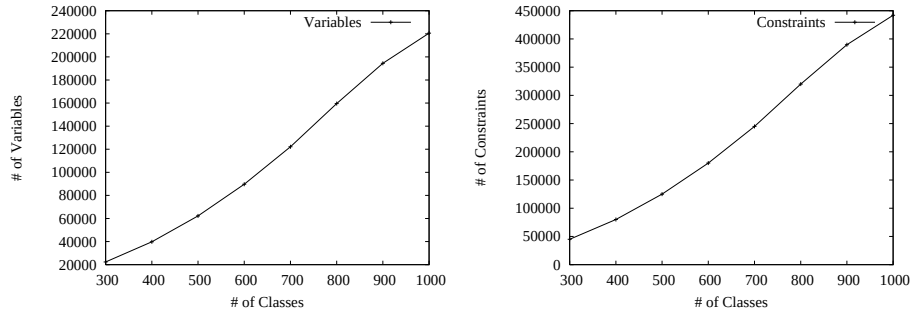

Figure 5.4: Number of Variables (left) / Constraints (right) of the *LP1* instances produced to generate G_p

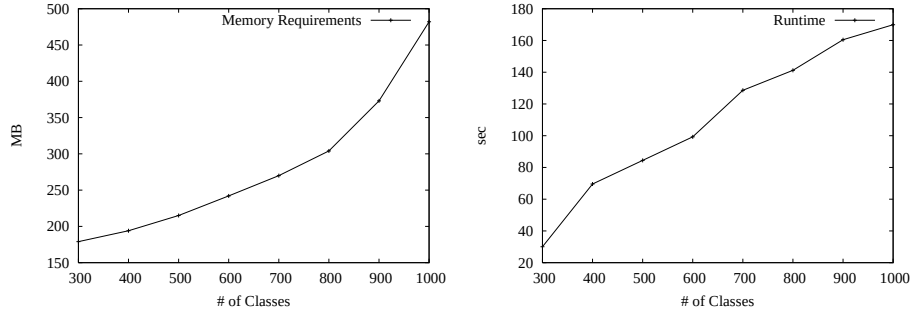

Figure 5.5: Memory Requirements (left) / Runtime (right) of the *LP1* instances

that are produced to generate G_p for various number of classes, namely $300 - 1000$. Specifically, we considered as constant $b = 0.6$ and $N_p = 1000$ and we vary N_c. Since we assume that 50% of nodes have zero total-degree, the number of variables is $numOfVars = \frac{N_c}{2} * (\frac{N_c}{2} - 1)$ and the number of constraints is $2 * numOfVars$. Note that for $N_c = 900$ or 1000, the percentage of zero total-degree nodes is slightly bigger than 50%, as a result of the value of b. Hence, the number of variables and constraints is slightly smaller than that computed by the above formula. Figure 5.5 shows the time and space requirements of Soplex to solve the *LP1* instances produced to generate G_p.

Concerning the *LP* instances that are produced to generate G_s, we compare (see Figure 5.6) the sizes (in terms of variables and constraints) of *LP2* (assuming that

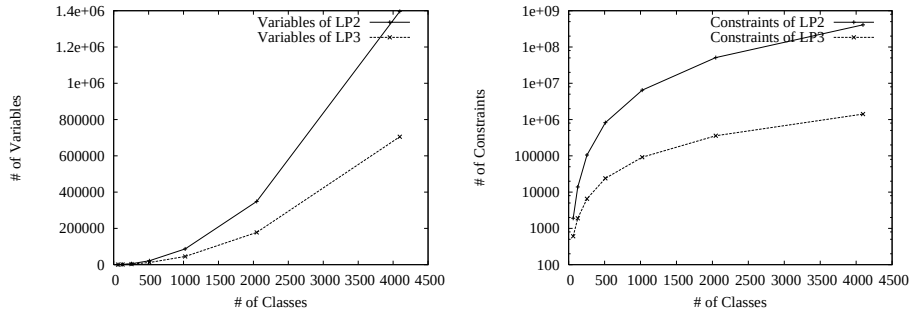

Figure 5.6: Number of Variables (left) / Constraints (right) of *LP2* and *LP3* Instances

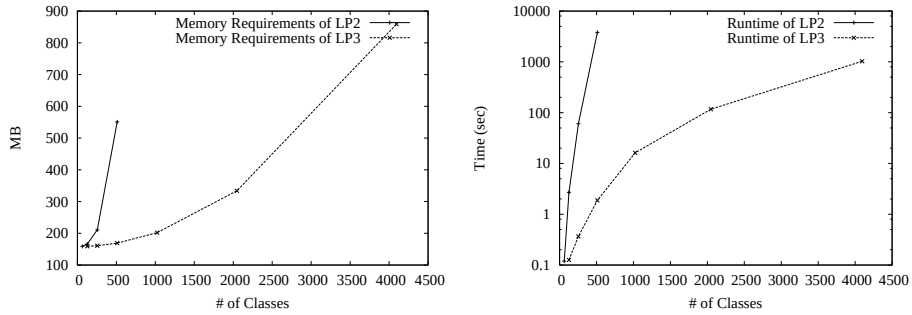

Figure 5.7: Memory Requirements (left) / Runtime (right) of *LP2* and *LP3* Instances

G_s is a DAG) and *LP3* instances (assuming that G_s is a tree). We also compare (see Figure 5.7) the space and time requirements of Soplex to solve them. In particular, we consider the degree sequences of the *transitive closures* of trees (whose 75% of nodes are leaves) and we reduce their generation to an *LP2* and to an *LP3* instance. As one can observe in Figure 5.6 (left), the number of variables of the *LP3* instance is smaller compared to *LP2* instance for the same given *Dout* and *Din*. This is due to the fact that in case of *LP3* instances, not all possible edges are candidates, i.e., only nodes of successive levels can be connected since G_s is considered to be a tree. Furthermore, the number of constraints corresponding to *LP2* is orders of magnitude bigger than *LP3*, as Figure 5.6 (right) shows ($y-$axis is plotted in *log scale*). This is due to the fact that *LP2* instances, except for degree constraints, additionally have transitivity constraints, the number of which is $\mathcal{O}(N^3)$, where N is the number of

classes. The big difference in the sizes of the *LP2* and *LP3* instances is reflected in the space requirements and runtime of Soplex to solve them (see Figure 5.7). In particular, Soplex crashes for $N = 1023$ (needs more than $1GB$ memory), while the runtime to solve *LP2* instances is orders of magnitudes bigger than *LP3* instances.

Chapter 6

Conclusion and Future Work

To the best of our knowledge, this is the first work that measured and analyzed the graph features of individual schemas coming from an adequately big *SW* corpus. The main conclusions drawn from our analysis (see Figure 6.1) are that the majority of schemas with a significant number of classes and properties approximate a *power-law* for class descendants and property total degree distributions while they also exhibit the *small world* phenomenon. These findings reveal emerging conceptual modeling practices employed by *SW* schema developers, namely: a) the existence in each schema of few central concepts that have been analyzed in detail (i.e., have many properties and subclasses) and are further connected with central concepts of other schemas, b) the trend to form clusters of connected classes, c) the fact that class subsumption hierarchies are unbalanced, with large branches and many leaves, d) the fact that most properties have as domain/range classes that are located highly at the class subsumption hierarchies, e) the importance of recursive/multiple arcs and f) the relatively large cyclic paths in the *undirected closure of the property* graph of most schemas in our corpus.

We also exploited these findings to generate synthetic *SW* schemas. Specifically, we proposed the generation of the *property* and the *subsumption* graph, given the schema size (in terms of classes and properties), as well as the *power-law* exponents for

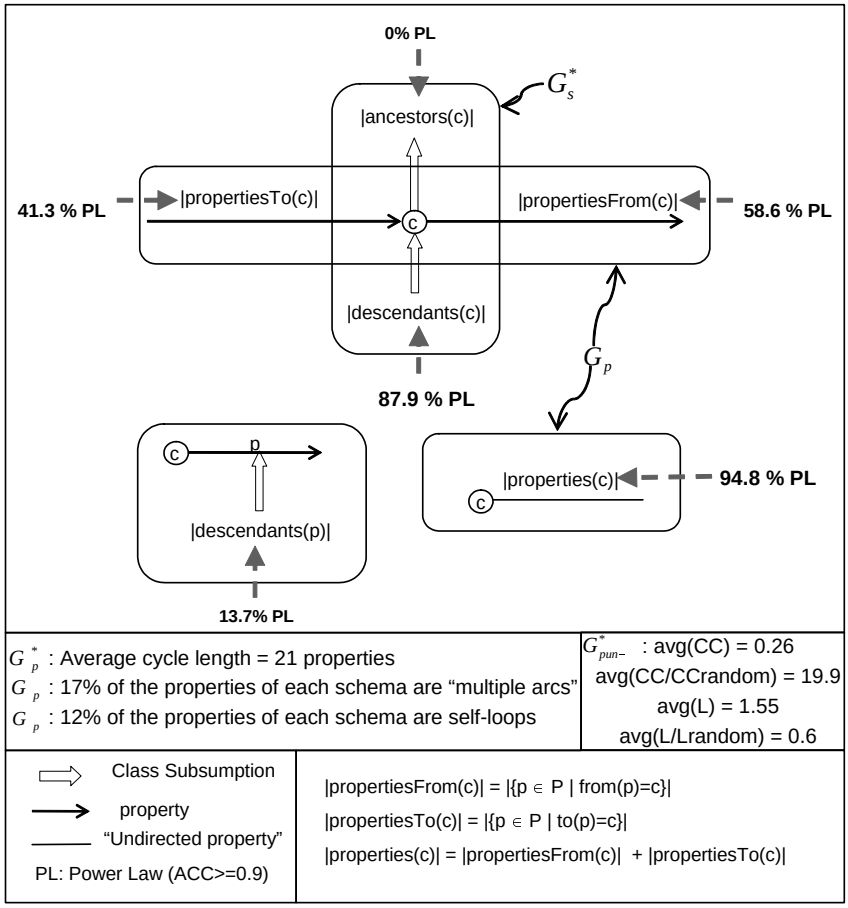

Figure 6.1: The main results of our analysis

the total-degree and the class descendants distributions. The former was constructed given its degree sequences, while the latter was constructed given the degree sequences of its *transitive closure*. Especially for the generation of the *subsumption* graph, we faced the problem of generating graphs given the out- and in-degree sequences of their *transitive closure*, problem not previously addressed in the bibliography. In this thesis we proposed its reduction to the *Linear Programming* problem. In order to combine the *property* and *subsumption* graph to compose the synthetic schema, the combinatoric feature reported in Section 2.2.6, i.e., that nodes with high out-degree in the G_p are located highly in the G_s, was proven very useful since it combines

characteristics of both graphs (*property* and *subsumption*) on the same set of nodes.

We plan to exploit the generator proposed in this thesis for synthetic *SW* data generation as a part of our *RDF* benchmarking efforts [49]. In particular, we aim at creating large datasets and testing the scalability of storage, query or update methods implemented by current *SW* tools. The refinement of our generator to also conform to other features, except for the degree distributions, such as the *small world* phenomenon, is also left as future work. Finally, it would be useful if we could devise a more efficient method (of complexity lower than $\mathcal{O}(N^5)$) to generate G_s when it is considered to be a DAG. To this end, it would be interesting to investigate if graph generation methods exploiting the ideas of *self-similarity* and *fractals* [39], could be modified to take as input the degree sequences, or at least the characteristic *power-law* exponent of their distribution.

Bibliography

[1] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, *"Search in Power-law Networks"*, Physical Review E **64** (2001), no. 46135.

[2] W. Aiello, F. Chung, and L. Lu, *"A random graph model for massive graphs"*, Procs of the 32nd Annual ACM Symposium on Theory of Computing (Toronto, Canada), 2000, pp. 171–180.

[3] W. Aiello, F. Chung, and L. Lu, *Handbook of massive data sets*, ch. "Random Evolution in Massive Graphs", Kluwer, 2002.

[4] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle, *"The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases"*, 2nd International Workshop on the Semantic Web, May 2001.

[5] A. Barabási and R. Albert, *"Emergence of scaling in random networks"*, Science **286** (1999), no. 509.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila, *The semantic web*, Scientific American (2001).

[7] Zhiqiang Bi, Christos Faloutsos, and Flip Korn, *"The "DGX" Distribution for Mining Massive, Skewed Data"*, Procs of the KDD 2001 (San Francisco, CA), August 2001.

[8] Joseph Blitzstein and Persi Diaconis, *"A Sequential Importance Sampling Algorithm For Generating Random Graphs With Prescribed Degrees"*, Annals of Applied Probabilty.

[9] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády, *"The degree sequence of a scale-free random graph process"*, Random Structures and Algorithms **18** (2001), no. 3, 279–290.

[10] K. H. Borgwardt, *"The Average number of pivot steps required by the Simplex-Method is polynomial"*, Mathematical Methods of Operations Research **26** (1982), 157–177.

[11] Dan Brickley and R. V. Guha, *"Resource Description Framework Schema (RDF/S) Specification 1.0"*.

[12] Deepayan Chakrabarti and Christos Faloutsos, *"Graph mining: Laws, Generators, and Algorithms"*, ACM Computing Surveys (CSUR) **38** (2006), no. 2.

[13] Fan Chung and Linyuan Lu, *"Connected Components in Random Graphs With Given Expected Degree Sequences"*, Annals of Combinatorics **6** (2002), 125–145.

[14] Fan Chung, Linyuan Lu, and Van Vu, *Eigenvalues of random power law graphs*, Annals of Combinatorics **7** (2003), 21–33.

[15] Berge Claude, *Graphs and hypergraphs*, North Holland Publishing Company, 1973.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed., ch. 26.4-26.5, MIT Press and McGraw-Hill, Cambridge, Mass, 2001.

[17] G. B. Dantzig, *"Activity Analysis of Production and Allocation"*, ch. Programming of Interdependent Activities, II, Mathematical Model, pp. 19–32, John Wiley and Sons Inc., New York, 1951.

[18] M. Dean and G. Schreiber, *"OWL Web Ontology Language Reference: W3C Recommendation",*W3C Recommendation, Febrouary 2004.

[19] L. Ding, T. Finin, and A. Joshi, *"Analyzing Social Networks on the Semantic Web"*, IEEE Intelligent Systems (2005).

[20] Li Ding and Tim Finin, *Gauging ontologies and schemas by numbers*, Procs of the 5th International Semantic Web Conference, ISWC'06 (Athens, GA, USA), 2006.

[21] E. A. Dinic, *"Algorithm for Solution of a Problem of Maximum Flow in a Network With Power Estimation"*, Soviet Math. Doklady **11** (1970), 1277–1280.

[22] Jack Edmonds and Richard M. Karp, *"Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems"*, Journal of the ACM **19** (1972), no. 2, 248–264.

[23] P. Erdös and T. Gallai, *"Graphs with prescribed degree of vertices"*, Mat. Lapok **11** (1960), 264–274.

[24] P. Erdös and A. Rényi, *"On the evolution of random graphs"*, Publ. Math. Inst. Hungar. Acad. Sci. **5** (1960), 17–61.

[25] M. Faloutsos, P. Faloutsos, and C. Faloutsos, *"On power law relationships of the internet topology"*, Procs of ACM SIGCOMM'99, 1999.

[26] R. Gil, R. García, and J. Delgado, *"Measuring the Semantic Web"*, Semantic Web Challenges for Knowledge Management: towards the Knowledge Web **1** (2004), 69–72.

[27] S.L. Hakimi, *"On the realizability of a set of integers as degrees of the vertices of a graph"*, SIAM Appl. Math. **10** (1962), 496–506.

[28] Harry Halpin, Valentin Robu, and Hana Shepherd, *The dynamics and semantics of collaborative tagging*, http://www.cwi.nl/themes/sen4/seminar/sen4seminar/bottom.html, 2006.

[29] V. Havel, *"A remark on the existence of finite graphs"*, Casopis Pest. Mat. **80** (1955), 477–480.

[30] Patrick Hayes, *RDF Semantics. W3C Recommendation*, February 2004.

[31] A. J. Hoffman and J. B. Kruskal, *Linear inequalities and related systems*, ch. Integral Bounding Points of Convex Polyedra, pp. 223–246, Princeton University Press, 1956.

[32] Bettina Hoser, Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme, *"Semantic Network Analysis of Ontologies"*, Procs of the 3rd European Semantic Web Conference (Budva, Montenegro), 2006.

[33] M. Hsu and T. E. Cheatham, *"Rule Execution in CPLEX: A Persistent Objectbase"*, Advances in Object-Oriented Database Systems: Proc. of the 2nd International Workshop (K. R. Dittrich, ed.), Springer, Berlin, Heidelberg, 1988, pp. 150–155.

[34] S. Jin and A. Bestavros, *"Small-World Internet Topologies. Possible Causes and Implicationson Scalability of End-System Multicast"*, Tech. Report BUCS-TR-2002-004, Boston University, Janouary 2002.

[35] L. G. Khachiyan, *"A Polynomial Algorithm for Linear Programming"*, Doklady Akad. Nauk USSR **244** (1979), no. 5, 1093–1096, Translated in *Soviet Math. Doklady*, 20, 191-194.

[36] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal, *"Stochastic models for the web graph"*, Procs of the 41st Annual Symposium on Foundations of Computer Science, FCOS 2000, 2000, pp. 57–65.

[37] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, *"Extracting large scale knowledge bases from the Web"*, Procs of 8th WWW Conference, April 1999.

[38] O. lassila and R. Swick, *"Resource Description Framework (RDF) Model and Syntax Specification",* W3C Recommendation, February 1999.

[39] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, and Christos Faloutsos, *"Realistic, mathematically tractable graph generation and evolution, using kroneckermultiplication"*, PKDD"05: 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, 2005, pp. 133–145.

[40] Lun Li, David Alderson, John C. Doyle, and Walter Willinger, *"Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications"*, Internet Math **2** (2005), no. 4, 431–523.

[41] A. Magkanaraki, S. Alexaki, V. Christophides, and D. Plexousakis, *"Benchmarking RDF schemata for the Semantic Web"*, Procs of the 1st International Semantic Web Conference, ISWC'02, 2002.

[42] M. Mihail and N. Visnoi, *"On generating graphs with prescribed degree sequences for complex network modeling applications"*, Procs of Approx. and Randomized Algorithms for Communication Networks (ARACNE), 2002.

[43] P. Mika, *"Ontologies Are Us: A Unified Model of Social Networks and Semantics"*, Procs of the 4th International Semantic Web Conference, 2005.

[44] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial optimization : Algorithms and complexity*, Dover Publications, Mineola, N.Y., 1998.

[45] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in c*, Cambridge University Press, 1992, (2nd edition).

[46] K. Shen and L. Wu, *"Folksonomy as a Complex Network"*, Cornell University Library, http://arxiv.org/PS_cache/cs/pdf/0509/0509072.pdf.

[47] L. Tari, C. Baral, and P. Dasgupta, *Understanding the global properties of functionally-related gene networks using the gene ontology*, Pacific Symposium on Biocomputing **10** (2005), 209–220.

[48] C. Tempich and R. Volz, *"Towards a benchmark for Semantic Web reasoners-an analysis of the DAML ontology library"*, Procs of The 2nd Int. Workshop on Evaluation of Ontology-based Tools, EON2003, 2003.

[49] Y. Theoharis, V. Christophides, and G. Karvounarakis, *"Benchmarking Database Representations of RDF/S Stores"*, Procs of the 4th International Semantic Web Conference, ISWC'05, 2005.

[50] Y. Tzitzikas and J. L. Hainaut, *"How to Tame a Very Large ER Diagram (using Link Analysis and Force-Directed Placement Algorithms)"*, Procs of 24th International Conference on Conceptual Modeling, ER'05, 2005.

[51] Arthur F. Veinott and George B. Dantzig, *"Integral Extreme Points"*, SIAM Review **10** (1968), no. 3, 371–372.

[52] Taowei David Wang, *"Gauging Ontologies and Schemas by Numbers"*, Procs of the 4th International EON Workshop, located at the 15th International World Wide Web Conference WWW 2006 (Edinburgh, United Kingdom), 2006.

[53] D.J. Watts and S.H. Strogatz, *"Graph structures in the Web"*, Collective dynamics of small world networks **393** (1998), 440–442.

[54] Roland Wunderling, *"Paralleler und Objektorientierter Simplex-Algorithmus"*, Ph.D. thesis TR 96-09, ZIB, 1996.

[55] J. Xu and M. Li, *"Assessment of RAPTOR"s Linear Programming Approach in CAFASP3"*, Proteins **53** (2003), 579–584.