AUTOMATED KNOWLEDGE SOURCE SELECTION AND SERVICE COMPOSITION

Patrick N. Bless
Department of Mechanical & Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois
Email: pbless@gmail.com

Diego Klabjan
Department of Mechanical & Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois
Email: klabjan@uiuc.edu

Soo Y. Chang
Department of Industrial Engineering
Pohang University of Science and Technology - Hyoja San 31,
Pohang, Republic of Korea
Email: syc@postech.ac.kr

ABSTRACT

We introduce a new combinatorial problem referred to as the component set identification problem, arising in the context of knowledge discovery, information integration, and knowledge source/service composition. The main motivation for studying this problem is the widespread proliferation of digital knowledge sources and services. Considering a granular knowledge domain consisting of a large number of individual bits and pieces of domain knowledge (properties) and a large number of knowledge sources and services that provide mappings between sets of properties, the objective of the component set identification problem is to select a minimum cost combination of knowledge sources that can provide a joint mapping from a given set of initially available properties (initial knowledge) to a set of initially unknown properties (target knowledge). We position the component set identification problem relative to other combinatorial problems and provide a classification scheme for the different variations of the problem. The problem is next modeled on a directed graph and analyzed in terms of its complexity. The directed graph representation is then augmented and transformed into a time-expanded network representation that is subsequently used to develop an exact solution procedure based on integer programming and branch-and-bound. We enhance the solver by developing preprocessing techniques. All findings are supported by computational experiments.

Keywords: Knowledge-based systems, information integration, knowledge source integration, service composition, integer programming.

1 Introduction

In many areas, such as natural science, engineering and medicine, researchers are accumulating an ever-growing amount of domain knowledge. Even though this is not a new phenomenon, the methodologies and tools to generate, store and utilize domain knowledge have been changed fundamentally by the digital revolution. Disciplines such as data mining, knowledge discovery, machine learning, and artificial intelligence are just a few examples of efforts that have been undertaken to successfully cope with and take full advantage of an ever-increasing amount of knowledge and information.

While the methodologies and tools of knowledge management are continuously evolving, the fundamental concepts of accessing knowledge and retrieving information remain unchanged. No matter whether information is being retrieved or requested from a living organism, a printed log table, a finite element model, a neural network, a web service, or a sensor attached to a satellite, the basic transaction required to obtain information or knowledge from any of these sources can be modeled as a function that converts a set of input arguments (provided information) into a set of output arguments (sought information). Any such function can be defined as a knowledge source. This very broad and rather loose definition of the notion of a knowledge source shows the wide reaching applicability and relevance of the presented combinatorial problems.

Digital knowledge sources have become a dominant concept for distributed computing and the use of communication networks as an aggregation point for heterogeneous knowledge sources and services is becoming more and more important. One example for such digital knowledge sources are web services. Web services are applications, which can be described, published, located and accessed from anywhere in the network, McIlraith (2001). They are fully characterized by the messages they send and receive, and by the operations or services they provide, Christensen *et al.* (2001). The aforementioned definition of a knowledge source includes the definition of web services. A lot of work has been done on the description and the syntax of web services, Staab (2003), and the declared future plan is to develop methodologies for automatic web services composition. Different approaches have been proposed to pursuit this objective, including web service choreography, Arkin *et al.* (2002), and web service orchestration, Peltz (2003). Web service choreography tracks the sequence of messages among multiple parties and multiple sources, and web service orchestration deals with the issue of how web services can interact and communicate with each other at the message level, including the business logic and execution order of the interactions.

Besides these relatively new and very specific efforts, an abundant amount of research has focused on the overarching problem of logical and physical integration of heterogeneous information. A large number of different approaches (integration paradigms) have been considered, including e.g. object orientated programming, reusable software modules, Cheng and Jeng (1997), ontology engineering, Chandrasekaran *et al.* (1999), schema matching, Domshlak *et al.* (2007), and middleware standards, Bakken (2001). Schema matching, for example, is the task of matching concepts describing the *meaning* of data from various data sources, Domshlak *et al.* (2007). An automated schema matcher tries to identify a mapping between the elements (attributes) of two or more data sources that maximizes the sum of the *degree of similarity* of the individual elements. Automated schema matching and the combinatorial problem presented herein are fundamentally different for several reasons. The underlying graph or network of the schema matching problem can be characterized by an n-by-n similarity matrix for all mappings between a set of attributes. The network structure of the problem presented herein is characterized by a infinite number of inhomogeneous transformation matrices mapping sets of input to sets of output attributes. Using appropriate data transformations, including a conversion of the notion of *degree of similarity* between data elements (or the lack of it) to a general cost, the basic version of the schema matching problem can be transformed to a special, highly restricted case of the problem presented here. In a paper recently presented by Saha *et al.* (2010) the notions of *alignment scores* and *schema covering* problems are introduced. Neither the schema covering nor schema matching problem considers precedence constraints and time expansion and thus they significantly differ from our work. The authors consider *ambiguity* of overlaps in the schema covering problem, a problem that is very specific to the data exchange and data transformation domains. The approaches presented in this paper are applicable to both problems. Despite these and many other efforts in the area of integration of heterogeneous knowledge sources, there has, to the best of our knowledge, so far been no fundamental discussion of the underlying combinatorial problems that have to be solved to automate the process of knowledge source/service selection and composition.

The objective of this work is to initiate this discussion and to provide an overview of the different combinatorial problems that have to be addressed to automate knowledge source integration and service composition. We provide a characterization scheme and a complexity analysis for these problems and develop suitable preprocessing and integer programming techniques for the component set identification problem. Our discussion is not tied to any particular integration paradigm and hence it will not become obsolete as new technologies for logical and physical information integration evolve.

The remainder of this paper is divided into five sections. In Section 2 we provide a description of the component set identification (CSI) problem, introduce a characterization scheme, and state the problem formally using a graph/network representation. In the same section we position the CSI problem relative to other combinatorial problems and present an overview of two other important aspects of automated knowledge integration and service composition, including the component set scheduling problem and the combined selection-and-scheduling problem. In Section 3 we elaborate on an example based on various potential knowledge sources from the world wide web. Section 4 presents a detailed discussion of the complexity of the different variations of the CSI problem based on the developed characterization scheme. In Section 5 the basic network representation of the CSI problem introduced in Section 2 is transformed into a time-expanded network representation. Based on this representation an integer programming formulation for the CSI problem is developed. In Section 6 we present several preprocessing techniques. Finally, Section 7 provides a summary of results obtained from computational experiments designed to assess the performance of the developed solution procedures and preprocessing techniques.

2 The Component Set Identification Problem

Any knowledge domain can be represented by a finite set $\mathbf{P} = \{p_j : j = 1,2...,m\}$ of knowledge bits or *properties* and a set of services or *knowledge sources* $\mathbf{K} = \{k_i : i = 1,2,...n\}$. The properties can be thought of as a granular representation of all relevant domain knowledge and the set of knowledge sources represents all available mappings from certain subsets of required input properties to certain subsets of output properties. Knowledge source k_i maps input properties $\mathbf{P}_i^{in} \subseteq \mathbf{P}$ to output properties $\mathbf{P}_i^{out} \subseteq \mathbf{P}$. Assuming

a domain representation with a complete set of property and knowledge source libraries, every information or knowledge request that can potentially be made in this domain, is characterized by two sets; a set $\mathbf{P}^I \subseteq \mathbf{P}$ of known properties (available knowledge) and a set $\mathbf{P}^T \subseteq \mathbf{P}$ of target properties (sought knowledge). Let $\tilde{p}^I = |\mathbf{P}^I|$ and $\tilde{p}^T = |\mathbf{P}^T|$. Assuming that each knowledge source has a non-negative cost associated with it, the objective of the CSI problem is to find the *minimum cost* combination of knowledge sources that can provide a joint mapping from $\mathbf{P}^I$ to $\mathbf{P}^T$.

Depending on the knowledge source costs, and depending on whether a single cost criterion or a weighted combinations of multiple cost criteria is used, minimum cost can correspond to fastest, most accurate, most affordable combination, or a combination of these criteria. In addition, besides the true cost, we can combine the cost other attributes such as performance metrics (availability, reliability) or quality of service. Ultimately, these various objectives could be addressed by means of goal programming, where we first optimize with respect to the most important objective criteria, next we optimize with respect to the second most important criteria in the neighborhood of the already obtained solution, and so forth. The formulation presented in Section 5 can easily be adopted to such a scheme.

The underlying decision problem can be modeled as a directed network consisting of a set $\mathbf{P}$ of m property nodes, a set $\mathbf{K}$ of n knowledge source nodes, and a set of arcs $\mathbf{A}$ connecting property nodes with knowledge source nodes. More specifically, there exists a directed arc (p_j, k_i) in the network if and only if property p_j is an input property of knowledge source k_i ($p_j \in \mathbf{P}_i^{in}$). Furthermore, there exist a directed arc (k_i, p_j) if and only if property p_j is an output property of knowledge source k_i ($p_j \in \mathbf{P}_i^{out}$). The CSI problem is formally stated as follows. Given the network $G(\mathbf{P}, \mathbf{K}, \mathbf{A}, c)$, consisting of the described property and knowledge source sets, a set of arcs $\mathbf{A}$, a set of root nodes $\mathbf{P}^I$, a set of terminals $\mathbf{P}^T$, and nonnegative knowledge source cost function c, the objective is to find the minimum cost acyclic subnetwork rooted at $\mathbf{P}^I$ that spans all target nodes $\mathbf{P}^T$ and meets all *input-property-induced precedence constraints*. A knowledge source node $k_i \in \mathbf{K}$ can only be part of a feasible subnetwork if all property nodes $p_j \in \mathbf{P}_i^{in}$ are connected with k_i via a directed path in the subnetwork starting at one of the root nodes. This is equivalent to requiring that

- if k_i is in the subnetwork, then all p_j, $j \in \mathbf{P}_i^{in}$ are in the subnetwork as well (if a knowledge source is used, all of its input properties have to be known), and
- if $p_j \in \mathbf{P} \backslash \mathbf{P}^I$ is in the subnetwork, then at least one k_i, $j \in \mathbf{P}_i^{out}$ is in the subnetwork (property has to be returned by at least one knowledge source), and
- all $p_j \in \mathbf{P}^T$ are in the subnetwork.

Finding the minimum cost acyclic subnetwork corresponds to finding the combination of knowledge sources that can generate the sought target properties $\mathbf{P}^T$ in the *most desirable* fashion. Figure 1 shows an example problem with three initially known properties (root nodes) and two target properties. The feasible solution marked in bold, provides paths from the initially known properties (P_B^I, P_C^I, P_F^I) to

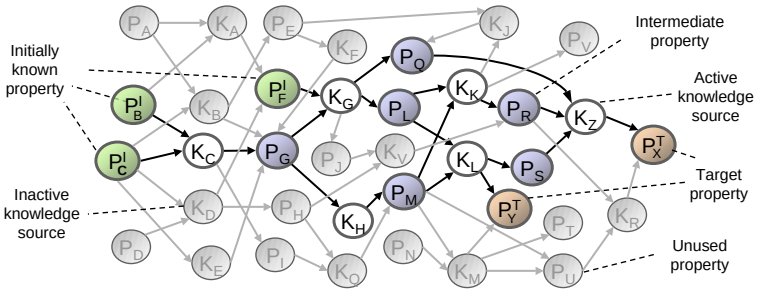

Figure 1. EXAMPLE CSI PROBLEM INSTANCE AND FEASIBLE SOLUTION.

both target properties (P_x^T and P_y^T). The subnetwork consists of six (*active*) knowledge sources (K_C, K_G, K_H, K_K, K_L, and K_Z) and six intermediate property nodes (P_G, P_Q, P_L, P_M, P_R, and P_S). A knowledge source is considered active if it is used within the solution subnetwork. The example shows both, knowledge sources that require a single input property and knowledge sources that require multiple input properties. Similarly, there are knowledge sources with a single output and knowledge sources with multiple outputs. The depicted solution satisfies all input property-induced precedence constraints, i.e. it is an acyclic subnetwork.

We stress that the requirements of the subnetwork being acyclic is critical since it guarantees that the selected knowledge sources are 'implementable', i.e. they satisfy the input-induced precedence constraints. We consider the example shown in Figure 2. At first

the depicted solution seems to be feasible since there exists a path from the initially known properties, P_1 and P_2, to the single target property P_9. However, knowledge source K_2 generates property P_5, a required input for knowledge source K_3 and knowledge source K_3 generates property P_4, a required input property for knowledge source K_2. This clearly results in a deadlock, causing the solution to be infeasible. This solution clearly has a cycle. By definition, a knowledge source can only be used in a solution if all its required input properties are available. The availability of these input properties is a function of the set of knowledge sources that have been used prior to a knowledge source. The scheduling aspect is discussed later in Section 2.2.

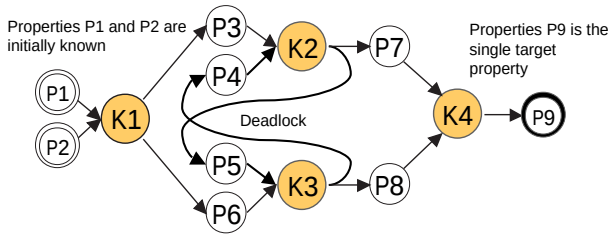

Figure 2. A DEADLOCK EXAMPLE.

The in- and out-degree p_i^{in}, p_i^{out} of knowledge source i is $|\mathbf{P}_i^{in}|$, $|\mathbf{P}_i^{out}|$, respectively. It corresponds to the in- and out-degree of k_i in $G(\mathbf{P},\mathbf{K},\mathbf{A},c)$. Similarly, the in- and out-degree of a property corresponds to the in- and out-degree of the corresponding node in $G(\mathbf{P},\mathbf{K},\mathbf{A},c)$.

Based on the number of initially known properties, the number of target properties, the number of inputs and outputs per knowledge source, and the cost associated with knowledge sources, we distinguish among several classes of CSI problem instances. The characterization scheme, Figure 3, that is used in the sequel consists of five fields, $\tilde{p}^I|\tilde{p}^T|\tilde{p}^{in}|\tilde{p}^{out}|\tilde{c}$, where $\tilde{p}^I$ represents the number of initially known properties, $\tilde{p}^T$ denotes the number of target properties, $\tilde{p}^{in}$ refers to the number of input properties per knowledge source, and $\tilde{p}^{out}$ represents the number of output properties per knowledge source. The last field, $\tilde{c}$, provides information about the knowledge source costs. In the first four fields we differentiate between two settings, 'M' for multiple and 'S' for single. For example, if $\tilde{p}^I = M$ and $\tilde{p}^T = S$ the problem instance has multiple initially known properties but only a single target property. If $\tilde{p}^{in} = S$ and $\tilde{p}^{out} = M$, the instance has a single input property for every knowledge source (p_i^{in}=1 for every knowledge source i) while the number of output properties per knowledge source can vary. For the last field characterizing the knowledge source costs, we differentiate between two settings, 'U' for unit ($c_i = 1$ for every $i \in \mathbf{K}$), and 'G' for general costs.

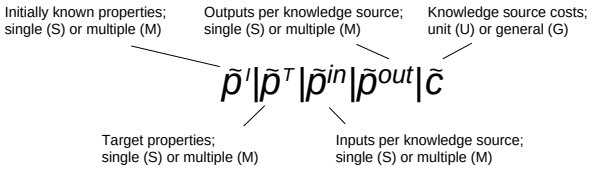

Figure 3. CHARACTERIZATION SCHEME FOR THE CSI PROBLEM.

2.1 The Two Aspects of Automated Knowledge Source Integration: Selection and Scheduling

Table 1 presents an overview of the different aspects of automated knowledge source selection and service composition. There are two main problems that have to be dealt with, the component set identification and the component set scheduling (CSS) problem. The CSI problem is only concerned with the identification of a suitable set of knowledge sources that provides a joint mapping from the set of initially known properties to the set of target properties. Once such a set $\mathbf{K}^s$ has been identified, the second problem, referred to as CSS, arises. Before focusing on the CSI problem for the rest of this paper, we will briefly discuss the CSS problem and also the combined CSI+CSS problem as summarized in the second and third row of Table 1.

Table 1. COMBINATORIAL PROBLEMS IN KNOWLEDGE SOURCE SELECTION AND SERVICE COMPOSITIONS.

Problem \ # of Processors	Single Processor	Multiple Processors	Infinite # of Processors
CSI Problem *"Select a set of knowledge sources"*	$\tilde{p}^{l}\|\tilde{p}^{T}\|\tilde{p}^{in}\|\tilde{p}^{out}\|\tilde{c}$ (see Table 2 for details) **NP-Hard**		
CSS Problem *"Schedule the set of knowledge sources"*	$P1\|prec\|C_{max}$ Polynomial	$Pm\|prec\|C_{max}$ **NP-Hard**	$P\infty\|prec\|C_{max}$ Polynomial
CSI + CSS Problem *(combined problem)* *"Select and schedule a set of knowledge sources. Solve problems simultaneously."*	Solving sequentially is optimal **NP-Hard**	Must be solved simultaneously **NP-Hard**	Must be solved simultaneously **NP-Hard**

2.2 The Component Set Scheduling Problem

CSS is stated as follows. Given a set $\mathbf{K}^s \subseteq \mathbf{K}$ of n knowledge sources with variable processing times d_i, and a set $\mathbf{M}$ of machines or processors, the objective is to find a schedule that minimizes the makespan[1], while satisfying all precedence constraints among the selected knowledge sources. An arc $i \rightarrow j$ in the precedence graph from knowledge source i to knowledge source j means that knowledge source j requires a subset of the output properties of knowledge source i as input. Since we require the underlying subnetwork of the solution to be acyclic, this precedence graph is acyclic. Assuming identical and parallel processors that can process exactly one knowledge source at a time, and considering the standard notation developed to denote scheduling problems, see e.g. Lawler *et al.* (1993), the CSS problem corresponds to the *makespan minimization problem on parallel machines with precedence constraints* ($Pm|prec|C_{max}$). The class of $Pm|...|C_{max}$ scheduling problems has been studied extensively and many variations of the problem have been proposed, including preemption, processors with different speeds, and due dates. The most basic variation of the problem, $Pm||C_{max}$, is equivalent to partitioning and is therefore strongly *NP*-hard, Garey and Johnson (1979). There are two polynomially solvable cases, $P1|prec|C_{max}$ and $P\infty|prec|C_{max}$. The first of these two problems is trivial because it is sufficient to keep the single processor busy at all times, resulting in a makespan equal to the sum of the processing times of all knowledge sources. The latter problem, $P\infty|prec|C_{max}$, features an unlimited number of processors and corresponds to a classical problem in the field of project planning. Methods such as the critical path method (CPM) and the project evaluation and review technique (PERT) can be used to solve the problem to optimality in polynomial time, French (1982).

2.3 The Combined Problem

Formulating the selection problem and the scheduling problem as a single combinatorial problem and solving both problems simultaneously, is the ultimate objective of automated knowledge source selection and service composition (see the last row of Table 1). The solution to the combined problem has to satisfy the objectives of both, the selection and the scheduling problem. In general, these two objectives, including minimizing the makespan (CSS) and minimizing the sum of the costs of all knowledge sources and properties in the selected subnetwork (CSI), impose a trade-off. Thus combining the two objectives into one single objective requires the definition of weights or utilities associated with the total cost and the makespan. Except for the one-processor case, the two subproblems have to be solved simultaneously to find the global optimal solution. In the single processor case the makespan is equal to the sum of the processing times of all selected knowledge sources and the two objectives can simply be combined by defining appropriate joint knowledge source costs. Given the complexity of the scheduling subproblem, it is obvious that the combined problem must be hard as well, no matter whether the two subproblems can be solved sequentially (the single processor case) or must be solve simultaneously (all other cases).

In order to design good heuristics or approximation algorithms for the combined problem, a thorough understanding of the two individual subproblems should first be obtained. As far as the scheduling subproblem is concerned a considerable amount of understanding has already been gained. The opposite is true for the selection problem, which has not been given any attention until now. Among the two subproblems, we focus on the CSI problem since its solution can be turned into a feasible solution to the combined problem.

[1] The makespan is the maximum completion time of all knowledge sources in $\mathbf{K}^s$. More specifically, it represents the total time it takes to execute the schedule of all $\mathbf{K}^s$ knowledge sources.

3 Example Problem

This section presents an example of the CSI problem as applied to a real world problem. The example is concerned with the development of an intelligent planning and decision support tool for relocation projects of private households and considers many facets of relocation, including location preferences, job search, the real estate market, and the financial implications.

A wide range of knowledge sources specializing on particular aspects of relocation are available and since the objective of this section is to demonstrate the structure of the problem formulation each type of a knowledge source will be briefly introduced. However, since we are dealing with domain-specific knowledge sources, the details of which are beyond the scope of this example, we make certain assumptions with respect to the costs and the input and output properties of the knowledge sources considered in this example.

3.1 Problem Statement

A couple would like to identify a suitable place to relocate to. They both want to be able to work full-time or part-time in their respective fields and afford a certain lifestyle (afford a home with certain specifications). In addition to their qualifications, the couple is further constrained by their desire to live in a country where the primary language is either English or Spanish. Furthermore, they want to live within 100 miles of a major body of water (ocean, lake).

Known Properties: Advanced geographic properties, qualifications, job category, job type properties, personal financial information properties, desired home characteristics (life-style properties).

Target Properties: Identified location, completed job application, identified future home, completed mortgage application.

Knowledge Sources and Domain specific CSI Network: Figure 4 provides an overview of the nine different types or *clusters* of knowledge sources considered in this example. All properties shown to the left of a knowledge source represent input properties while properties shown to the right are output properties. It is important to note that Figure 4 provides a generalized mapping for each cluster of knowledge sources, i.e. a specific knowledge source instance might only require a subset of the depicted input properties and only provide a subset of the output properties shown here.

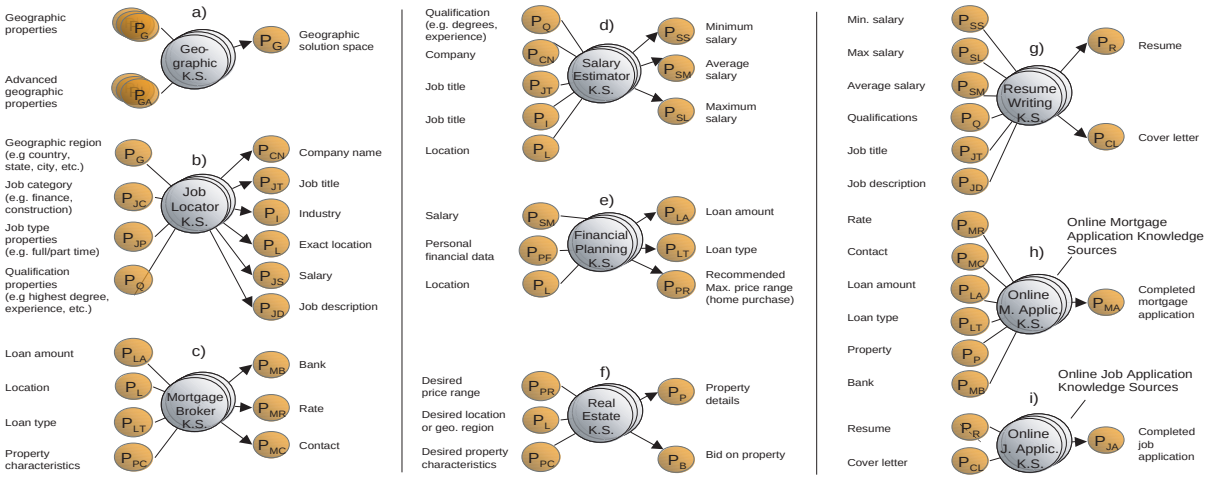

Figure 4. Knowledge Sources for the Relocation Domain

The knowledge sources of the first relevant cluster can best be described as geographic knowledge sources (Fig. 4-a). A geographic knowledge source takes in certain input parameters and/or constraints and returns a geographic location or a geographic area. The geographical resolution of the output varies and can range from a specific street address to a large geographic region that, as a whole, meets all input criteria. Besides the difference in output resolution, geographic knowledge sources also differ with respect to the array and level of sophistication of allowed input properties (constraints), search mechanism, the accuracy of the solution, and the costs associated with their use. Applications such as Google Earth [www.earth.google.com], MapQuest [www.mapquest.com], mapDex [www.mapdex.org], GeoTags [www.geotags.com] already have most of the information necessary to qualify as a geographical knowledge source.

The next cluster of knowledge sources are job locator knowledge sources, which allow users to search for job openings meeting certain input criteria (Fig. 4-b). There are significant differences in terms of the nature and the number of search properties and information required to perform the search. A very large number of internet-based job locator services are available. Most prominent examples include Monster [www.Monster.com] and the job search engines provided by Yahoo or AOL. There are also more specialized search engines serving only selected industries, geographical regions, degrees, etc.

Powerful mortgage broker knowledge sources are another critical ingredient when it comes to a thorough evaluation of a relocation project (Fig. 4-c). Given the location, a loan amount, and certain information about the property these knowledge sources provide information on suitable banks, rates, lending conditions, and contact information. The number of examples for readily available mortgage related knowledge sources is pretty much endless.

The relocation example also assumes availability of a variety of different salary estimation knowledge sources (Fig. 4-d). These services differ in the type and number of necessary input properties and vary greatly in terms of accuracy and cost. Many salary estimation knowledge sources are available and examples include Salary.com [www.salary.com] or SalaryExpert [www.Salary-Expert.com]. Job database providers such as Monster also offer salary estimation tools as separate services.

For the case study presented herein we also assume the existence of financial planning knowledge sources (Fig. 4-e). These tools can answer questions such as "how much can I afford to borrow?" or "what type of a loan is best suited for my financial situation?" Countless examples of financial planning knowledge sources that meets the above description can be found online.

Other important types of knowledge sources with respect to relocation are real estate locator knowledge sources (Fig. 4-f). An extremely wide range of real estate knowledge sources are available on the world wide web. Some of these knowledge sources can be used to automatically submit a bid on the desired property. Search parameters (input properties) vary considerably but usually include location, property characteristics and desired price range. A few examples are: Realtor.com [www.realtor.com], RealEstateFinder.com or [www.realestatefinder.com].

Furthermore we consider automated or semi-automated resume writing knowledge sources (Fig. 4-g). Similar to all previously discussed knowledge source clusters resume writing knowledge sources vary in the type and number of necessary input properties and have different price tags associated with them. Readily available examples of these type of knowledge sources include SeekingSuccess [www.seekingsuccess.com] and E-Resume [www.e-resume.us].

Depending on the provider and the loan amount, the services of mortgage broker internet portals also include the option of automatically initiating a mortgage request via an online application knowledge source (Fig. 4-h). Examples include [www.quickenloans.com; www.MyLoanSeeker.com].

Most large employers offer online application processes that allow applicants to upload a resume and a cover letter to the employer's candidate database. These tools can best be described as job application knowledge sources [2] and represent the last cluster of a knowledge source considered in the relocation example. Figure 4-i shows common property mappings for this type of knowledge sources.

Figure 5 depicts the resulting CSI network based on the mappings of the nine knowledge source clusters discussed above. Assuming the availability of 10 to 15 knowledge sources per cluster the resulting CSI network consists of approximately 120 knowledge sources and 35 properties. The leftmost column of properties shows all initially known properties and the four target properties (identified target location, completed job application, identified future home, and completed mortgage application) are shown in dark ovals. The network reveals some of the key intermediate properties that must be part of any feasible solution and indicates the complex nature of the precedence constraints that exist among the various knowledge sources.

4 Complexity of the CSI Problem

According to the aforementioned characterization scheme, there are several variations of the CSI problem and depending on the particular combination of network characteristics, the time complexity of the problem can vary considerably. This section provides a detailed discussion of the complexity of all different classes of CSI problem instances.

The number of different combinations of network characteristics that have to be formally analyzed can be reduced based on the following observation. Any CSI problem instance with more than one initially known property can be transformed into an equivalent instance with a single initially known property by introducing a dummy property node and a set of zero cost dummy knowledge source nodes as shown in Figure 6. Since this transformation can be performed in constant time and space, the complexity of $\text{CSI}^{\mathbf{S}|\cdot|\cdot|\cdot|\cdot}$ and $\text{CSI}^{\mathbf{M}|\cdot|\cdot|\cdot|\cdot}$ can be analyzed together.

We start the discussion of the complexity of the various CSI problem variations with $\text{CSI}^{\cdot|S|S|S|G}$, which represents the most restricted class of problem instances in terms of the network topography. Table 2 provides an overview of all classes of CSI problem instances.

[2] Examples include: Intel [jobs.intel.com/jobs], General Motors [www.gm.com/company/careers/job]

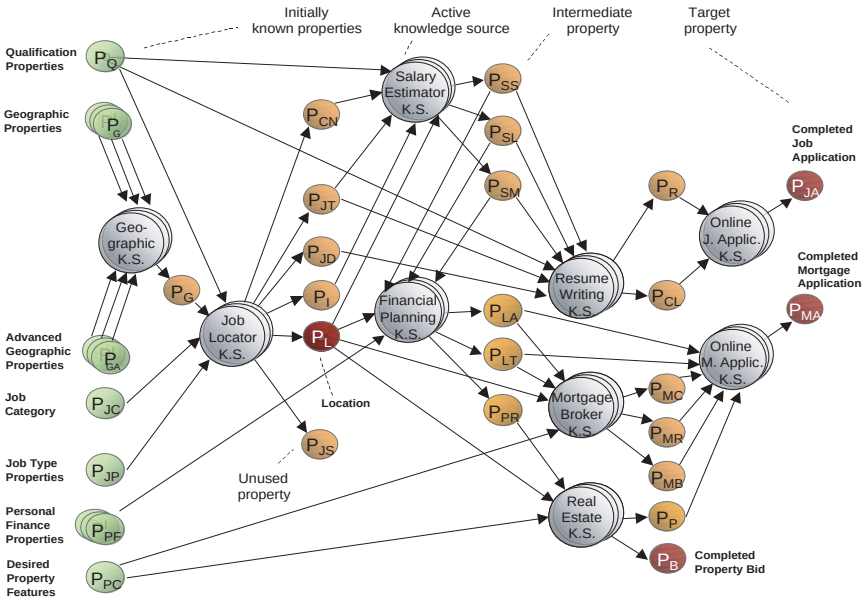

Figure 5. CSI Network for the Relocation Domain

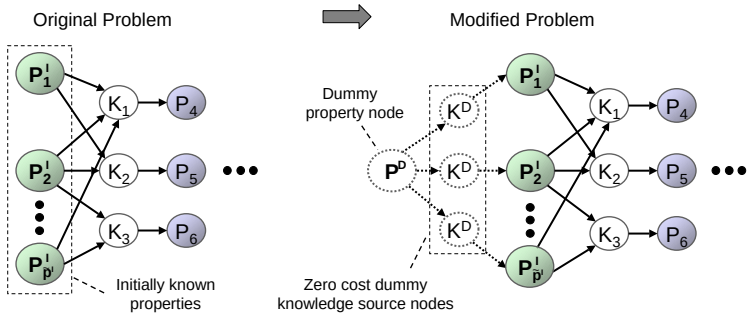

Figure 6. TRANSFORMATION FROM $\text{CSI}^{\mathbf{M}|.|.|.|.}$ to $\text{CSI}^{\mathbf{S}|.|.|.|.}$

$\text{CSI}^{\cdot|S|S|S|G}$ can be found in the first row of this table. Having a single initially known property ($\tilde{p}^I = 1$), a single target property ($\tilde{p}^T = 1$), and a set of knowledge source nodes with in- and out-degree equal to one ($\tilde{p}^{in} = \tilde{p}^{out} = 1$), the solution to the CSI problem is simply the shortest path from the initially known property to the target property of the problem instance. The shortest path problem is well understood and many efficient polynomial algorithms are available, see e.g. Ahuja *et al.* (1993).

Dropping the single output restriction for the knowledge source nodes leads to $\text{CSI}^{\cdot|S|S|M|G}$, a slightly more general class of problem instances (see second row of Table 2). Even though a knowledge source can now provide multiple outputs, we show next that there exists an optimal solution consisting of a simple path from the single initially known property to the single target property. Let p_j^* be the single target property of the problem and let $\mathbf{K}^T$ be the set of all knowledge sources that provide p_j^* as output. Note that all properties different from p_j^* that represent leaf nodes[3] in a given feasible solution to $\text{CSI}^{\cdot|S|S|M|G}$, can be removed without causing the solution to become infeasible. It is also obvious that knowledge sources that represent leaf nodes in a solution can be removed as well. This removal process of property and knowledge source leaf nodes can be performed repeatedly until no further simplification of the solution network is possible. Furthermore, it is clear that the problem can be considered *'solved'* as soon as p_j^* is returned by a knowledge source. From this it follows immediately that any solution that uses more than one $k_i \in \mathbf{K}^T$ must be suboptimal. Based on these two observations, and

[3] A property leaf node is a property node that has no outgoing arc in a given solution, i.e. a property that is not used by any of the selected knowledge source.

assuming that there are two partially disjoint paths in a solution to $\text{CSI}^{\cdot|S|S|M|G}$, the two paths have to merge somewhere ahead of reaching the selected $k_i \in \mathbf{K}^T$. If this is not the case, at least one of the two paths does not end at p_j^* and hence the corresponding leaf node can be removed. According to the problem definition, each knowledge source takes only a single input. Therefore, two paths can only merge at a property node. Having two paths merging at a property is clearly suboptimal because the disjoint portion of one of the two paths can be removed without affecting feasibility (see Figure 7). This completes the argument that the optimal solution to an instance of $\text{CSI}^{\cdot|S|S|M|G}$ can be found in polynomial time by simply solving the shortest s-t path problem from the initially known property to the target property.

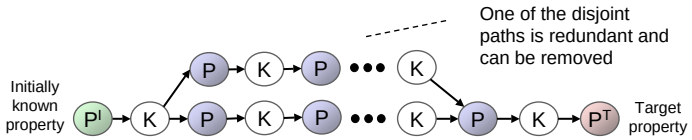

Figure 7. STRUCTURAL CONSIDERATIONS FOR $\text{CSI}^{\cdot|S|S|M|G}$.

The next case that is being considered is $\text{CSI}^{\cdot|M|S|S|U}$ (the third row in Table 2). According to the specifications for this class of CSI problem instances, all knowledge source nodes have unit cost and exactly one incoming and one outgoing arc, but many target properties. The decision version, denoted by $\text{CSI}_{\Pi}^{\cdot|M|S|S|U}$, is to find a subgraph $\hat{G}$ consisting of a subset $\mathbf{K}^s$ of knowledge source nodes and a subset $\mathbf{P}^s$ of property nodes that provide paths from elements in $\mathbf{P}^I$ to all elements of $\mathbf{P}^T$ and with $|\mathbf{K}^s| \leq B$, where B is a given integer.

Theorem 1. *The decision problem $\text{CSI}_{\Pi}^{\cdot|M|S|S|U}$ is NP-complete.*

Proof. $\text{CSI}_{\Pi}^{\cdot|M|S|S|U}$ is in *NP*. In order to prove the completeness a transformation from the Steiner arborescence problem is used. The Steiner arborescence problem is a generalization of the Steiner tree problem, which is shown to be *NP*-complete by a reduction from the vertex cover problem, Plesník (1983).

The decision version of the Steiner arborescence problem (SAP_{Π}) in graphs is stated as follows. Given a directed graph $G = (\mathbf{N}, \mathbf{A})$[4], a specified root $p_0 \in \mathbf{N}$, a set of terminals $\mathbf{T} \subseteq \mathbf{N}$, and an upper bound B, the question is whether G contains a directed acyclic subgraph with no more than B arcs, that connects root p_0 with all terminals in $\mathbf{T}$. An instance of the decision version of the Steiner arborescence problem can be transformed in polynomial time to an instance of $\text{CSI}_{\Pi}^{\cdot|M|S|S|U}$ as follows. The first step is to set $\mathbf{P} = \mathbf{N}$, $\mathbf{P}^I = \{p_0\}$, and $\mathbf{P}^T = \mathbf{T}$. In the second step every directed arc $a_d = (p_i, p_j) \in G$ is replaced by a knowledge source node k_d connected in the graph of $\text{CSI}_{\Pi}^{\cdot|M|S|S|U}$ by two directed arcs (p_i, k_d) and (k_d, p_j). Figure 8 illustrates this transformation.

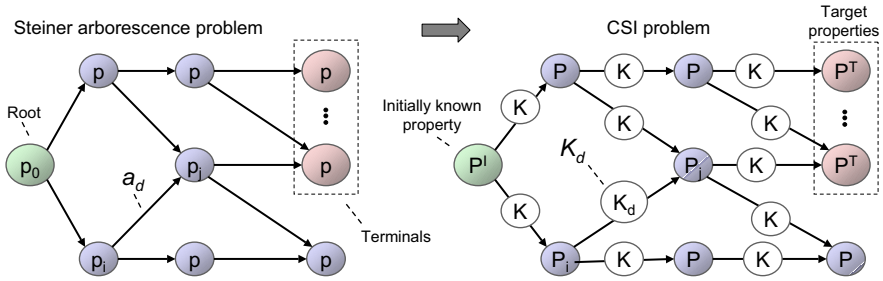

Figure 8. TRANSFORMATION FROM SAP_{Π} TO $\text{CSI}_{\Pi}^{\cdot|M|S|S|U}$.

The transformation neither adds nor deletes any path from G and it maintains a one-to-one correspondence between paths in the original and paths in the transformed problem. Note also that a subnetwork in G is acyclic if and only if the corresponding subnetwork in $\text{CSI}_{\Pi}^{\cdot|M|S|S|U}$ is acyclic. Furthermore, the number of arcs on a path in G corresponds to the number of unit cost knowledge source nodes

[4] G does not have to be acyclic.

on the corresponding path in the transformed CSI problem. Hence, it is obvious that a solution to SAP_Π of length B exists if and only if there is a solution consisting of B knowledge sources in the transformed problem. This completes the proof. □

The next three classes of CSI problem instances listed in rows 4-6 of Table 2 all have multiple target properties. Hence, these problem instances are at least as hard as $\text{CSI}^{\cdot|M|S|S|U}$ and no separate analysis of the complexity of these problem variations is needed. The next interesting class of problem instances is $\text{CSI}^{\cdot|S|M|S|U}$. Let us denote by $\text{CSI}_\Pi^{\cdot|S|M|S|U}$ the corresponding decision problem, which is stated in a very similar way as $\text{CSI}_\Pi^{\cdot|M|S|S|U}$ above.

Theorem 2. *$CSI_\Pi^{\cdot|S|M|S|U}$ is NP-complete.*

Proof. $\text{CSI}_\Pi^{\cdot|S|M|S|U}$ is clearly in *NP*. The completeness is shown using a transformation from $\text{CSI}_\Pi^{\cdot|M|S|S|U}$,which is *NP*-complete by Theorem 1.

An instance Q of $\text{CSI}_\Pi^{\cdot|M|S|S|U}$ can be transformed in linear time and space into an instance $\hat{Q}$ of $\text{CSI}_\Pi^{\cdot|S|M|S|U}$ as follows. Initially, $\hat{Q}$ is identical to Q. Next we add to $\hat{Q}$ a dummy knowledge source K^D, and a dummy property P^D. As illustrated in Figure 9, the new knowledge source is defined in such a way that it takes all target properties in Q as input and provides the dummy property P^D as output. The final step of the transformation is to define P^D as the only target property in $\hat{Q}$, while the set of initially known properties remains unchanged. Having a single target property and a knowledge source with multiple input properties (K^D), the transformed problem clearly meets the characterization scheme of $\text{CSI}_\Pi^{\cdot|S|M|S|U}$. Since the transformation does not modify the topography of the original instance, it follows that a solution to $\text{CSI}_\Pi^{\cdot|M|S|S|U}$ with B knowledge sources exists if and only if there is a solution to $\text{CSI}_\Pi^{\cdot|S|M|S|U}$ consisting of $B+1$ knowledge sources. This completes the proof. □

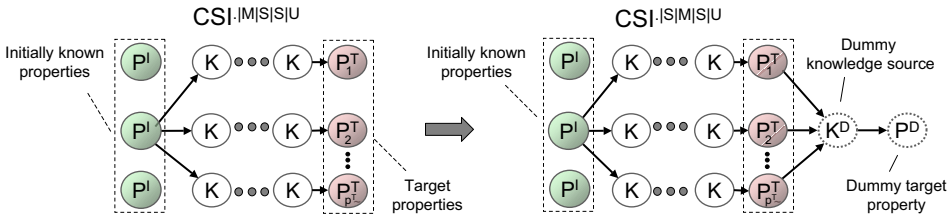

Figure 9. TRANSFORMATION FROM $\text{CSI}_\Pi^{\cdot|M|S|S|U}$ to $\text{CSI}_\Pi^{\cdot|S|M|S|U}$.

The last class of CSI problem instances that has to be considered is $\text{CSI}_\Pi^{\cdot|S|M|M|U}$. Since the knowledge sources in this class can feature multiple input properties, the class is 'dominated' by the complexity of $\text{CSI}_\Pi^{\cdot|S|M|S|U}$ and therefore it is at least as hard as $\text{CSI}_\Pi^{\cdot|S|M|S|U}$. This completes the discussion of the complexity of the CSI problem. Except for the two heavily restricted cases $\text{CSI}^{\cdot|S|S|S|G}$ and $\text{CSI}^{\cdot|S|S|M|G}$, which can be solved in polynomial time using existing shortest path algorithms, the CSI problem is NP-hard. Table 2 summarizes all findings presented in this section and provides a list of the complexity of all classes of CSI instances.

5 An Integer Programming Formulation

In this section an integer programming (IP) formulation for the CSI problem is presented. The problem is modeled as a discretized time-expanded network consisting of two types of nodes, nodes representing the knowledge sources and nodes representing the properties. An alternative IP formulation, which turned out to be on average less efficient, can be found in Bless (2004). As stated in Section 2.2, the complexity of the CSS problem is well understood and hence this paper focuses on the CSI problem.

5.1 Time Expansion

The problem of identifying a suitable combination of knowledge sources has an underlying temporal dimension. This temporal dimension is needed to account for the precedence constraints in the CSI problem. Hence, to solve the CSI problem the directed network from Section 2 has to be expanded.

One way of incorporating a temporal dimension is to develop a layered network representation of the directed graph that is capable of accounting for the time-related precedence constraints. In order to construct such a network representation multiple copies of all

Table 2. THE MAIN CLASSES OF THE CSI PROBLEM.

Row	Known Properties $\tilde{p}^I$	Target Properties $\tilde{p}^T$	Inputs per K.S. $\tilde{p}^{in}$	Outputs per K.S. $\tilde{p}^{out}$	K.S. Cost $\tilde{c}$	Com-plexity	(A) Equivalent to (B) Reduction from	Notes
(1)	S/M	S	S	S	G	Polynomial	Shortest Path (A)	
(2)		S	S	**M**	G	Polynomial	Shortest Path (A)	
(3)		**M**	S	S	U/G	NP-Hard	Steiner Network (B)	**Theorem 1**
(4)		**M**	**M**	S	U/G	NP-Hard	Steiner Network (B)	Dominated by (3)
(5)		**M**	S	**M**	U/G	NP-Hard	Steiner Network (B)	Dominated by (3)
(6)		**M**	**M**	**M**	U/G	NP-Hard	Steiner Network (B)	Dominated by (3)
(7)		S	**M**	S	U/G	NP-Hard	Steiner Network (B)	**Theorem 2**
(8)		S	**M**	**M**	U/G	NP-Hard	Steiner Network (B)	Dominated by (7)
S: Single M: Multiple U: Unit cost G: General cost								

knowledge source and property nodes have to be considered. Arcs that link the different copies across layers describe temporal linkages in the system. Hence, each node in the resulting network has a *time step* or *layer* associated with it and the resulting structure will be referred to as a time-expanded or layered network representation. The terms time step and layer will be used synonymously throughout the remainder of this paper. It is easy to see that any feasible solution to the CSI problem that has a time step in which either no knowledge source is active or no previously unknown property is newly generated has to be suboptimal. If there is a feasible solution with such an idle layer, the solution can be improved by simply skipping the respective layer. Therefore, in each time step we generate a new property and use a different knowledge source. Based on this observation a problem with n knowledge sources and m properties will have a maximum of $N = \min\{n,m\}$ time steps.

Formally, the time-expanded network is defined as follows. For each t, $1 \le t \le N$, for each knowledge source $k_i \in \mathbf{K}$, and for each property $p_j \in \mathbf{P}$ there are nodes $nk_{i,t}$ and $np_{j,t}$. We have an extra copy of property nodes denoted by $np_{j,N+1}$, $p_j \in \mathbf{P}$. Ordered pair $(np_{j,t}, nk_{i,t})$, $p_j \in P$, $k_i \in K$, $1 \le t \le N$ is an arc if and only if $p_j \in \mathbf{P}_i^{in}$. In addition, there are arcs $(nk_{i,t}, np_{j,t+1})$, $k_i \in K$, $p_j \in P$, $1 \le t \le N$ if and only if $p_j \in \mathbf{P}_i^{out}$. Nodes with the same time index t form a layer. Note that this network is acyclic and has $O(N(n+m))$ nodes. We discuss in Section 6 possible refinements of this network based on preprocessing. Figure 10 provides an example of the time-expanded network for the CSI problem.

In the following an IP formulation for the CSI problem is presented. The formulation explicitly selects knowledge sources, i.e. nodes in the time-expanded network.

5.2 The Formulation

We use two types of variables Z_{it} and X_{jt}, where

$$Z_{it} = \begin{cases} 1 & \text{if knowledge source } i \text{ is active in layer } t, \\ 0 & \text{otherwise,} \end{cases}$$

$$X_{jt} = \begin{cases} 1 & \text{if property } j \text{ becomes known in layer } t, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $Z_{it} = 1$ if node $nk_{i,t}$ is used and $X_{jt} = 1$ if node $np_{j,t}$ is used.

The objective function reads

$$\min \sum_{t=1}^{N} \sum_{i=1}^{n} c_i Z_{it}. \tag{1}$$

Next we give the constraints. To ensure that the required target properties are computed, we need

$$\sum_{t=1}^{N+1} X_{jt} \ge 1 \qquad j \in \mathbf{P}^T. \tag{2}$$

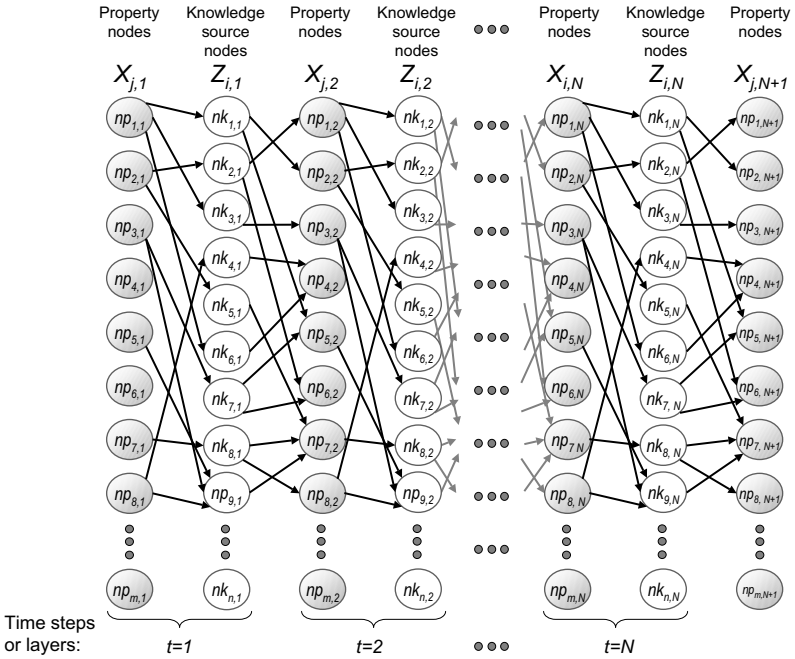

Figure 10. EXAMPLE OF A TIME-EXPANDED CSI NETWORK.

To model that the set of all properties in $\mathbf{P}^I$ are initially known, we require

$$X_{j1} = 1 \qquad j \in \mathbf{P}^I, \tag{3}$$

$$X_{j1} = 0 \qquad j \notin \mathbf{P}^I. \tag{4}$$

Let $\mathbf{L} = \{1,...,N\}$. To ensure that all input properties for knowledge source i are available in layer t^* if knowledge source i is active in in this layer, we need

$$\sum_{t=1}^{t^*} X_{jt} \geq Z_{it^*} \qquad t^* \in \mathbf{L}, \quad i \in \mathbf{K}, \quad j \in \mathbf{P}_i^{in}. \tag{5}$$

To guarantee consistency of the solution we need to enforce that property j is known in layer $t+1$, the layer following the determination of property j by knowledge source i in layer t. Hence we need,

$$X_{j(t+1)} \geq Z_{it} \qquad i \in \mathbf{K}, \quad t \in \mathbf{L}, \quad j \in \mathbf{P}_i^{out}. \tag{6}$$

The following constraints enforce that an $X_{j(t+1)}$ can only be nonzero if at least one knowledge source with property j as an output is nonzero at time step t. In absence of these constraints, $X_{j(t+1)}$ could be nonzero even if all Z_{it} are zero and this would obviously result in an infeasible solution. Hence, we require

$$\sum_{i:j\in\mathbf{P}_i^{out}} Z_{it} \geq X_{j(t+1)} \qquad t \in \mathbf{L}, \quad j \in \mathbf{P}. \tag{7}$$

Constraints (8) enforce that each knowledge source is used at most once throughout a particular solution and hence they prevent the generation of suboptimal solutions. Therefore, we add

$$\sum_{t=1}^{N} Z_{it} \leq 1 \qquad i \in \mathbf{K}. \tag{8}$$

Note that even without (8) the formulation is correct, however, these constraints are included since they were found to lead to an improved performance of the branch-and-bound algorithm.

By definition we have

$$Z_{it} \in \{0,1\} \qquad i \in \mathbf{P}, \quad j \in \mathbf{K}, \quad t \in \mathbf{L}, \tag{9}$$

$$X_{jt} \in \{0,1\} \qquad i \in \mathbf{P}, \quad j \in \mathbf{K}, \quad t \in \mathbf{L} \cup \{N+1\}. \tag{10}$$

The integer programming formulation consists of the objective function (1), constraints (2)–(8), and integrality requirements (9) and (10). The formulation has $N(n+m) = O(N^2)$ variables and $N\left(\sum_{i\in\mathbf{K}}\left(p_i^{in}+p_i^{out}\right)+m\right)+m+n+\tilde{p}^T = O(N^3)$ constraints.

6 Preprocessing of Problem Instances

The proposed formulation is solved by branch-and-bound. The development and application of effective preprocessing techniques has shown to lead to significant reductions in complexity and to considerable improvements of the performance of branch-and-bound on difficult combinatorial problems. For an in-depth discussion of preprocessing techniques for integer problems see e.g. Savelsbergh (1994) or Atamtürk *et al.* (2000).

To simplify instances of the CSI problem as much as possible before formulating and solving them by branch-and-bound, several preprocessing techniques have been developed. Each of these techniques exploits certain characteristics of the network topography of the CSI problem and most of them fall into one of the two main categories: reduction techniques and inclusion techniques. The former rules reduce the network, while the latter detect knowledge sources and properties that are in any feasible solution.

6.1 Reduction Techniques

The first type of preprocessing technique is aimed at reducing the actual problem size before solving the problem. Reduction techniques have successfully been applied to other network optimization problems with similar network topologies, including the travelling salesman problem, Osorio Lama and Pinto (2004), and the Steiner tree problem, Duin (2000). Next we give three rules that eliminate nodes or arcs.

1. *Property Exclusion Test (PET):*

   (a) A property that is not in $\mathbf{P}_I$ and has an in-degree of zero, is unobtainable and hence, can be removed from the network. Properties that meet these criteria will be referred to as *zero-in-degree* properties.
   (b) A property that does not belong to $\mathbf{P}_T$ and has an out-degree of zero, is of no value for any feasible solution and hence it can be removed from the network. These properties are referred to as *zero-out-degree* properties.
2. *Knowledge Source Exclusion Test (KET):* Knowledge sources that take one or more zero-in-degree properties as input can be removed from the network (*infeasible knowledge source*).
3. *Knowledge Source Domination Test (KDT):* A knowledge source may be dominated by another knowledge source or by a combination of knowledge sources. To determine whether knowledge source k_i is dominated by knowledge source k_g, the inputs, the outputs, and the cost of k_i and k_g have to be considered. If $\mathbf{P}_g^{in}$ is a subset of $\mathbf{P}_i^{in}$, $\mathbf{P}_g^{out}$ is a superset of $\mathbf{P}_i^{out}$, and $c_i \geq c_g$, then we say that knowledge source k_g dominates k_i and k_i can be removed from the network (*dominated knowledge source*). This rule can be generalized by considering dominance between two disjoint sets of knowledge sources. Identifying these more complex cases of dominance requires the development of suitable heuristics, Bless *et al.* (2006).

6.2 Inclusion Techniques

Here we discuss methods that are able to identify knowledge sources and properties that have to be present in every feasible solution.

1. *Knowledge Source Inclusion Test (KIT):* A knowledge source k_i that is the sole provider of a property $p_j \in \mathbf{P}^T$ has to be included in any solution. Knowledge sources that meet this criterion will be referred to as *required knowledge sources*.
2. *Property Inclusion Test (PIT):* This preprocessing rule is directly related to the previous rule. It is easy to see that all input properties of required knowledge sources can be classified as additional target properties and hence have to be part of any feasible solution.

Each time *KIT* or *PIT* apply to a knowledge source or a property, additional constraints have to be introduced into the IP formulation. For example, if *KIT* applies to knowledge source i, any feasible solution must satisfy

$$\sum_{t=1}^{N} Z_{it} = 1.$$

It is important to note that the developed preprocessing techniques can be applied iteratively until no further simplification of the problem is possible. This can be illustrated using an example that considers the elimination of a zero-in-degree property j based on *PET*. After removing property j from the network it is possible to eliminate all knowledge sources that take j as an input using *KET*. Eliminating these infeasible knowledge sources can result in new zero-in-degree properties, which can, once again, be removed using

PET. This procedure can be repeated until no further simplification is possible (preprocessing loop). Another important aspect of the presented preprocessing techniques is that the order in which these rules are applied within the preprocessing loop is irrelevant. Each preprocessing technique described in this section exploits a unique characteristic of the underlying graph and hence, the techniques do not cannibalize on each other or impact each other negatively.

7 Computational Experiments

This section provides a summary of computational experiments that have been performed to test the formulation and the preprocessing techniques. Unless specifically stated, all experiments presented in this section were performed on personal computers with INTEL® Pentium® 4 1.7GHz processors and with 256KB integrated level 2 cache, a bus speed of 100MHz, and 256MB ECC RDRAM system memory. CPLEX 8.0 (www.ilog.com) is used as the integer programming solver. Furthermore, the problem instances are generated randomly and unless stated otherwise all data points in all graphs are based on 100 runs (random problem instances).

7.1 Effectiveness of Preprocessing Techniques

To gain a complete understanding of the performance and effectiveness of preprocessing for different types of problem instances, a variety of computational experiments have been conducted. The objective of the first set of experiments was to identify, among all CSI network characteristics, those that have the most significant impact on the effectiveness of the various preprocessing techniques. To accomplish this, a 2^7 full factorial design of experiments was performed. The network characteristics that are considered include the number of knowledge sources, property saturation r^s defined as $\frac{m}{n}$, knowledge source in- and out-degree distribution ($\tilde{D}^{in}$ and $\tilde{D}^{out}$), knowledge source cost distribution, number of initially known properties, and number of target properties. Table 3 summarizes their ranges. Here $U^D(l,u)$ represents the uniform distribution of integers in the interval $[l,u]$, $\tilde{p}^{in}_{max}$ ($\tilde{p}^{out}_{max}$) is the maximum size of $\mathbf{P}^{in}_i$ ($\mathbf{P}^{out}_i$), and c^{max} is the largest knowledge source cost. The ranges were chosen based on the expected characteristics of CSI problem instances.

Table 3. DESIGN SUMMARY OF PREPROCESSING DESIGN OF EXPERIMENTS.

Factor	Symbol	Low Actual	High Actual
Number of knowledge sources	n	100	200
Property saturation ($r^s = m/n$)	r^s	0.35%	0.70%
Knowledge source in-degree distribution: $\tilde{D}^{in} = U^D(1, \tilde{p}^{in}_{max})$	$\tilde{p}^{in}_{max}$	3	4
Knowledge source out-degree distribution: $\tilde{D}^{out} = U^D(1, \tilde{p}^{out}_{max})$	$\tilde{p}^{out}_{max}$	3	4
Knowledge source cost distribution: $c_i = U^D(1, c^{max})$	c^{max}	100.00	500.00
Number of initially known properties in percentage of n	$\tilde{p}^I$	0.03%	0.10%
Number of target properties in percentage of n	$\tilde{p}^T$	0.03%	0.10%

The results obtained from these experiments indicate that the applicability of the different preprocessing rules is a function of the network characteristics of the preprocessed problem instance. The four network characteristics with the most significant effect on preprocessing are property saturation, knowledge source out-degree distribution, number of target properties, and knowledge sources in-degree distribution. Table 4 provides a list of the three most significant network characteristics for each type of preprocessing rule.

Table 4. NETWORK CHARACTERISTICS WITH THE MOST SIGNIFICANT EFFECTS ON DIFFERENT PREPROCESSING RULES.

	Ranking According to Significance of Estimated Effects on Performance		
Rule	1^{st} Place	2^{nd} Place	3^{rd} Place
PET	$r^s \uparrow$	$\tilde{p}^{out}_{max} \downarrow$	$\tilde{p}^{in}_{max} \downarrow$
PIT	$r^s \uparrow$	$\tilde{p}^{out}_{max} \downarrow$	interaction of r^s and $\tilde{p}^{out}_{max} \downarrow$
KET	$r^s \uparrow$	$\tilde{p}^I \uparrow$	$\tilde{p}^{out}_{max} \downarrow$
KIT	$r^s \uparrow$	$\tilde{p}^I \uparrow$	$\tilde{p}^{out}_{max} \downarrow$
KDT	$r^s \downarrow$	$n \downarrow$	interaction of r^s and $n \uparrow$
$\uparrow$: positive effect (more successful) $\downarrow$: negative effect (less successful)			

Property saturation has clearly the most decisive effect on preprocessing. This is expected since property saturation mimics the

notion of density in graphs. In networks with relatively few knowledge sources and many properties the level of connectivity among all the nodes (including knowledge sources and properties) is more likely to be low compared to the networks with many knowledge sources and few parameters. Certain preprocessing rules, such as PET are expected to eliminate more nodes in networks with a low level of connectivity. Since the performance of all preprocessing techniques is a function of the network characteristics, it is critical to include the notion of property saturation and to perform the extensive computational experiments shown in Figures 11 and 12. Considering all preprocessing rules knowledge source in- and out-degree distribution is the network characteristic with the second most significant effect on preprocessing.

For all subsequent experiments the following assumption is made. To mimic the expected knowledge source in- and out-degree distribution of CSI problem instances, Bless *et al.* (2005), a uniform distribution with a lower bound of 1 and an upper bound of 3 was imposed on both distributions.

Using these settings for the knowledge source in- and out-degree, the number of knowledge sources was held constant at 100, while the number of properties was gradually increased from 7 to 100, corresponding to a property saturation of 7% to 100%. The remaining network characteristics were $\tilde{p}^I = 3$, $\tilde{p}^T = 3$, and $c_i = U^D(1,500)$ for every $k_i \in \mathbf{K}$. Figure 11 summarizes the average performance of all preprocessing rules for a set of 10,000 randomly generated problem instances. The results indicate that for problems with a low property saturation ($10\% \leq r^s \leq 30\%$) the knowledge source domination test (*KDT*) is the most successful preprocessing technique. For high values of r^s, the property (*PET*) and knowledge source elimination rule (*KET*) is the most effective preprocessing rule. Depending on the level of property saturation, up to 50% of all knowledge source and property nodes are eliminated from the randomly generated problem instances. On average for $r^s = 25\%, 50\%, 75\%, 100\%$ the percentage of eliminated knowledge source (property) nodes is 4% ($\leq 1\%$), 2.5% (8%), 15% (22%), and 40% (47%), respectively.

While preprocessing techniques *KIT* and *PIT* result in additional constraints, *PET*, *KET*, and *KDT* translate directly into a reduction of the absolute problem size. As illustrated above, inclusion techniques do not share this effect but can be used to incorporate additional constraints to the IP formulation. Figure 12 shows the effect of this reduction in terms of the absolute number of knowledge source and property nodes that remain after preprocessing. Besides the immediate effect on the number of nodes in the network, preprocessing also reduces the maximum number of layers, N, that have to be considered in the IP formulation. This secondary effect has generally an even larger impact on the complexity of the resulting integer program.

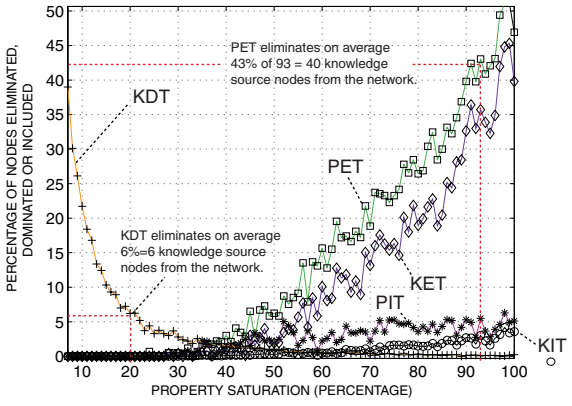

Figure 11. EFFECTIVENESS OF PREPROCESSING RULES AT DIFFERENT LEVELS OF PROPERTY SATURATION.

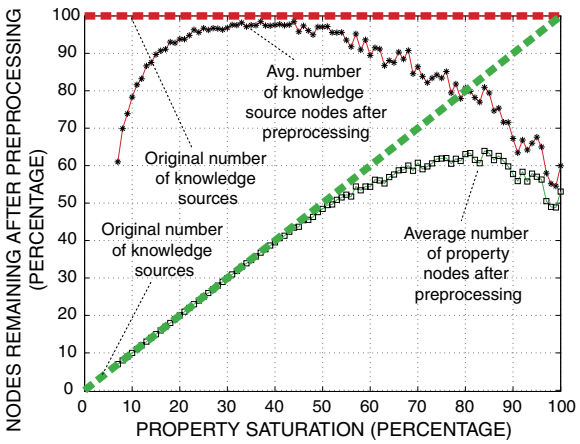

Figure 12. ABSOLUTE PROBLEM SIZE AFTER CSI-SPECIFIC PREPROCESSING.

To test the performance of the preprocessing rules in practice and to validate the results presented above, randomly generated test instances with varying property saturation were solved with and without preprocessing. We compare default CPLEX, i.e. general purpose integer programming preprocessing, with our CSI-specific preprocessing rules. The number of knowledge sources was held constant at 30 and the property saturation was gradually increased from 23.3% to 100%. All other network characteristics were the same as in the previous experiments. A time limit of one hour was imposed for each problem instance. The label "F1 w/o PP" denotes the option without CSI-specific preprocessing and "F1 w. PP" corresponds to the case with enabled our problem specific preprocessing.

Figures 13-18 provide an overview of the results obtained from these experiments. Figures 13 and 14 show the number of rows and columns in the IP. The data indicate that the application of the developed preprocessing techniques reduces the number of rows and columns at all levels of property saturation. The most significant reductions are realized at high property saturations. The main reason why the maximum complexity in terms of number of rows and column occurs at around 70-80% property saturation and not between 30-40%, where according to Figure 11 preprocessing is least successful, is the fact that the maximum number of layers peaks at a property saturation of about 70-80% as shown in Figure 12.

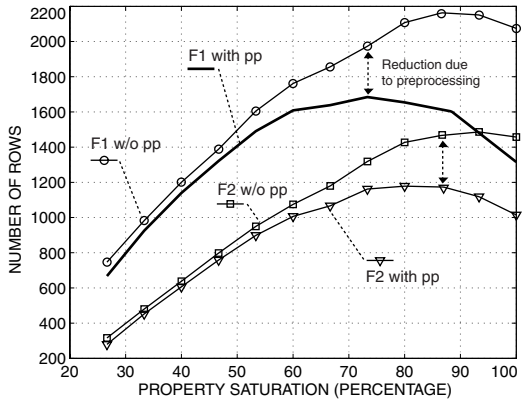

Figure 13. NUMBER OF ROWS (30 KNOWLEDGE SOURCES).

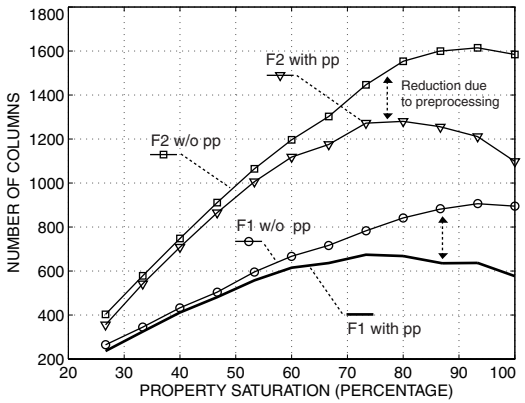

Figure 14. NUMBER OF COLUMNS (30 KNOWLEDGE SOURCES).

Figure 15 shows the average optimality gap at the root node. We define the gap as $(z^{IP} - z^{LP})/z^{IP}$, where z^{IP} is the optimal value and z^{LP} is the value of the LP relaxation. We observed that CPLEX was always able to find a feasible solution at the root node. This is not surprising since feasible solutions with many knowledge sources are easily obtainable. In the preprocessed case, the gap peaks at a property saturation of approximately 50%. Figures 16 and 17 provide plots of the average CPU time and the average number of evaluated branch-and-bound nodes. Figure 16 also indicates how often CPLEX stopped after one hour without finding an optimal solution. To ensure a fair comparison of the CPU times with and without preprocessing at different levels of property saturation, instances that could not be solved to optimality within the one hour time limit were not considered for the determination of the average CPU time and the average number of evaluated branch-and-bound nodes visited shown in Figures 16 and 17. The time reduction that can be achieved by using the developed preprocessing rules is significant and becomes progressively larger with increasing property saturation. At a property saturation of 70% the average reduction in CPU time due to preprocessing is 48.2%. The required CPU time peaks at a property saturation of 60-80% when used in combination with the CSI-specific preprocessing rules. This is consistent with earlier observations.

7.2 Limitations

The peak in problem complexity at a property saturation of 60-80% observed for relatively small problem dimensions becomes more pronounced when solving larger problems instances. This is shown in Figure 18, which illustrates the average time required to solve the root node of the branch-and-bound tree for problem instances with 100 knowledge sources and varying property saturation.

Having identified the range of property saturation that results in the most complex problem instances, the next step was to run a set of experiments with a fixed property saturation and an increasing number of knowledge sources. The objective is to determine the maximum problem size that can reliably be solved to optimality. These experiments were performed on a personal computer with a Pentium® 4 3.0GHz processor with 512KB integrated level 2 cache, a bus speed of 800MHz, and 1024MB DDR SDRAM system memory.

We use the same network characteristics as before to generate the random problem instances with a property saturation of 70% and an increasing number of knowledge sources. All available preprocessing rules are turned on, including the inclusion preprocessing techniques described in Section 6.2. These inclusion techniques effectively represent cuts that are being added to the IP formulation. To further reduce the average optimality gap at the root node the idea of adding cuts should be explored further. Previous work on variants of the Steiner Tree problem has shown that adding cuts and applying cutting plane algorithms can help to further reduce the

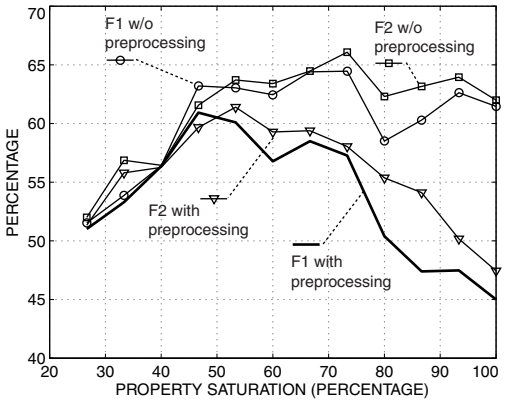

Figure 15. AVERAGE OPTIMALITY GAP AT THE ROOT NODE (30 KNOWLEDGE SOURCES).

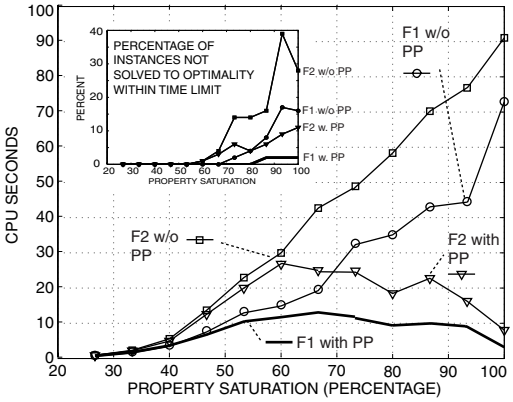

Figure 16. OVERALL CPU TIMES (30 KNOWLEDGE SOURCES).

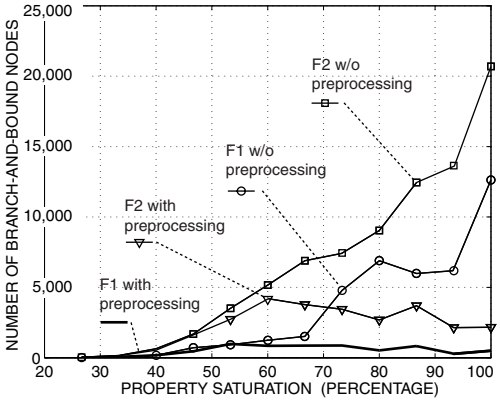

Figure 17. NUMBER OF EVALUATED BRANCH-AND-BOUND NODES (30 KNOWLEDGE SOURCES).

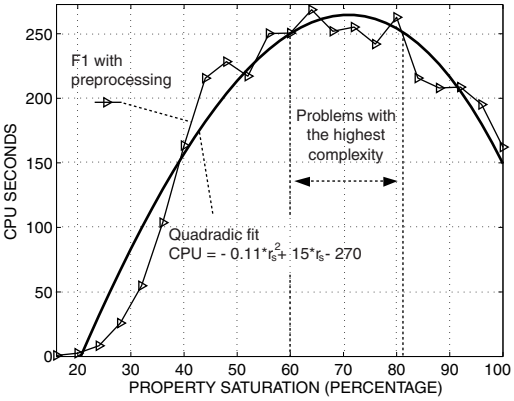

Figure 18. ROOT NODE LP RELAXATION WITH 100 KNOWLEDGE SOURCES.

optimality gap, Grotchel *et al.* (1996). Figure 19 shows the gap at the root node over all levels of knowledge sources. It is clear that the gap increases with the number of knowledge sources. Figure 20 shows the average CPU time together with an exponential function fit. Figure 21 shows the relative frequency with which the formulation was unable to find the optimal solution within the 1 hour time limit.

8 Summary

This paper addressed the combinatorial aspects of automated knowledge source selection and service composition. It mainly focuses on the component set identification problem, a combinatorial problem arising in the context of knowledge discovery, information integration, and knowledge source/service composition. Using a transformation from the Steiner arborescence problem, it was shown that the CSI problem is *NP*-hard. We give an IP formulation for solving the problem. It was found that problem-specific preprocessing can lead to considerable reductions in problem complexity, amounting to more than 50% reduction in CPU time even for small problem instances. Property saturation, knowledge source in- and out-degree distribution, and the number of initially known properties were identified as the network characteristics with the most significant effect on the success of preprocessing. Assuming a knowledge source in- and out-degree distribution of $\tilde{D}^{in} = U^D(1,3)$ and $\tilde{D}^{out} = U^D(1,3)$, it was found that the property saturation of 60-80% leads to the most complex problem instances. Unfortunately the formulation yields large optimality gaps at the root node. Problem instances with up to 35 knowledge sources and a property saturation of 70% can reliably be solved to optimality within one hour of CPU time in over 97% of all cases. Slightly larger instances with more than 40-50 knowledge sources on the other hand can only be solved to optimality in about 50% of all cases within the same time limit. The exponential increase in execution time suggests that problem instances beyond

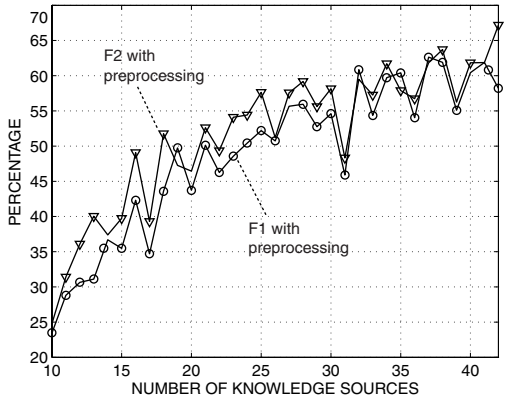

Figure 19. AVERAGE OPTIMALITY GAP AT THE ROOT NODE.

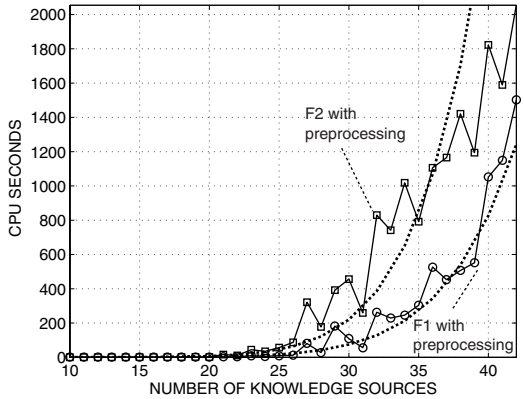

Figure 20. EXPONENTIAL FIT FOR AVERAGE CPU TIMES.

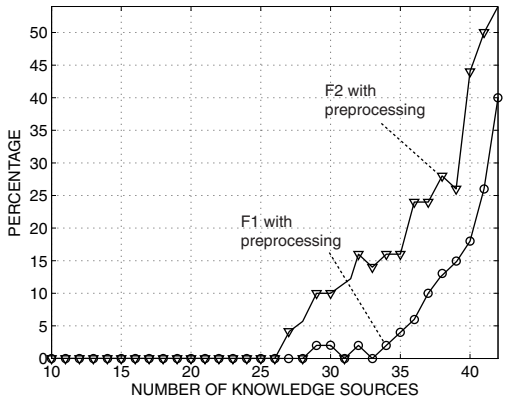

Figure 21. AVERAGE PERCENTAGE OF PROBLEMS THAT CAN NOT BE SOLVED TO OPTIMALITY WITHIN 1 HOUR TIME LIMIT.

60 knowledge sources and a property saturation of 70% are generally impossible to solve within a reasonable time frame using the IP formulation.

REFERENCES

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows*. Prentice Hall, Upper Saddle River.

Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Strublen, S., Takacsi-Nagy, P., Trickovic, I., and Zimek, S. (2002). Web service choreography interface (WSCI) 1.0. Technical report, W3C. http://www.w3.org/TR/WSCI.

Atamtürk, A., Nemhauser, G., and Savelsbergh, M. (2000). Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, **121**, 40–55.

Bakken, D. (2001). *Middleware*. Kluwer Academic Press, Norwell, MA.

Bless, P. (2004). *Automated knowledge source selection and service composition in manufacturing*. Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

Bless, P., Kapoor, S., DeVor, R., and Klabjan, D. (2005). An algorithmic strategy for automated generation of multi-component software tools for virtual manufacturing. *ASME Journal of Manufacturing Science and Engineering*, **127**, 866–874.

Bless, P., Klabjan, D., and Chang, S. (2006). Heuristic for automated knowledge source integration and service composition. *Computers and Operations Research*. To appear.

Chandrasekaran, B., Josepson, J., and Benjamins, V. (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems*, **14**, 20–26.

Cheng, H. C. and Jeng, J.-J. (1997). Reusing analogous components. *Knowledge and Data Engineering*, **9**, 341–349.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web services description language 1.1 (WSDL). Technical report, W3C. http://www.w3.org/TR/WSDL.

Domshlak, C., Gal, A., and Roitman, H. (2007). Rank aggregation for automatic schema matching. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, **19**(4), 538–553.

Duin, C. (2000). Preprocessing the Steiner problem in graphs. In D. Du, J. Smith, and J. Rubinstein, editors, *Advances in Steiner Trees*, volume 6 of *Combinatorial Optimization*, pages 175–233. Kluwer Academic Publishing, Norwell, MA.

French, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop*. Horwood, Chichester.

Garey, M. and Johnson, D. (1979). *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, NY.

Grotchel, M., Martin, A., and Weismanel, R. (1996). Packing Steiner trees: a cutting plane algorithm and computation. *Mathematical Programming*, **72**(4), 125–145.

Lawler, E., Lenstra, J., Rinnooy-Kan, A., and Shmoys, D. (1993). Sequencing and scheduling: Algorithms and complexity. In S. Graves, A. Rinnooy-Kan, and P. Zipkin, editors, *Logistics of Production and Inventory*, pages 445–522. North-Holland, Amsterdam.

McIlraith, S. (2001). Semantic web services. *IEEE Intelligent Systems*, **16**, 46–53.

Osorio Lama, M. and Pinto, D. (2004). A preprocessing that combines heuristic and surrogate constraint analysis to fix variables in TSP. In R. Monroy, G. Arroyo-Figueroa, L. Sucar, and J. Azuela, editors, *MICAI-Advances in Artificial Intelligence*, volume 2972 of *Lecture Notes in Computer Science*, pages 727–734. Springer, New York, NY.

Peltz, C. (2003). Web services orchestration - A review of emerging technologies, tools, and standards. Technical report, Hewlett Packard.

Plesník, J. (1983). *Graph Theory*. Veda, Bratislava.

Saha, B., Stanoi, I., and Clarkson, K. (2010). Schema covering: a step towards enabling reuse in information integration. In *2010 IEEE 26th International Conference on Data Engineering (ICDE)*, pages 285 –296.

Savelsbergh, M. W. P. (1994). Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, **6**, 445–454.

Staab, S. (2003). Web services: Been there, done that? *IEEE Intelligent Systems*, **18**, 72–85.