Universidade Federal de Alagoas
Instituto de Computação

Pós-Graduação em Modelagem Computacional de Conhecimento

Descoberta e Composição de Serviços Web Semânticos Através de Algoritmo Genético Baseado em Tipos Abstratos de Dados

Elvys Alves Soares

Dissertação de Mestrado

Maceió
13 de novembro de 2009

Universidade Federal de Alagoas
Instituto de Computação

Elvys Alves Soares

Descoberta e Composição de Serviços Web Semânticos Através de Algoritmo Genético Baseado em Tipos Abstratos de Dados

Trabalho apresentado ao Programa de Pós-Graduação em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para obtenção do grau de Mestre em Modelagem Computacional de Conhecimento.

Orientador: *Prof. Dr. Evandro de Barros Costa*
Co-orientador: *Profa. Dra. Roberta Vilhena Vieira Lopes*

Maceió
13 de novembro de 2009

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária Responsável: Helena Cristina Pimentel do Vale

S676d Soares, Elvys Alves.
Descoberta e composição de serviços Web semânticos através de algoritmo genético baseado em tipos abstratos de dados / Elvys Alves Soares, 2009.
xii, 65 f. : il.

Orientador: Evandro de Barros Costa.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2009.

Bibliografia: f. 62-65.

1. Algoritmo genético. 2. Web semântica. 3. Serviço na web. I. Título.

CDU: 004.021

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Modelagem Computacional de Conhecimento pelo Programa Multidisciplinar de Pós-Graduação em Modelagem Computacional de Conhecimento, da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina:

Prof. Dr. Evandro de Barros Costa
UFAL – Instituto de Computação
Orientador

Profa. Dra. Roberta Vilhena Vieira Lopes
UFAL – Instituto de Computação
Co-orientadora

Prof. Dr. Henrique Pacca Loureiro Luna
UFAL – Instituto de Computação
Examinador

Profa. Dra. Leliane Nunes de Barros
USP – Instituto de Matemática e Estatística
Examinadora

Maceió, novembro de 2009.

À minha mãezinha, cujo amor infinito e verdadeiro vence
as mais elaboradas barreiras e me faz, finalmente,
entender o que é ser sábio.

Agradecimentos

Agradeço aos meus orientadores, Prof. Evandro pela paciência e Profa. Roberta pela confiança, sem os quais dificilmente eu conseguiria concluir este trabalho. Ainda, não poderia deixar de prestar meus agradecimentos aos colegas de trabalho pelas injeções frequentes de ânimo e boa companhia, da qual certamente sentirei falta. No IF-AL, agradeço aos professores, valiosos e esperançosos colegas de trabalho, e aos alunos, cujo interesse sem fim uso como combustível para crescer mais e mais. E finalmente à FAPEAL, pelo apoio financeiro em parte deste trabalho.

“Fazia cinqüenta anos. Cinqüenta anos para se estabelecerem e darem início à Fundação Enciclopédica Número Um, tornando-a uma unidade de trabalho sem obstáculos. Cinqüenta anos colhendo material. Cinqüenta anos de preparação.

Estava tudo pronto. Dentro de cinco anos viria à luz a publicação do primeiro volume do trabalho mais gigantesco até então concebido na Galáxia. Depois, com intervalos de dez anos, regularmente, volume após volume. Juntamente com eles apareciam suplementos, artigos especiais sobre assuntos de interesse mais atual...”

— ISAAC ASIMOV (Os Enciclopédicos, Fundação, 1951)

Resumo

A Web Semântica é uma ampliação da web atual onde a disposição da informação viabiliza a cooperação entre homens e, sobretudo, entre máquinas. O surgimento de padrões web que expressam significado compartilhado possibilitam a construção de aplicações que resolvem problemas de integração, colaboração e automação já identificados pela comunidade científica e mercado consumidor de tecnologias.

A utilização de Serviços Web trouxe grandes ganhos neste sentido, e sua anotação em termos semânticos, tornando-os Serviços Web Semânticos, viabiliza a proposta da Web Semântica. Diversas tecnologias viabilizam a construção de tais elementos e sua conseqüente utilização como blocos básicos do desenvolvimento de aplicações cujo escopo é embarcado na web. Assim, dado o rápido crescimento da quantidade de serviços, tornam-se necessárias abordagens que resolvam de forma efetiva, com garantias de qualidade e tempo de resposta aceitável, a integração e posterior utilização destes.

Este trabalho propõe a modelagem de uma solução de software para o problema da Descoberta e Composição de Serviços Web Semânticos através do uso do Algoritmo Genético Baseado em Tipos Abstratos de Dados. Também é proposta uma implementação utilizando OWL, OWL-S e a OWL-S API. São apresentadas a definição formal do problema, as expectativas da comunidade científica quanto às soluções elaboradas e os resultados obtidos com respeito à viabilidade da proposta.

Palavras-chave: Serviços Web Semânticos, Algoritmo Genético, Descoberta e Composição

Abstract

The Semantic Web is an extension of the current Web, where the availability of information is expected to enable the cooperation between man and, above all, machines. The creation of standards which express shared meaning enable the construction of applications to solve integration, collaboration and automation problems which were already been identified by scientific community and technology consumers.

The use of Web Services has brought several advances in this sense, and their annotation in semantic terms, transforming them into Semantic Web Services, enables the Semantic Web intent. Several technologies also enable the creation of such elements and their inherent use as basic blocks of application development whose scope is embedded on Web. This way, due to the fast growing of the number of services, some approaches to effectively solve the problem of services integration and use become necessary.

This work proposes a modeling of a software solution to the discovery and composition of Semantic Web Services problem through the use of a genetic algorithm based on abstract data types. It is also proposed a tool implementation using OWL, OWL-S and OWL-S API languages and frameworks as well as the formal problem definition along with the scientific community expectations to the given solution.

Keywords: Semantic Web Services, Genetic Algorithm, Discovery and Composition

Sumário

Lista de Figuras

Lista de Algoritmos

CAPÍTULO 1

Introdução

First principles, Clarice. Simplicity.
— HANNIBAL LECTER (The Silence of the Lambs)

Any intelligent fool can make things bigger and more complex... It takes a touch of genius - and a lot of courage to move in the opposite direction.
— ALBERT EINSTEIN

1.1 Apresentação

A Web Semântica, tal como definida por Berners-Lee, Hendler e Lassila(1), é uma ampliação da web atual na qual a informação passa a ter um significado definido, viabilizando o trabalho cooperativo entre homens e, principalmente, entre máquinas através da inclusão de dados e informações que seriam automaticamente manipulados por agentes inteligentes e utilitários. O surgimento de padrões web que expressam significado compartilhado, tais como os definidos e citados por Shadbolt, Berners-Lee e Hall(2), viabilizam a solução de grandes problemas já identificados anteriormente, tais como integração, colaboração e automação.

A integração, segundo Mahmoud e Gomez(3), é um dos maiores problemas enfrentados por departamentos de TI e equipes de desenvolvimento mundo afora. Foi este problema que motivou a filosofia da Arquitetura Orientada a Serviços (SOA). Nela, os blocos de funcionalidade das aplicações estariam separados em unidades chamadas de serviços, que seriam disponibilizados em rede e, por sua vez, se integrariam para formar aplicações inteiras. Em um cenário ideal, todos os novos sistemas de determinado projeto ou empresa seriam compostos de funcionalidades encapsuladas em serviços que seriam herdadas das já existentes em sistemas legados. Assim, tal como descrito por Brehm, Gomez e Rautenstrauch(4), seria viabilizado o sonho utópico do reuso de software atingindo a casa dos 100% e, finalmente, viabilizada a integração tanto idealizada em sistemas ERP[1], por exemplo.

Vislumbrando um cenário onde os serviços de determinada rede seriam disponibilizados para que parceiros e clientes externos contribuíssem com o desenvolvimento de novos serviços

[1] Enterprise Resource Planning.

e também com o fluxo de atualização de informação, a colaboração destes no desenvolvimento de novas soluções alcançaria níveis sem precedentes através do uso de SOA. Tal abordagem colaborativa promoveria ganhos para ambos os lados.

Quanto à implementação de SOA, diversas tecnologias e protocolos foram utilizados anteriormente[2] mas todas tinham problemas graves quanto à extensibilidade, dependência de plataforma e segurança em seus padrões de comunicação. A reação da comunidade científica aos problemas de integração e colaboração no desenvolvimento de aplicações veio com o surgimento dos Serviços Web[3], que expandiu as fronteiras da SOA e permitiu que a integração se desse, daí por diante, na Web.

Serviços Web são componentes de software fracamente acoplados, publicados, localizados e invocados através da Web, definição dada por Ma, Zhang e He(5). Sua popularidade tem aumentado bastante, principalmente, pelo fato de se basearem em padrões abertos para interfaces e definições de protocolos (HTTP com linguagens de definição estendidas do XML), o que permite abstrair a tecnologia utilizada para a implementação da unidade de negócio que um serviço representa.

Um exemplo clássico de uma aplicação de cotações em lojas virtuais é exibido na figura 1.1. Nela, uma aplicação cliente se comunica com o servidor e este, por sua vez, utiliza protocolos baseados na Web para se comunicar com os Serviços Web de cada uma das lojas participantes do sistema. O retorno é a cotação dos preços para o produto desejado.

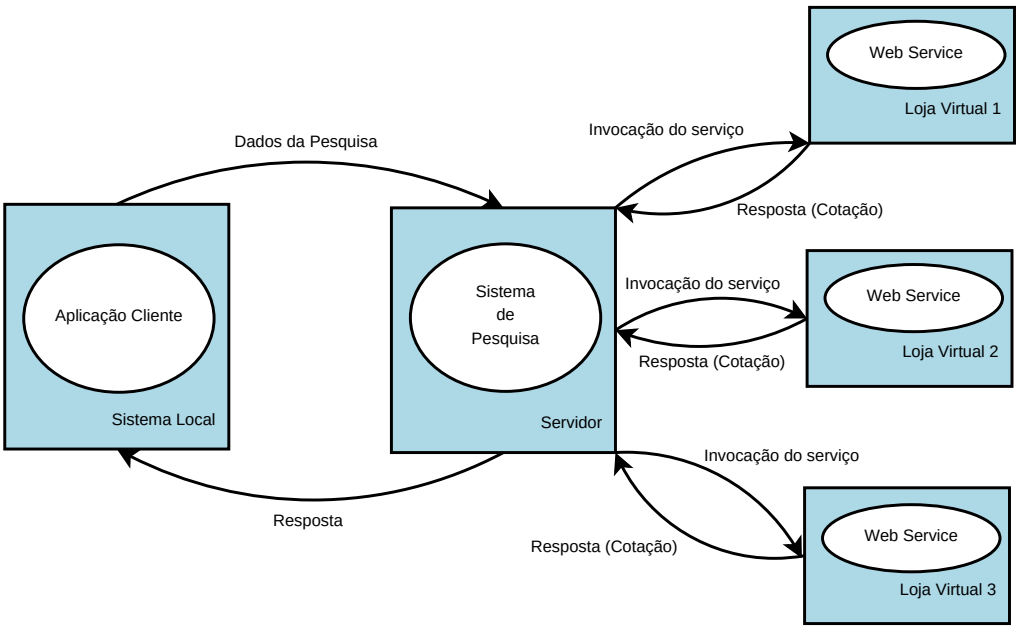

Figura 1.1 Sistema de pesquisa em lojas online com o servidor utilizando diversos Serviços Web

[2]Tais como DCOM com RPC, CORBA com IIOP e EJB com RMI que são, respectivamente, exemplos de tecnologias e protocolos.

[3]Também conhecidos como Web Services, em sua terminologia oficial na língua inglesa.

A possibilidade de disponibilizar serviços na Web, a quem estiver interessado, expandiu o modelo de negócios onde os desenvolvedores de produtos colaboravam apenas com os seus parceiros. Neste novo modelo de negócios, produtores independentes de serviços os disponibilizariam em "páginas amarelas"[4] de sítios que representam catálogos de serviços na Web. E com o crescente volume de interessados neste novo modelo de negócios, surgiram também as necessidades de padronização com fins de facilitar a busca por serviços. Tecnologias foram criadas neste sentido, das quais o UDDI[5], ou *Universal Description, Discovery and Integration*, pode ser citado.

A prosperidade de tal abordagem ganhou uma variedade de aplicações em domínios distintos e a comunidade científica destacou duas grandes áreas de investimento: automação na descoberta e na composição de Serviços Web. Automação na descoberta de serviços porque, dado o crescimento na quantidade de serviços dispostos em páginas amarelas, a ineficiência em procurar manualmente por serviços que melhor cumpririam determinadas tarefas cresceu proporcionalmente à multiplicação de serviços disponíveis na rede. E uma vez que a interface dos serviços já está padronizada no sentido de que todos são basicamente expressos em termos de entradas, saídas e uma descrição da unidade de negócio que o mesmo implementa, a integração de dois ou mais serviços para prover um novo serviço mais completo - e assim atender a uma necessidade momentânea e específica - é, portanto, atividade desafiadora e encorajada. Tal integração, também conhecida como Composição de Serviços Web, tem o propósito de prover um serviço unificado ao usuário final, escondendo assim a distribuição - principalmente geográfica - e a heterogeneidade - quanto às tecnologias de implementação - dos serviços oferecidos pelos provedores.

No entanto, uma vez que a descrição dos serviços é geralmente feita como texto livre em WSDL, tal como exemplificado por Christensen et al.(6), adicionar uma descrição semântica baseada no contexto aos serviços é uma atividade que tem ganhado importância e tem conduzido a pesquisa no desenvolvimento da habilidade de combinar serviços semanticamente, ao invés de fazê-lo sintaticamente. Tal desafio conduz ramos de pesquisa à criação de mecanismos para recuperação de semântica a partir das descrições sintáticas dos serviços cujas estratégias descritas por Ma, Zhang e He(7) e Hicks, Govindaraju e Meng(8) podem ser dadas como exemplo. Os Serviços Web que se utilizam de uma linguagem ontológica[6] para especificar sua descrição são chamados Serviços Web Semânticos - ou SWS.

Como será visto nas seções posteriores deste trabalho, há diversas estratégias que objetivam descobrir e/ou compor Serviços Web Semânticos automaticamente. E a solução do problema utilizando uma abordagem específica, baseada em algoritmos evolucionários, é o objetivo deste trabalho.

[4] Exemplo em http://seekda.com/

[5] http://uddi.xml.org

[6] Segundo Vasconcelos et al.(9), trata-se de uma linguagem que usa uma terminologia partilhada e consensual que torna adequada a partilha e a reutilização de informação.

1.2 Objetivos

1.2.1 Objetivos Gerais

- Apresentar um estudo comparativo das abordagens existentes que se propõem a resolver o problema da descoberta e composição automáticas de Serviços Web Semânticos.
- Apresentar uma proposta de solução (modelagem e implementação) baseada em algoritmo evolucionário, para o problema dado.

1.2.2 Objetivos Específicos

- Modelar o problema da descoberta e composição automáticas de Serviços Web Semânticos:
  - Fornecer uma descrição formal para o problema em questão;
  - Instanciar o formalismo fornecido para uma solução baseada em computação evolucionária.
- Implementar, numa linguagem orientada a objetos, uma ferramenta que realize a modelagem criada:
  - Carregar um conjunto de serviços disponíveis em um repositório web;
  - Descobrir e compor de Serviços Web Semânticos, dada uma requisição;
  - Fornecer composições utilizando *split* de serviços, desde que possível;
  - Manipular ontologias utilizadas na descrição dos SWS;
  - Permitir ao usuário refinar a similaridade entre sua consulta e a resposta fornecida através do ajuste do grau de relevância do conjunto resposta encontrado;
  - Gerenciar a explosão combinatorial resultante do processo de composição de serviços;
  - Retornar como melhor resposta a composição de SWS que tenha o menor número de serviços envolvidos e que atenda ao critério de similaridade mínima fornecido.
- Analisar o comportamento da ferramenta mediante dados de entrada fornecidos:
  - Confrontar os resultados obtidos com as expectativas esperadas em termos de qualidade e desempenho da solução;
  - Explicitar a contribuição dada por este trabalho à solução do problema nos termos nos resultados encontrados;

1.3 Relevância do Problema

Como será visto nos capítulos e seções posteriores deste trabalho, há diversas estratégias que objetivam descobrir e/ou compor Serviços Web Semânticos automaticamente. No entanto, não há uma alternativa aceita amplamente pela comunidade científica como solução ideal para o problema, tal como amplamente discutido em eventos como o *2008 IEEE International Conference on Web Services* (10).

Este problema se torna relevante em aplicações do mundo real quando entendemos que as necessidades dos usuários podem figurar entre as mais diversas, criando requisições que resultam quase que obrigatoriamente em respostas dadas por composições de serviços. Ainda, Meditskos e Bassiliades(11) afirmam que, dada uma requisição, não se espera que haja um serviço (ou composição) que combine em 100% (em termos de entrada e saída desejados) com esta requisição. Assim, Klusch, Fries e Sycara(12) estabelece que, na estratégia utilizada, devem ser retornados como resposta os casos de maior semelhança possível com a solução procurada. Portanto, seria possível traduzir o problema em um cenário real, dado por uma requisição de serviço que é atendida, de forma transparente ao usuário, por uma composição cujos serviços têm a capacidade de fornecer uma solução que se assemelhe à requisição, e esta solução, embora não seja exatamente o que o usuário procura, seja considerada satisfatória para o mesmo.

Ambos o problema da descoberta e composição automáticas de SWS e o problema do grau de semelhança entre um candidato encontrado (abstraindo a quantidade de serviços que este encapsula) e uma requisição são tratados tanto pela modelagem proposta por este trabalho quanto pela implementação realizada.

1.4 Estrutura da Dissertação

O Capítulo 2 apresenta conceitos gerais sobre descoberta e composição de SWS, traz as definições formal e informal do problema e o “estado da arte” desta pesquisa, com abordagens e algoritmos utilizados para descobrir e/ou compor Serviços Web Semânticos. O Capítulo 3 traz a definição do GAADT, com tipos e operadores particulares, bem como sua especificação para o problema da descoberta e composição de SWS. Detalhes técnicos sobre a implementação realizada são apresentados e discutidos no Capítulo 4, que também traz os resultados sobre as experimentações realizadas. As considerações finais sobre a pesquisa desenvolvida são apresentadas no Capítulo 5. O Apêndice A traz o formalismo do Algoritmo Genético Baseado em Tipos Abstratos de Dados - GAADT - e o Apêndice B traz maiores detalhes sobre a definição e o cálculo do cosseno para a determinar a similaridade entre conceitos, utilizada na função de avaliação do GAADT.

CAPÍTULO 2

Descoberta e Composição de Serviços Web Semânticos

Se, a princípio, a ideia não é absurda, então não há esperança para ela.
— ALBERT EINSTEIN

Conhecer não é demonstrar nem explicar, é aceder à visão.
— ANTOINE DE SAINT-EXUPÉRY (O Pequeno Príncipe)

2.1 Introdução

Como já citado no Capítulo 1, os serviços web possibilitaram um aumento sem precedentes nos níveis de reaproveitamento de funcionalidades, integração e colaboração entre sistemas. Porém, a adoção de serviços web no desenvolvimento de novas aplicações expande a lista de requisitos necessários para além da descoberta e composição de serviços - ainda que estes últimos possam ser considerados grandes grupos de pesquisa da comunidade científica na área. Yu(13) define mais itens para a lista de necessidades que precisam ser atendidas na adoção de serviços web, bem como suas motivações, que seguem:

1. *Descoberta automática de Serviços Web*: Encontrar o serviço desejado pode ser difícil, especialmente quando o requerente do serviço não sabe da existência do serviço provido; a única esperança do requerente seria a de alguém já ter provido o serviço *online*. No entanto, para tornar os Serviços Web um verdadeiro sucesso, uma forma de descobrir o serviço requerido deve ser provida; também, o mesmo deve ser descoberto automaticamente, com grande precisão e eficiência.

2. *Invocação Automática do Serviço*: Após o serviço requerido ter sido descoberto, o agente de software deve ser capaz de invocar o serviço automaticamente. O benefício é óbvio: sem atrasos causados por intervenção humana, pode-se levar um negócio a uma larga escala com eficiência muito maior. Também, em muitos casos, essa invocação automática é simplesmente uma obrigação; algumas aplicações devem rodar continuamente e sem interrupções.

3. *Composição Automática dos Serviços Necessários*: Muito frequentemente, uma necessidade de negócio específica requer que vários Serviços Web trabalhem em conjunto. Por exemplo, um replanejamento de inventário envolverá a consulta de preços em vários distribuidores - chamando os Serviços Web providos por esses distribuidores -, comparando os preços e criando os pedidos - este é outro Serviço Web provido pelo distribuidor de onde se decide comprar. Claramente, um agente de software deve ser capaz de encontrar todos os serviços necessários e invocá-los na ordem correta para cumprir o objetivo do negócio.

4. *Monitoramento Automático do Processo de Execução*: Claramente, se todos os serviços precedentes são automáticos, então como saber se um serviço foi encontrado e executado com sucesso e corretamente? Deve haver um mecanismo para detectar e informar possíveis falhas, caso elas aconteçam.

Desde o estabelecimento dos serviços web, diversos trabalhos já foram publicados com o intuito de prover uma solução para um ou mais dos problemas listados acima - o que tornaria a tentativa de citá-los todos uma tarefa árdua e dinâmica, visto que a produção acadêmica relacionada tem grande escala e dinamicidade, além de resultados interessantes em termos de qualidade e eficiência que têm sido apresentados com frequência cada vez maior nos últimos eventos relacionados.

Para os fins deste trabalho, é suficiente entender um Serviço Web Semântico como uma estrutura dada pela Figura 2.1, que representa um SWS com cada elemento de seus conjuntos de entradas e saídas sendo descrito através de referências a termos de uma ontologia.

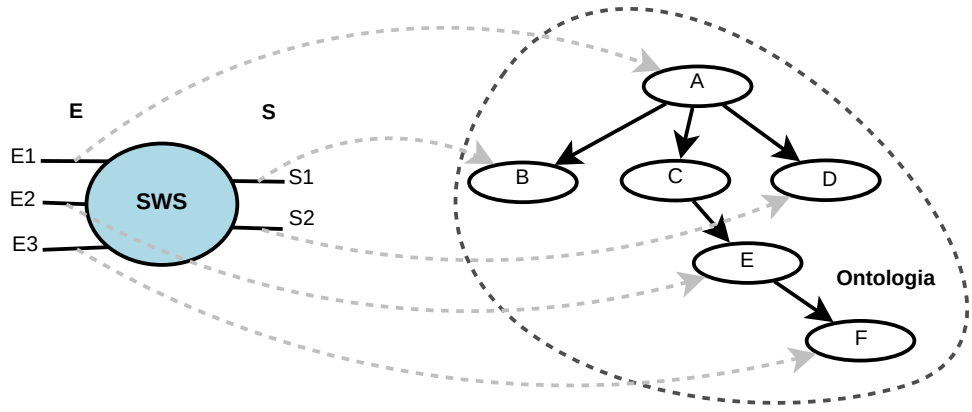

Figura 2.1 Representação de um SWS

A anotação semântica de serviços, tal como descrita por Liu, Peng e Chen(14), elimina o processo de extração semântica da descrição dos mesmos, bem como de seus parâmetros de entrada e saída. Assim, elimina-se o problema de haver dois serviços idênticos em suas funções e parâmetros, mas com descrições ligeiramente diferentes (texto descritivo utilizando sinônimos ou até mesmo em outra língua), fato este que exigiria um esforço na construção de um mecanismo de recuperação de informação.

A descoberta de SWS dá-se através da comparação entre as entradas e saídas de uma requisição e as entradas e saídas dos serviços presentes no repositório onde se deseja pesquisar. Caso haja um serviço cujas entradas e saídas combinem, então diz-se que houve um *match* direto, e a requisição é atendida por um serviço unitário.

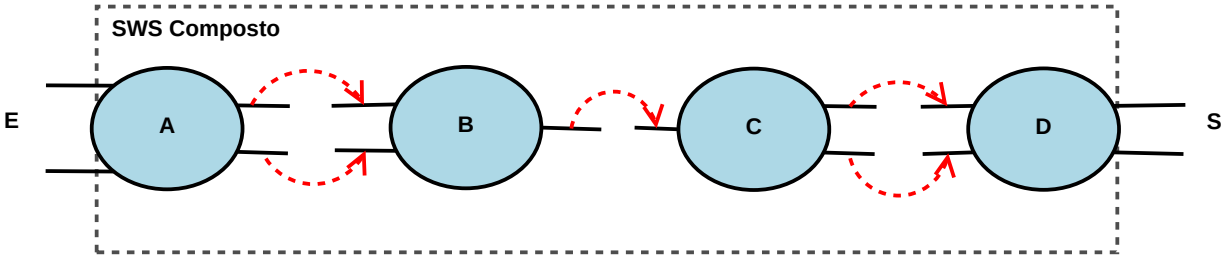

Figura 2.2 Composição de SWS

A Figura 2.2 representa uma composição de Serviços Web Semânticos. Nesta, um serviço que engloba a composição de 4 outros serviços é representado pelo conjunto de entradas do primeiro serviço e de saídas do último. E os serviços encapsulados se ligam através da relação de herança entre os conceitos que eles referenciam da seguinte forma: se um conceito utilizado para definir a saída do serviço A é, ao menos, um subconceito de uma entrada do serviço B, então os dois podem ser ligados. Um serviço B é considerado apto para a composição se todas as suas entradas forem providas pelos serviços anteriores a ele.

Há diversas estratégias de descoberta e composição de serviços. Os detalhes destas serão analisados nas seções posteriores deste capítulo.

2.2 Descrição Formal

Nesta seção, utilizaremos o formalismo definido em Cardoso(15) para descrever as operações e propriedades desejadas de uma composição de Serviços Web Semânticos.

De forma a discutir esta ideia propriamente, são necessários alguns pré-requisitos. Assim, vamos inicialmente definir o conjunto de todos os conceitos semânticos $\mathbb{M}$. Todos os conceitos que existem na base de conhecimento são membros de $\mathbb{M}$ e podem ser representados como nós de uma árvore, e destas, são respeitadas todas as propriedades de ordens parcial e total, como segue:

Definição 2.1. A relação $\subseteq$ de ordem parcial para grafos é reflexiva, anti-simétrica e transitiva. Considere α e β grafos:

1. *Reflexiva*: $\alpha \subseteq \alpha$;
2. *Anti-simétrica*: Se $\alpha \subseteq \beta$ e $\beta \subseteq \alpha$ então $\alpha = \beta$.
3. *Transitiva*: Se $\alpha \subseteq \beta$ e $\beta \subseteq \gamma$ então $\alpha = \gamma$

Provas para as propriedades listadas podem ser encontradas em MENEZES(16).

Assumindo e estendendo as propriedades dos grafos citadas acima para os elementos do conjunto $\mathbb{M}$, definiremos então a relação *compreende*:

Definição 2.2. Compreende: Dois conceitos $A, B \in \mathbb{M}$ podem se relacionar de quatro formas diferentes. A função *compreende* : $(\mathbb{M} \times \mathbb{M}) \Rightarrow \{Verdadeiro, Falso\}$ expressa essa relação, como segue:

1. $compreende(A,B)$ é verdadeiro se, e somente se, $A \subseteq B$ (B é então uma especialização de A);
2. $compreende(B,A)$ é verdadeiro se, e somente se, $B \subseteq A$ (B é então uma generalização de A);
3. Se nem $compreende(A,B)$ nem $compreende(B,A)$ são verdadeiros, A e B não se relacionam entre si, ou seja, $A \cap B = \emptyset$;
4. $compreende(A,B)$ e $compreende(B,A)$ é *verdadeiro* se, e somente se, $A = B$.

Assim, a função *compreende* é transitiva, anti-reflexiva e associativa e também o são a generalização e a especialização.

Se um parâmetro x de um serviço é anotado com A e, no entanto, somente um valor y anotado com B está disponível, se $compreende(A,B)$ for verdadeiro (*contravariância*), poderemos invocar o serviço referido assumindo $x = y$. Isto significa que y representa informação pelo menos igual à dada por x. Um exemplo pode ser visualizado por um serviço que necessite de uma instância de *veículo* e que pode ser executado com uma instância de *carro*, visto que *carro* seria uma especialização de *veículo*.

Do ponto de vista de um algoritmo de composição, não há necessidade de distinção entre parâmetros e os conceitos anotados. O conjunto $\mathbb{S}$ contém todos os serviços s conhecidos do repositório. Cada serviço $s \in \mathbb{S}$ tem um conjunto de conceitos requeridos de entrada $s.in \subseteq \mathbb{M}$ e um conjunto de conceitos de saída $s.out \subseteq \mathbb{M}$ que será entregue como retorno. Pode-se ativar um serviço se forem providos todos os seus parâmetros de entrada.

Similarmente, um pedido de composição R sempre consiste de um conjunto de conceitos de entrada $R.in \subseteq \mathbb{M}$ e um conjunto de saídas requeridas $R.out \subseteq \mathbb{M}$. Um algoritmo de composição descobre um conjunto de n serviços $S = s_1, s_2, ..., sn : s_1, ..., s_n \in \mathbb{S}$. Assim como mostrado na equação (2.1), o primeiro serviço (s_1) de uma composição válida pode ser executado com instâncias dos conceitos de entrada $R.in$. Juntamente com $R.in$, suas saídas ($s.out$) estão disponíveis para executar os próximos serviços (s_2) de S, e assim por diante. A composição provê saídas que podem ser anotadas tanto com os conceitos exatos de $R.out$ quanto com outros mais específicos (covariância). Podemos, também, definir uma relação *ehObjetivo* : $(\mathbb{S} \times \mathbb{S}) \Rightarrow \{Verdadeiro, Falso\}$ para cada composição S que soluciona a requisição R. Assim, a função $ehObjetivo(S)$ é dada por:

$$\begin{aligned} ehObjetivo(S) \Leftrightarrow \forall A \in s_1.in\ \exists B \in R.in : compreende(A,B) \wedge \\ \forall A \in s_i.in, i \in \{2..n\}\ \exists B \in s_{i-1}.out \cup ... \cup s1.out : compreende(A,B) \wedge \\ \forall A \in R.out\ \exists B \in s_n.out : compreende(A,B) \end{aligned} \tag{2.1}$$

2.3 Abordagens e Algoritmos de Descoberta e Composição de SWS

Muitas abordagens de descoberta e composição automáticas de Serviços Web foram e vêm sendo propostas. Estas variam desde Composição baseada em Planejamento Yan e Zheng; Sheshagiri, Desjardins e Finin; Wu et al.(17–19) a Composição baseada em Fluxo de Trabalho (*workflow*), passando por Composição baseada em Transições (interativa) Hull e Su(20). Para a descoberta e composição de Serviços Web Semânticos, podemos classificar as abordagens disponíveis em quatro grandes grupos:

- Abordagem Não-Informada
- Abordagem Heurística (Informada)
- Abordagem SMA (Sistemas Multi-Agentes)
- Abordagem Evolucionária

2.3.1 Abordagem Não-Informada

Como os algoritmos de busca não-informada não se utilizam de nenhuma informação diferente de predicados de objetivo, tal como definidos na seção anterior, pode-se construir um algoritmo de composição baseado em busca de profundidade iterativa, ou *iterative deepening depth-first search (IDDFS)*, que é considerada a mais geral e direta para a composição de Serviços Web Semânticos de acordo com Weise et al.(21). Tal algoritmo é rápido para encontrar soluções para pequenos repositórios de serviços e é ótimo se o problema requer uma busca exaustiva.

Considerando que o espaço de busca que precisa ser investigado na composição de SWS é o conjunto de todas as permutações possíveis de todos os conjuntos de serviços. Definimos, então, a operação *promissor* que contém o conjunto de todos os serviços $s \in \mathbb{S}$ que produzem um parâmetro de saída anotado com o conceito A (independentemente de suas entradas).

$$\forall s \in promissor(A)\ \exists B \in s.out : compreende(A,B) \tag{2.2}$$

O Algoritmo 1 mostra uma proposta para a composição não-informada.

```
Input: R a requisição de composição
Data: profundidadeMaxima, profundidade as profundidades máxima e atual de pesquisa
Data: in, out conjuntos de parâmetros
Output: S uma composição de serviços que soluciona R
begin
    profundidadeMaxima ← 2
    repeat
        S ← dfs(R.in,R.out,∅,1)
        profundidadeMaxima ← profundidadeMaxima + 1
    until S ≠ ∅
end
dfs(in,out,composition,profundidade)
begin
    foreach A ∈ out do
        foreach s ∈ promissor(A) do
            wanted ← out
            foreach B ∈ wanted do
                if ∃C ∈ s.out : compreende(B,C) then
                    wanted ← wanted \ {B}
                end
            end
            foreach D ∈ s.in do
                if ∄E ∈ in : compreende(D,E) then
                    wanted ← wanted ∪ {D}
                end
            end
            comp ← s ⊕ composition
            if wanted = ∅ then
                return ∅
            end
            if profundidade < profundidadeMaxima then
                comp ← dfs(in,wanted,comp,profundidade + 1)
                if comp ≠ ∅ then
                    return comp
                end
            end
        end
    end
    return ∅
end
```

Algoritmo 1: Composição baseada em busca de profundidade iterativa IDDFS (Não-Informada)

O algoritmo 1 constrói um SWS válido por uma estratégia de encadeamento para trás

(*Backward Chaining*). A cada recursão, seu método interno *dfs* testa todos os elementos *A* do conjunto *wanted* de parâmetros ainda desconhecidos. Ele então itera sobre o conjunto de todos os serviços *s* que possam prover *A*. Para cada *s*, *procurado* é recomputado. Se ele se tornar o conjunto vazio $\emptyset$, uma composição válida foi encontrada e então é retornada. Se a função *dfs* não for capaz de encontrar uma solução dentro da profundidade máxima estabelecida (que denota o número máximo de serviços em uma composição), ela retora $\emptyset$. O *loop* no Algoritmo 1 invoca iterativamente *dfs* através do incremento da profundidade limite passo a passo, até que uma solução válida seja encontrada.

Pode-se citar, entre a literatura relacionada a esta abordagem:

- *SWSDS*[1]: O sistema de composição SWSDS Xu et al.(22) pode ser utilizado para *matching* sintático e semântico através da mudança de um índice de busca. O sistema usa um algoritmo de composição muito similar ao apresentado nesta sessão, mas estendido com a restrição de um serviço ser considerado apenas uma vez para ser parte de uma composição. Isso mitiga o fraco desempenho da abordagem da busca não-informada mas contrasta com a otimalidade garantida do resultado.

2.3.2 Abordagem Heurística (Informada)

O algoritmo mostrado na seção anterior é devagar e um grande consumidor de memória para repositórios maiores, uma vez que ele não utiliza nenhuma informação adicional sobre o espaço de estados. Se tal informação for utilizada, pode-se aumentar a eficiência de uma busca, tal como afirmado por Weise et al.; Paliwal, Adam e Bornhovd(21, 23). Resultados experimentais sobre estratégias que consideram informações de contexto podem ser obtidos em Pietro et al.; Ma, Zhang e He(5, 24).

Em uma busca informada, uma heurística *c* ajuda a decidir quais nós devem ser expandidos a seguir. Se a heurística é boa, estes algoritmos terão uma performance melhor que estratégias de busca não-informada, tal como afirmam Russell e Norvig(25). Como um segundo método, será definido um algoritmo de busca gulosa (*greedy search*) que internamente organiza a lista de composições candidatas conhecidas atualmente em ordem *decrescente* de acordo com a heurística na forma de uma função de comparação $e : \mathbb{S}^n \in \mathbb{R}$. A função comparadora $e(S_1, S_2)$ terá valor abaixo de zero se S_1 parecer estar mais próximo da solução que S_2 e maior que zero se S_2 for um melhor candidato. Assim, os melhores elementos estarão no final da lista *X* do algoritmo 2

[1]SWSDS = *SEWSIP Web Service Discovery System*, SEWIP = *Semantic Web Services Integration Platform*.

Input: R a requisição de composição
Data: X a lista sorteada de composições para explorar
Output: S a composição solução encontrada, ou $\emptyset$

```
begin
    X ← ∪_{∀A∈R.out}(promissor(A))
    while X ≠ ∅ do
        X ← sort(decrescente,X,e)
        S ← retireUltimoElemento(X)
        if ehObjetivo(S) then
            return S
        end
        foreach A ∈ procurado(S) do
            foreach s ∈ promissor(A) do
                X ← acrescentarLista(X,s⊕S)
            end
        end
        return ∅
    end
end
```

Algoritmo 2: Composição baseada em busca gulosa (*greedy search*).

Pode-se derivar funções de comparação que levam em conta vários fatores, aumentando assim a eficiência da heurística e. Por exemplo, pode-se combinar o tamanho do conjunto de parâmetros não-satisfeitos $\forall A \in procurado(S) \Rightarrow \exists s \in S : As.in \wedge A \notin conhecidos(S)$, os tamanhos das composições, o número de parâmetros satisfeitos $\forall B \in eliminados(S) \Rightarrow \exists s \in S : B \in s.in \wedge B \in conhecidos(S)$, e o número de conceitos conhecidos $conhecidos(S) = R.in \cup_{\forall s \in S} s.out$ como definido no Algoritmo 3.

Primeiro, o algoritmo compara o número de parâmetros requeridos. Se uma composição não contém tais parâmetros não atendidos, é uma solução válida. Se ambos S_1 e S_2 são válidos, a solução envolvendo a menor quantidade de serviços vence. Se somente uma delas é completa, ela também vence. Caso contrário, ambos os candidatos ainda possuem conceitos não atendidos. Somente são comparados novamente os conceitos requeridos se ambos os serviços tiverem o mesmo número de parâmetros atendidos. Se os números de parâmetros dos serviços é igual, então a composição mais curta é preferida. Se até mesmo as composições envolverem a mesma quantidade de serviços, então a decisão é baseada no número total de conceitos.

```
Input: S1,S2 duas composições candidatas
Output: r ∈ ℤ indicando se S1 (r < 0) ou S2 (r > 0) deve ser expandido em seguida
begin
    i1 ←| procurado(S1) |
    i2 ←| procurado(S2) |
    if i1 = 0 then
        if i2 = 0 then
            return | s1 | − | s2 |
        end
        return-1
    end
    if i2 = 0 then
        return 1
    end
    e1 ←| eliminado(S1) |
    e2 ←| eliminado(S2) |
    if e1 > e2 then
        return-1
    else
        if e1 < e2 then
            return 1
        end
    end
    if i1 > i2 then
        return-1
    else
        if i1 < i2 then
            return 1
        end
    end
    if | S1 |≠| S2 | then
        return | S1 | − | S2 |
    end
    return | conhecido1 | − | conhecido2 |
end
```

Algoritmo 3: Heurística de parâmetros de classificação entre duas composições candidatas.

Pode-se citar, entre a literatura relacionada a esta abordagem:

- Zhang et al.(26): No seu sistema de composição, são utilizadas tabelas *hash* que mapeam o serviço aos seus parâmetros de entrada e saída. Com estas tabelas, ambas as composições usando encadeamento para frente e para trás podem ser feitas utilizando busca em largura. Um encadeamento para trás sucessivamente encontra serviços que

provém parâmetros desconhecidos até que todas as saídas requeridas sejam encontradas. O encadeamento para frente significa que os serviços que podem ser invocados com os parâmetros conhecidos são adicionados iterativamente à composição até que todos as saídas requeridas possam ser geradas. Zhang et al.(26) afirma que o encadeamento para trás funciona melhor que o encadeamento para frente.

- Chahoud(27): No seu trabalho, a autora cria um mecanismo de composição automática de serviços web semânticos com base em estratégias de planejamento, cuja base se dá através das buscas heurísticas tal como definido por Russell e Norvig(25). Neste, as atividades de descobrir, compor e invocar serviços, detalhadas no início deste capítulo, são tratadas todas ao mesmo tempo. Assim, cenários que se reconfiguram durante a invocação de serviços são previstos e modelados a partir de uma linguagem própria de representação que é baseada em lógica de primeira ordem. Assim, as composições de serviços serão resolvidas previamente, e o trabalho direcionado à busca heurística é o de encontrar serviços, dentre os disponíveis em determinado repositório, que se adequam aos critérios informados.

2.3.3 Abordagem SMA (Sistemas Multi-Agentes)

Um Sistema Multi Agente (SMA) consiste em um time ou organização de agentes de software executando, em conjunto, uma tarefa que não poderia ser executada por nenhum agente individualmente (Ferber; Fernández e Ossowski(28, 29)). A literatura abordada entende um sistema de composição de SWS como um SMA da seguinte forma: cada serviço componente é considerado como uma capacidade de um agente implementada num componente de software auto-contido.

A Figura 2.3 ilustra uma proposta de arquitetura para Sistemas Multi-Agentes, tal como definido em Kumar e Mishra(30), que tratam o problema da descoberta e composição de Serviços Web Semânticos. Nesta concepção, o Agente Usuário U fornece os parâmetros de entrada ao sistema de composição, que então são especificados em termos de uma ontologia[2]. Usando os parâmetros especificados na descrição da ontologia do pedido, os agentes coordenadores candidatos são descobertos e então o melhor é selecionado. O agente coordenador selecionado C agora decide, a partir da descrição da ontologia, se o requisito de entrada é uma atividade atômica ou complexa. No caso de uma atividade complexa, ela é então decomposta em tarefas atômicas $Tarefa_1, Tarefa_2, Tarefa_3, ..., Tarefa_N$. No entanto, antes de decompor a requisição, C pode executar uma validação sobre a requisição de entrada para verificar se todos os parâmetros, preferências e restrições especificadas na requisição são adequadas ou

[2]A definição de Ontologia não é clara e as fontes bibliográficas variam quanto à sua formalização. Almeida M.(31) a define como:

> "Um tipo de estrutura utilizada na organização da informação ... que se organiza a partir de conceitos e de seus relacionamentos de forma semelhante a uma estrutura taxonômica."

Para os fins deste trabalho, é suficiente entender uma ontologia como uma estrutura computacional que realiza a definição dada acima.

não. Finalmente, o agente coordenador *C* descobre e então seleciona os agentes dos provedores de serviço correspondentes $Agente_1, Agente_2, Agente_3, ..., Agente_N$ para as tarefas atômicas $Tarefa_1, Tarefa_2, Tarefa_3, ..., Tarefa_N$, respectivamente. O agente coordenador *C* negocia com os agentes provedores e lhes atribui tarefas, que posteriormente invocam os serviços.

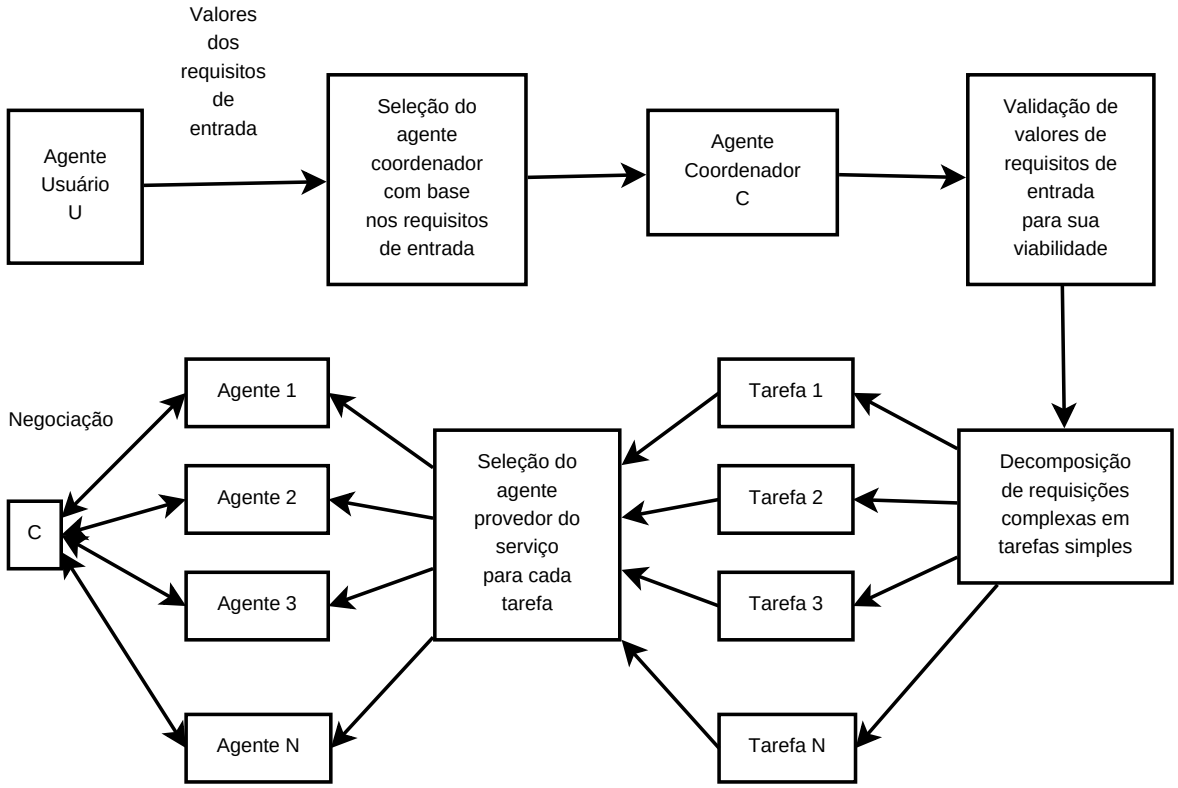

Figura 2.3 Arquitetura de um Sistema Multi-Agentes para descoberta e composição de Serviços Web Semânticos

Tal como definido em Kumar e Mishra(30) e disposto na Figura 2.3, um SMA de descoberta e composição de SWS consistiria, basicamente, de três tipos de agentes:

- Agente de Requisição de Serviço (ARS);
- Agente Coordenador (AC);
- Agente Provedor de Serviço (APS).

O Agente de Requisição de Serviço(ARS) tem a responsabilidade de fazer uma requisição ao Agente Coordenador(AC). A requisição do ARS é especificada em termos da ontologia e então é utilizada pelo AC. Um Agente Coordenador inteligente possui as seguintes propriedades e capacidades:

- O Agente Coordenador é um componente de *software* modular e auto-contido que encapsula serviços de coordenação cujas descrições são dadas através de referências a termos da ontologia (descrição ontológica);

- Ele possui a capacidade de validar restrições, preferências e outros parâmetros de alto nível da requisição de entrada dada pelo Agente de Requisição de Serviço;
- Ele tem a capacidade de decidir se a atividade de entrada é atômica ou complexa, interpretando-a como uma tarefa que consiste de várias atividades de granularidade variável e a decompondo em tarefas atômicas de acordo com suas descrições ontológicas.
- Ele pode avaliar os Agentes Provedores de Serviço usando seus parâmetros de Qualidade de Serviço (QoS);
- Ele pode negociar com os APS assim como com o ARS para ajustar as entradas das atividades, as preferências e restrições do ARS para obter parâmetros IOPE[3] comuns e satisfazer a requisição.

Já um Agente Provedor de Serviço tem as seguintes propriedades e capacidades:

- Um Agente Provedor de Serviço (APS) é auto-contido e modular, e encapsula serviços na forma de componentes de software com sua descrição ontológica correspondente.
- O propósito de um APS é decidido pelos serviços que ele encapsula.
- Um APS se junta a um processo de composição apenas durante o tempo em que seu serviço é requerido.

Diversas abordagens com a forma especificada acima estão disponíveis na literatura, dentre as quais citamos:

- *SCE*: O *Multiagent Web Service Composition Engine*, Buhler, Greenwood e Weichhart(33), consiste de dois componentes arquiteturais primários: JADE[4] e um repositório de descrições de serviços. Neste, as requisições de serviços e composições são representados por agentes. Estes agentes se comunicam entre si e solucionam as requisições de forma cooperativa.

Normalmente, abordagens baseadas em IA tais como o SCE são mais lentas que métodos otimizados e específicos para problemas tais como o algoritmo de busca gulosa apresentado na seção anterior.

[3] Entradas (*Inputs*), Saídas (*Outputs*), Pré-condições e Efeitos. Representam os tipos de anotações semânticas disponíveis na OWL-S. Mais detalhes e estratégias com base em todas as anotações disponíveis em Bener, Ozadali e Ilhan(32)

[4] *Java Agent Development Framework*, disponível em http://jade.tilab.com/

2.3.4 Abordagem Evolucionária

Segundo Bäck(34), algoritmos evolucionários são algoritmos genéricos de otimização meta-heurística baseados em população que utilizam mecanismos inspirados na biologia tais como mutação, cruzamento, seleção natural e sobrevivência do indivíduo mais adaptado. A vantagem dos algoritmos evolucionários em relação a outros métodos de otimização é que eles fazem apenas algumas considerações sobre o cenário de adaptação e, portanto, conseguem executar de forma satisfatória em muitas categorias de problemas.

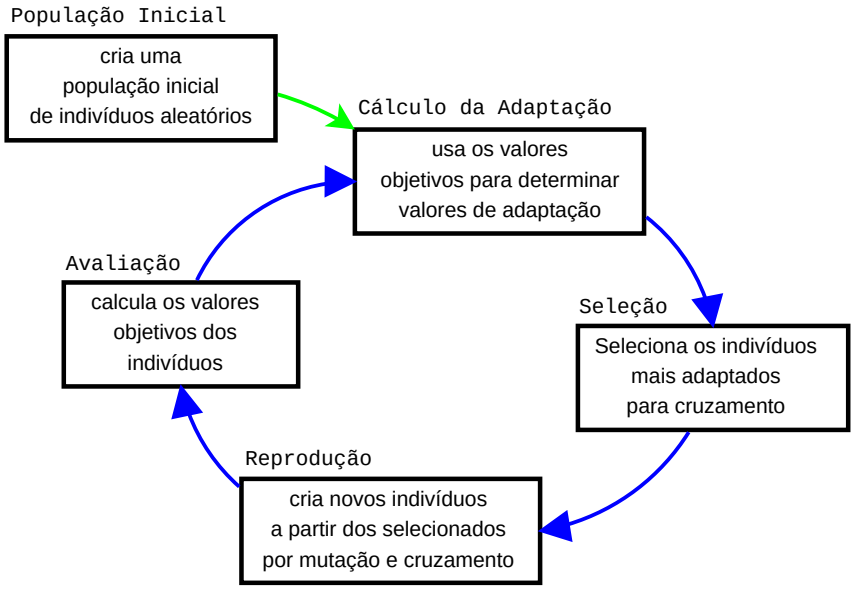

Figura 2.4 O ciclo básico dos algoritmos evolucionários

Todos os algoritmos evolucionários executam, a princípio, de acordo com os passos ilustrados abaixo:

1. Inicialmente, uma população totalmente aleatória de indivíduos é criada;
2. Todos os indivíduos são testados com relação à sua utilidade como solução;
3. Com base nessa avaliação, valores de adaptação são atribuídos aos indivíduos;
4. Um processo de seleção subsequente filtra os indivíduos com baixa adaptação e permite aos de boa adaptação participar na próxima rodada de reprodução com maior probabilidade;
5. Na fase de reprodução, uma descendência (prole) é criada através da variação ou combinação de candidatos à solução;
6. Se a condição de parada é alcançada, a evolução então para. Caso contrário, ela continua no passo 2.

Para a reprodução de candidatos a solução, os Algoritmos Evolucionários aplicam dois operadores diferentes:

- A mutação modifica ligeiramente um indivíduo existente e
- O operador de cruzamento combina dois candidatos a solução, criando resultados com características de ambos.

De forma a utilizar um algoritmo evolucionário para gerar composições de serviços semânticos, precisamos, primeiramente, definir um genoma capaz de representar sequências de serviços. Uma abordagem direta, porém eficiente, é utilizar strings (com comprimento variável) de identificadores de serviços que podem ser processados por algoritmos genéticos comuns. Então, pode-se também aplicar os operadores de criação, mutação e cruzamento.

No entanto, através da especificação de uma operação de mutação especializada, pode-se obter uma busca mais eficiente. Essa nova operação ou apaga o primeiro serviço em $\mathbb{S}$ (através do $mutacao_1$) ou adiciona um serviço promissor a $\mathbb{S}$ (como é feito em $mutacao_2$). Usando a variável ajustável σ como um limitador, pode-se determinar se uma busca deve preferir crescer ou encolher os candidatos à solução.

Definição 2.3. $mutacao_1 \equiv \begin{cases} \{s_2, s_3, \ldots, s_{|S|}\} & \text{se } |S| > 1 \\ S & \text{caso contrário} \end{cases}$

Definição 2.4. $mutacao_2 \equiv s \oplus S : A \in procurado(S) \wedge s \in promissor(A)$

Definição 2.5. $mutacao \equiv \begin{cases} mutacao_1(S) & \text{se } random() > \sigma \\ mutacao_2(S) & \text{caso contrário} \end{cases}$

Uma nova operação para construção da população aleatória inicial pode ser definida como uma sequência de invocações de tamanho variável de $mutacao_2$. Inicialmente, $mutacao_2(\emptyset)$ retornará uma composição consistindo de um único serviço que satisfaz pelo menos um parâmetro de $R.out$. Iterativamente aplica-se $mutacao_2$ ao seu resultado anterior um número aleatório de vezes para gerar um indivíduo.

A função de avaliação das composições formadas na abordagem evolucionária pode ser definida em termos da função da heurística que adiciona parâmetros de classificação para duas composições candidatas dada na seção anterior. Assim, pode-se definir uma função de avaliação que priorize:

- composições que são completas;
- composições pequenas;
- composições que solucionam muitos parâmetros não-conhecidos;
- composições que provêm muitos parâmetros.

Mais detalhes quanto a abordagens evolucionárias para a solução de composição de Serviços Web Semânticos podem ser vistos em Weise et al.; Xu e Reiff-Marganiec; Claro, Albers e Hao(21, 35, 36).

2.4 Síntese do Capítulo

Este capítulo apresentou as diversas estratégias utilizadas para resolver o problema da descoberta e composição de Serviços Web Semânticos. Ficou evidenciado que abordagens tais como a busca não-informada, embora sempre encontre uma solução ótima, pode ser um problema para bases de serviços muito grandes. A exemplo de bases reais de serviços, tais como a *Semantic Moby*[5] citada por DiBernardo, Pottinger e Wilkinson(37), que contém 2500 serviços, aproximadamente, não seria interessante utilizar tal estratégia de composição. A busca informada seria uma alternativa mais interessante, porém requer um alto nível de especialização em sua heurística e controle refinado do espaço de estados, tais como os fornecidos pelo Algoritmo 3 para conseguir bons resultados de execução.

Ainda na busca heurística, estratégias baseadas em planejamento tais como a criada por Chahoud(27) requerem quem um especialista aja previamente, resolvendo as composições e criando *templates* que indicam ao sistema o padrão de resolução de um possível problema informado, usando para isso tudo uma linguagem específica de representação em lógica de primeira ordem. Tal sistema, embora traga resultados excelentes, requer grande conhecimento acerca do domínio que está sendo tratado, e não conseguirá resolver requisições de composição que não tenham sido previstas e modeladas anteriormente.

A abordagem de Sistemas Multi-Agentes, embora venha sendo alvo de trabalhos tais como os citados neste capítulo, ainda se prova ineficiente em termos de tempo de resposta, justamente por estar apoiada em tecnologias ainda experimentais e, por isso, ainda não preocupadas em eficiência, como descreve Kumar e Mishra(30).

Uma solução interessante seria a utilização de algoritmos evolucionários para a modelagem do problema proposto, e consegue-se bons resultados tais como os mostrados em Kumar e Mishra(30). Porém, como explicado na relevância do problema, tratada na Seção 1.3, é de suma importância que uma abordagem para o problema da descoberta e composição de Serviços Web Semânticos trate do grau de similaridade entre requisições e respostas, ou seja, responda a uma requisição com “a melhor resposta possível”, ao invés da “resposta” simplesmente, independentemente destas serem compostas de um ou mais serviços.

Tal problema torna a modelagem tradicional dos algoritmos evolucionários (sequência de *bits* ou caracteres) de complexidade demasiada, pois o nível de abstração que essa abordagem traz é demasiadamente baixo. Ainda, dada a descrição formal da Seção 2.2, a abordagem desenvolvida deve prover o *split* de serviços, que é o cenário mostrado pela Figura 3.1, onde as saídas combinadas de dois ou mais serviços são utilizadas como entrada em outro serviço,

[5] Disponível em http://semanticmoby.org

num possível cenário de solução do problema. A modelagem do *split* de serviços em buscas informadas ou não-informadas também seria de difícil implementação.

O Algoritmo Genético Baseado em Tipos Abstratos de Dados, ou GAADT, proposto por Vieira(38), traz contribuições de grande importância para o desenvolvimento deste trabalho, tais como um maior nível de abstração (devido à utilização de tipos abstratos de dados ao invés de cadeias de caracteres) e, portanto, mais facilidade para modelar a similaridade entre serviços (encontrados e procurados) e o *split*, que é cenário desejável. Assim, o próximo capítulo trará uma proposta de modelagem de algoritmo evolucionário para o problema da descoberta e composição de SWS utilizando a filosofia do GAADT e maiores informações sobre ele, tais como seu formalismo e uma aplicação formal ao problema do caixeiro viajante, são encontradas no Apêndice A.

CAPÍTULO 3

Modelagem de um Algoritmo Evolucionário para Descoberta e Composição de SWS

Para ganhar conhecimento, adicione coisas todos os dias.
Para ganhar sabedoria, elimine coisas todos os dias.
— LAO-TSÉ

Foi o tempo que perdeste com tua rosa que fez tua rosa tão importante.
— ANTOINE DE SAINT-EXUPÉRY (O Pequeno Príncipe)

3.1 Introdução

O modelo de Algoritmo Genético Baseado em Tipos Abstratos de Dados (GAADT), utilizado neste trabalho, foi originalmente proposto por Vieira(38) e trabalha sobre um ambiente definido como uma estrutura na qual um dos componentes é a população. Segundo este modelo, as mudanças ambientais são vistas como o marco do início de um novo período de evolução durante o qual os cromossomos da população atual irão sofrer a ação dos operadores genéticos com o intuito de construir uma nova população formada somente por cromossomos que satisfazem aos requisitos do ambiente atual. Após o período de evolução vem o período de estagnação, durante o qual a população não evolui. O período de estagnação é finalizado quando uma nova alteração ambiental ocorre, dando início a um novo ciclo de um período de evolução seguido por um período de estagnação.

O resultado do problema para o ambiente atual é o cromossomo mais adaptado na população de estagnação atingida, se esta for compatível com uma condição de parada, a qual deverá ser definida durante o período de concepção do GAADT.

A cada período de evolução existe uma população de cromossomos extintos associada. Os cromossomos desta população são provenientes de uma das populações já trabalhadas pelo algoritmo genético, os quais foram avaliados e descartados devido ao seu baixo grau de adaptação às configurações atuais do ambiente. A presença da população de cromossomos extintos é justificada pela necessidade de evitar que seus genes possam reaparecer nas próximas populações.

O terceiro princípio fala sobre a hereditariedade das características adaptadas ao ambiente dos cromossomos pais que devem ser passadas para os cromossomos filhos. Sobre este ponto, é importante ressaltar que os operadores de cruzamento encontrados na literatura só se preocupam com o fato de que as características apresentadas nos cromossomos filhos estejam presentes nos cromossomos pais, sem se preocupar com quanto a referida característica contribui para a adaptação do cromossomo-pai ao ambiente.

O operador de cruzamento apresentado no GAADT constrói novos cromossomos somente com as características responsáveis pela adaptação dos cromossomos pais ao ambiente, as quais serão denominadas de genes dominantes. Uma função denominada grau de adaptação do gene é definida para informar o quanto uma dada característica pode contribuir com a adaptação do cromossomo ao ambiente.

O quarto princípio descarta a possibilidade dos cromossomos não adaptados ao ambiente evoluírem e até conduzirem à geração de cromossomos mais adaptados que os cromossomos mais adaptados da população atual. Neste quesito, o GAADT propõe que os cromossomos não adaptados antes de desaparecerem devam ser todos submetidos à ação do operador genético de mutação, como uma forma de garantir a presença das características adaptadas ao ambiente destes organismos nas próximas gerações.

Também faz parte da metodologia o registro da história genealógica (táxon) dos cromossomos da população atual, para que uma explicação sobre o resultado encontrado para um dado problema possa ser gerada sempre que necessário. Um exemplo de problema onde a explicação do resultado encontrado é relevante seria construir um algoritmo genético para exibir os erros dos exercícios de soma de dois termos ($a_4a_3a_2a_1a_0$ e $b_4b_3b_2b_1b_0$) propostos por um sistema tutor a uma criança. O papel do algoritmo genético seria operar os termos da soma de modo a convergir para o resultado declarado pela criança ($c_5c_4c_3c_2c_1c_0$). Ao atingir a população de estagnação, o sistema tutor não estaria interessado nos cromossomos obtidos pelo algoritmo, mas sim na história genealógica destes cromossomos, que poderia dizer, entre outras coisas, que a soma $a_2 + b_2 = c_2$ está errada porque o elemento *vai-um* de $a_1 + b_1$ não foi considerado, ou porque a soma $a_2 + b_2 = c_2$ está incorreta, etc.

A Seção 3.2 deste capítulo descreve os tipos base, gene, cromossomo e população definidos pelo GAADT. A Seção 3.3 define os operadores genéticos que trabalham sobre os tipos definidos na Seção 3.2. A Seção 3.4 apresenta a estrutura do ambiente trabalhado pelo GAADT e a Seção 3.5 traz a definição do algoritmo propriamente dito. Todas estas seções serão apresentadas em termos do problema da descoberta e composição automáticas de Serviços Web Semânticos.

3.2 Tipos Básicos

3.2.1 Base

O GAADT propõe que os cromossomos sejam representados por seu material genético, o qual têm nas bases suas unidades elementares de formação. Assim, uma base *B* é o conjunto de todas as unidades genéticas elementares que podem ser usadas na formação do material genético dos cromossomos de uma população.

Os elementos de uma base se agrupam em sequências para formar as características (genes) dos cromossomos. Mas nem toda sequência de bases representa uma característica para o cromossomo. Portanto, deve existir uma *lei de formação* para indicar como as bases devem ser agrupadas para formar uma dada característica. A *lei de formação* de características é representada pelo conjunto de *Axiomas de Formação de Genes* (*AFG*).

Voltando para o problema dos Serviços Web Semânticos, e considerando nossa representação que utiliza um conjunto de termos de entradas e saídas que se referem a uma ontologia, um SWS, então, seria uma associação entre um conjunto de termos de entradas e um conjunto de termos de saída[1]. Porém, como não é objetivo deste trabalho conceber serviços, e sim utilizar serviços pré-existentes em determinado catálogo, os conceitos de base e *AFG*, definidos pelo GAADT, não se aplicam à resolução deste problema.

3.2.2 Gene

Um gene *g* é uma sequência formada pelos elementos da base que pertencem ao conjunto *AFG*. Os genes são agrupados em conjuntos para formar os cromossomos da população e o conjunto de genes $\{g_1, g_2, \ldots, g_n\}$ que compõe um dado cromossomo serve para identificá-lo dentro da população. A identidade dos cromossomos será usada para impedir que várias cópias de um cromossomo possam coexistir ou renascer na população em qualquer tempo durante o processo de evolução da mesma na busca por um cromossomo mais adaptado. Para os fins deste trabalho, cada SWS presente no catálogo de serviços será considerado um gene.

3.2.3 Cromossomo

Para os cromossomos, de acordo com as características descritas por Vieira(38), deve haver um *relógio biológico*, capaz de garantir que o cruzamento de dois cromossomos de uma mesma espécie resulte em um cromossomo da mesma espécie. O GAADT batiza o *relógio biológico* do cromossomo de *Axioma de Formação de Cromossomos* (*AFC*). E um cromossomo *c* é um conjunto de genes que obedece às condições estabelecidas pelo *AFC*.

[1]Na verdade, o conjunto considerado deveria ser o IOPE - como citado no Capítulo 2 - porém a API utilizada, que será melhor detalhada no Capítulo 4 ainda não provê suporte para trabalhar com definições de pré-condições e efeitos.

3.2.3.1 Axioma de Formação de Cromossomos

Para o problema em questão, o *AFC* será uma função que garanta que todo cromossomo tenha a seguinte forma:

- As entradas do primeiro gene do cromossomo têm correspondência com as entradas de uma requisição;
- As saídas do último gene de um cromossomo têm correspondência com as saídas de uma requisição;
- Todo cromossomo é formado por um ou mais genes. Caso haja mais de um gene, todo gene g_i deve estar ativo, ou seja, suas entradas devem ser fornecidas por alguma das saídas de algum dos genes $g_{i-1}, \ldots, g_1$.

Assim, a Figura 3.1 representa cromossomos válidos:

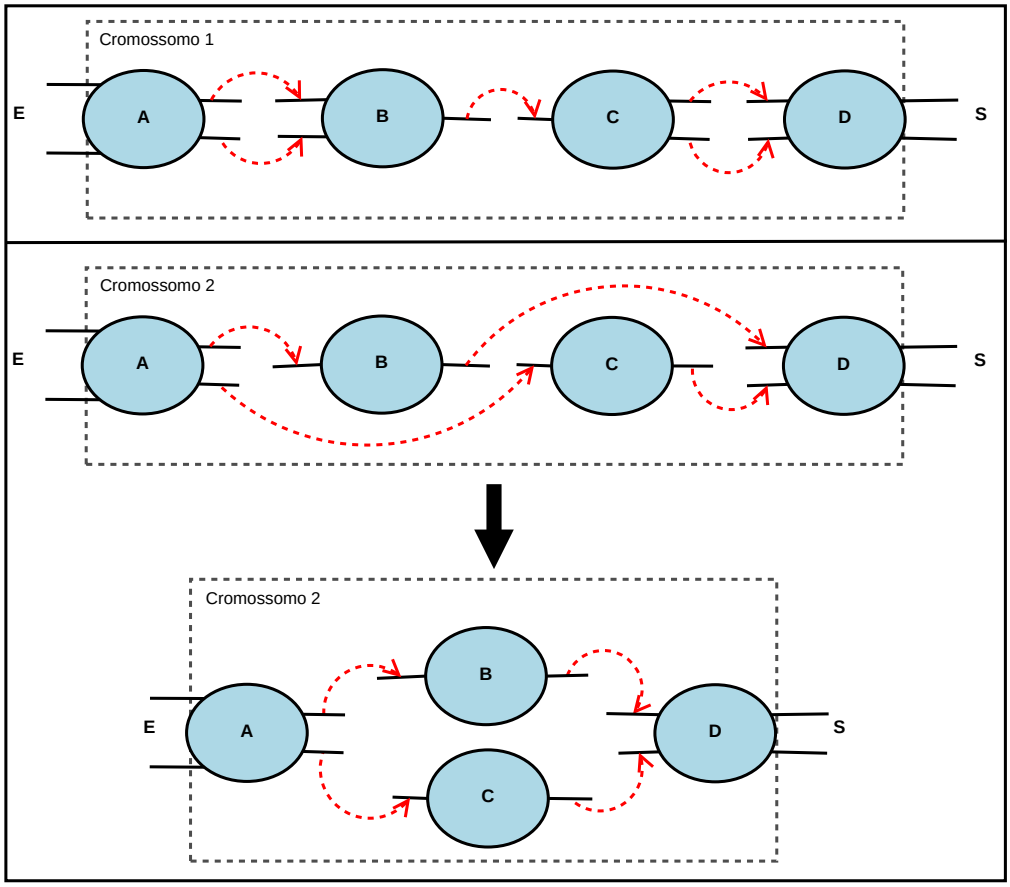

Figura 3.1 Exemplos de cromossomos válidos formados pelo AFC.

3.2.4 População

Os cromossomos são agrupados em conjuntos para formar uma população. Esta representação para a população irá garantir a imparcialidade na avaliação dos cromossomos que compõem a população, já que cada cromossomo só poderá ocorrer uma vez na população.

Seja uma população P_j um conjunto de cromossomos construídos conforme descrito no *AFC*, o tipo população é o conjunto formado por todos os conjuntos formados por objetos do tipo cromossomo (ou seja, $P = \mathbb{P}(\mathbb{P}(C))$). Estes são possíveis resultados para o problema em foco segundo a interpretação adotada para os tipos *C*, *G* e *B*.

3.3 Operadores Genéticos

O GAADT trabalha com dois tipos de operadores genéticos: o de reprodução e o de mutação. O operador genético de reprodução caracteriza-se por combinar os genes de dois cromossomos (cromossomos pais) para formar outros cromossomos (cromossomos filhos) enquanto o operador genético de mutação caracteriza-se por alterar a identidade de um cromossomo para formar um outro cromossomo (cromossomo-mutante).

O gene dos cromossomos pais para uma dada característica que fará parte dos cromossomos filhos é aquele que melhor satisfaz as restrições do problema sobre a característica expressa por este gene, o qual será denominado de gene-dominante. Dados dois genes g_1 e g_2, que expressem uma mesma característica com diferentes fenótipos, diz-se que um gene g_1 melhor satisfaz as restrições do problema em relação ao gene g_2 se o grau de adaptação do gene g_1 for superior ou igual ao grau de adaptação do gene g_2.

Neste trabalho, o grau de adaptação de um gene é dado por uma função *grau* que calcula a similaridade, usando a regra do cosseno, entre o conceito utilizado para definir uma entrada ou uma saída de um serviço web semântico e o conceito utilizado para definir uma entrada ou saída de uma requisição. Maiores detalhes quanto a Recuperação de Informação no Modelo Vetorial - modelo que enxerga uma consulta e uma resposta como vetores e através de regras como a do cosseno consegue atribuir um grau de similaridade entre os mesmos - podem ser encontradas no Apêndice B.

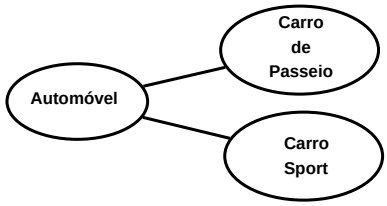

Figura 3.2 Exemplo de dois conceitos e um ancestral comum em uma ontologia.

A Figura 3.2 mostra a visualização de dois termos de uma ontologia e um ancestral comum entre eles. O cálculo da similaridade entre esses dois conceitos seria dado de acordo com a

fórmula abaixo, simplificação da equação (B.1):

$$cos\theta = \frac{t_1 \circ t_2}{|| t_1 || . || t_2 ||} \tag{3.1}$$

Assim, a similaridade entre os conceitos *Carro de Passeio* e *Carro Sport*, na ontologia dada, seria de 0.5, ou 50%.

Sejam e uma entrada de um gene, s uma saída, n_e o número de entradas, n_s o número de saídas e *Sim* a função *Similaridade* que implementa a regra descrita na equação acima, a função *adaptGene*, que informa o grau de adaptação de um gene em relação à requisição é expressa pela média dos somatórios das similaridades individuais de cada termo, como segue:

$$simEntrGene = \frac{\sum_{i=1}^{n_e} Sim(e_i)}{n_e} \tag{3.2}$$

$$simSaidaGene = \frac{\sum_{j=1}^{n_s} Sim(s_j)}{n_s} \tag{3.3}$$

$$adaptGene = \frac{simEntrGene + simSaidaGene}{2} \tag{3.4}$$

A produção de novos cromossomos durante o processo evolutivo de uma população serve para direcionar a busca por cromossomos mais adaptados através da transmissão das características de maior grau de adaptação presentes nos cromossomos da população atual. A adaptação de um cromossomo é dada pela função *adapt*, que para o problema deste trabalho, é dada pela média entre a adaptação da entrada do primeiro gene do cromossomo e a adaptação da saída do último gene, expressa por:

$$adapt = \frac{simEntrGene(g_1) + simSaidaGene(g_n)}{2} \tag{3.5}$$

3.3.1 Seleção

No GAADT, a função de seleção recebe uma população P_1 e retorna a subpopulação de P_1 formada pelos cromossomos que satisfazem um requisito do problema r, descrito por uma fórmula em lógica de primeira ordem, o qual indica quando um dado cromossomo é considerado apto a cruzar.

Para o problema da descoberta de composição de SWS, a função de seleção não faz distinção entre os cromossomos de uma população e os considera todos aptos a cruzar. Ressalta-se, apenas, que o *Axioma de Formação de Cromossomo* nem sempre vai permitir que dois cromossomos selecionados para cruzamento gerem cromossomos descendentes, visto que todas as suas restrições axiomáticas devem ser cumpridas.

Ainda, a reprodução no caso do problema deste trabalho é assexuada, ou seja, não há distinção entre cromossomos macho e fêmea. Tal distinção é prevista na função de fecundação do GAADT, responsável por discernir ambas as classes de cromossomos.

3.3.2 Cruzamento

O operador definido no GAADT, e utilizado em conjunto com o *AFC*, garante que o cruzamento entre dois cromossomos vai sempre gerar um cromossomo válido. Neste trabalho, o operador de cruzamento pode gerar até 8 cromossomos descendentes. Para melhor visualização, considere dois cromossomos candidatos a cruzamento A e B:

1. Se ambos os cromossomos possuem apenas um gene, são gerados filhos AB e BA, tal como mostrado na Figura 3.3;

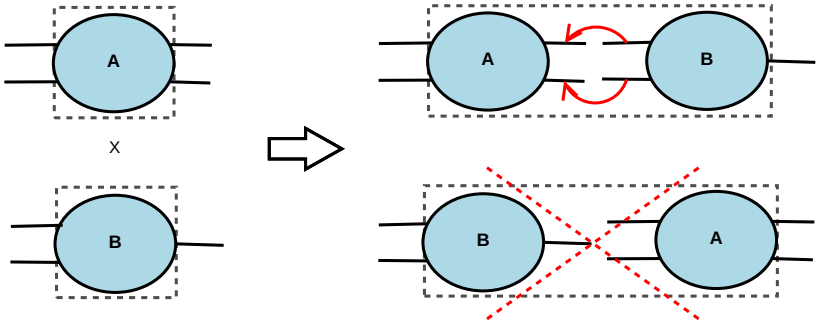

Figura 3.3 Exemplo de cruzamento entre dois cromossomos com um só gene onde apenas o primeiro descendente obedeceu ao *AFC*.

2. Se há um cromossomo com tamanho maior que 1, então verifica-se a possibilidade de divisão do cromossomo na metade de seu comprimento.

    - A divisão será considerada bem-sucedida se o genes posteriores ao gene g_n, presente no meio da sequência de genes e candidato a gene inicial da nova sequência, não dependerem de algum gene g_{n-1}. A Figura 3.4 representa uma divisão bem-sucedida de um cromossomo no gene C, pois o gene D depende apenas do gene C, que estará presente no cromossomo formado pela parte 2 do cromossomo original;
    - Em divisões bem-sucedidas, as partes A_1A_2 e B_1B_2 são combinadas, gerando os filhos A_1B_1, A_1B_2, A_2B_1, A_2B_2, B_1A_1, B_1A_2, B_2A_1 e B_2A_2 quando estes obedecerem todos ao *AFC*;
    - Se a divisão não for possível, então o cromossomo não será dividido e o cruzamento será dado como no item 1.

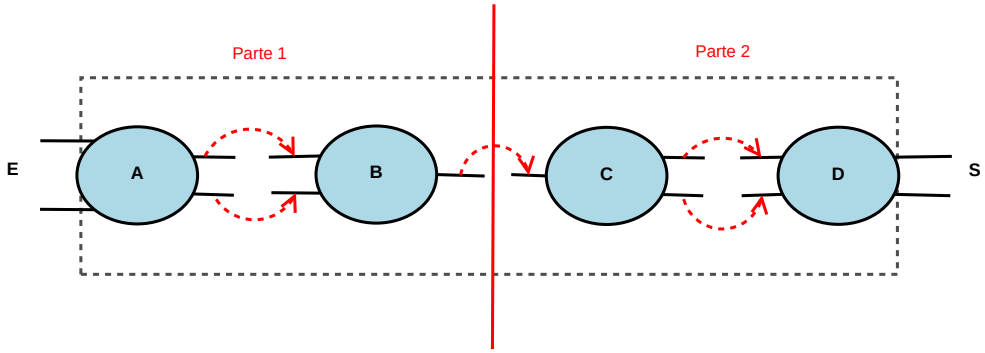

Figura 3.4 Exemplo de divisão possível de cromossomo com mais de um gene.

3.3.3 Mutação

O operador genético de mutação, definido para o GAADT, é composto pelas funções de inserção, supressão e troca, tal que os cromossomos resultantes da ação destes operadores apresentarão parte dos genes contidos no cromossomo que lhe deu origem, como segue:

- A operação de inserção *ins* adiciona um conjunto de genes ao cromossomo de origem;
- A operação de supressão *del* remove um conjunto de genes do cromossomo de origem;
- A operação de troca *troc* remove um conjunto de genes do cromossomo de origem e lhe adiciona outro conjunto de genes.
  - As ações da função de inserção e supressão podem ser vistas como casos particulares da ação da função de troca.

O operador de mutação do GAADT garante que um indivíduo mutante é mais adaptado que o indivíduo que o originou. Devido às relações de dependência entre genes de um cromossomo, particulares ao problema que este trabalho trata, considera-se probabilidade de criar uma composição inválida de Serviços Web Semânticos maior que a probabilidade de se criar uma composição válida através das funções de inserção, supressão e troca definidas neste operador. Portanto, as mesmas não foram implementadas.

3.4 Ambiente

Um algoritmo genético opera sobre populações de cromossomos que evoluem de acordo com as características de um ambiente A. Um ambiente A é uma 8-tupla $\langle P, \mathbb{P}(P), Rq, AFG, AGC, Tx, \Sigma, P_0 \rangle$, onde:

- P é a população;

- $\mathbb{P}(P)$ é o conjunto potência de P;
- *Rq* é o conjunto dos requisitos do problema, ou seja, os parâmetros de entrada e saída do serviço que se deseja encontrar de forma unitária ou através de composição e que influenciam a genealogia da população P;
- *AFG* é o conjunto de axiomas de formação dos genes dos cromossomos da população P, vazio para o problema deste trabalho;
- *AFC* é o conjunto de axiomas de formação dos cromossomos da população P;
- *Tx* é o conjunto de pares de cromossomos (x,y), onde x é um cromossomo construído a partir do cromossomo y, pela ação da operação de cruzamento ou mutação, registrando desta forma a genealogia dos cromossomos pertencentes às populações geradas pelo GA-ADT durante a sua execução;
- Σ é o conjunto de operadores genealógicos que atuam sobre a população P;
- P_0 é uma sub-população pertencente a $\mathbb{P}(P)$, chamada de população inicial, que neste caso será o conjunto de todos os cromossomos que representam, individualmente, os Serviços Web Semânticos providos pelo catálogo.

3.5 Algoritmo

O GAADT é uma função *GAADT* que recebe a população P_0 e, depois de submetê-la à simulação de um processo evolutivo, devolve uma população P_t. Os cromossomos da população P_t são os cromossomos das populações P_0,P_1,...,P_{t-1} que ainda satisfazem os requisitos do problema *Rq*, ou então são novos cromossomos resultantes da ação genealógica das operações de cruzamento e mutação sobre os cromossomos da população P_{t-1} que apresentam adaptação maior do que a adaptação dos cromossomos que lhes deram origem. Diz-se então que a população P_t evoluiu da população P_0.

Os cromossomos das populações P_0,P_1,...,P_{t-1} que não mais satisfaçam os requisitos do problema *Rq* não participarão da construção da população P_t, podendo ser assim entendidos como fazendo parte da população de cromossomos "mortos", que não figurarão entre os cromossomos da população P_t e das populações seguintes manipuladas pela função *GAADT*. Não obstante, tais cromossomos serão recuperados pela análise da taxonomia *Tx* dos cromossomos da população atual para evitar que eles apareçam novamente nas próximas iterações da função *GAADT*. Esta restrição atende ao entendimento do processo de evolução darwinista, que não contempla a possibilidade de uma espécie extinta voltar a aparecer num outro momento futuro.

Antes de ser apresentada uma definição para a função *GAADT*, deve-se observar a necessidade de se estabelecer um critério de preservação sobre a população atual P_t, para orientar o corte dos cromossomos que não devem figurar nas populações P_{t+1},P_{t+2},.... Na definição da

função *GAADT*, este ponto de corte será representado por um predicado unário p_{corte} pertencente ao conjunto de requisitos do problema *Rq*, que atua sobre os cromossomos de P_t.

Para o problema da descoberta e composição de SWS, o ponto de corte será fixado no tamanho máximo da população, que está estabelecido em duas vezes o tamanho da população inicial. Assim, até que se atinga o tamanho máximo da população, serão descartados os indivíduos compostos de mais de um gene e que possuem baixo grau de adaptação. É importante salientar que os indivíduos de um gene apenas são os presentes da população individual e não devem ser descartados, pois, apesar de não terem grau relevante de similaridade com a requisição, podem atuar como ponte entre um serviço que atenda aos parâmetros de entrada da requisição e um que atenda aos parâmetros de saída da mesma.

A adaptação média da população é obtida dividindo-se a soma da adaptação de todos os cromossomos da população pelo número de cromossomos desta população e os critérios de parada adotados pela função *GAADT* são: a) tempo máximo, aqui estipulado em 30s, de execução do algoritmo, b) valor da adaptação dos cromossomos considerado satisfatório para o resultado do problema em análise, aqui estipulado em 90%. Estes critérios também fazem parte do conjunto de requisitos do problema *Rq*.

CAPÍTULO 4

Solução Proposta: Arquitetura, Aspectos de Implementação e Resultados

Todos veem o que pareces, poucos percebem o que és.
— NICCOLO MAQUIAVEL (O Príncipe)

Man has qualities which can never be replaced by a robot...
They (robots) are there so that he shall have more time for the truly human tasks - those of creation.
— KARLHEINZ STOCKHAUSEN

4.1 Introdução

Em Shadbolt, Berners-Lee e Hall(2), os autores afirmaram que a Web Semântica permaneceria, basicamente, no plano das ideias até que padrões de representação de conhecimento e protocolos de comunicação fossem bem estabelecidos e acordados. Eles ressaltaram que, a exemplo do protocolo HTTP, cujo uso pioneiro pela comunidade da área da física abriu caminho para o sucesso da Web, o uso crescente de ontologias pela comunidade científica eletrônica ("e-science") também poderia levar a Web Semântica ao sucesso tal como hoje conhecemos a Web. Segundo os autores, além de fatores sociais e de decisões de projeto, parte do sucesso da Web Semântica reside na sequência de especificações (*Universal Resource Identifier* - URI, HTTP, RDF, ontologias, etc.) e de registros (esquema URI, conteúdos de Internet do tipo *Multipurpose Internet Mail Extensions* - MIME ou extensões multi-função para mensagens de Internet), os quais fornecem meios para que construções como uma ontologia derivem significado de um identificador URI.

Atualmente já há diversas tecnologias consolidadas e recomendadas por instituições tais como o W3C[1] a exemplo do WSDL, OWL e OWL-S, que provêm suporte para a construção

[1]http://www.w3.org/ - Segunda sua própria descrição, a entidade se define como um fórum para informação, comércio, comunicação e entendimento coletivo que desenvolve tecnologias interoperáveis (especificações, linhas gerais, *sofware* e ferramentas) na condução da Web ao seu potencial pleno.

das chamadas *Aplicações Semânticas* - termo que caracteriza aplicações que tratam de conteúdo semântico, tal como descrito por Santos e Carvalho(39).

A Seção 4.2 deste capítulo trata das principais tecnologias utilizadas na implementação da ferramenta proposta, são elas: OWL, OWL-S, OWL-S API e Jena. A Seção 4.3 traz detalhes quanto à implementação e a Seção 4.4 traz os experimentos e resultados obtidos.

4.2 Arquitetura e Ferramentas Utilizadas

Abaixo segue uma figura descrevendo a arquitetura e as tecnologias, bem como ferramentas, utilizadas na solução implementada:

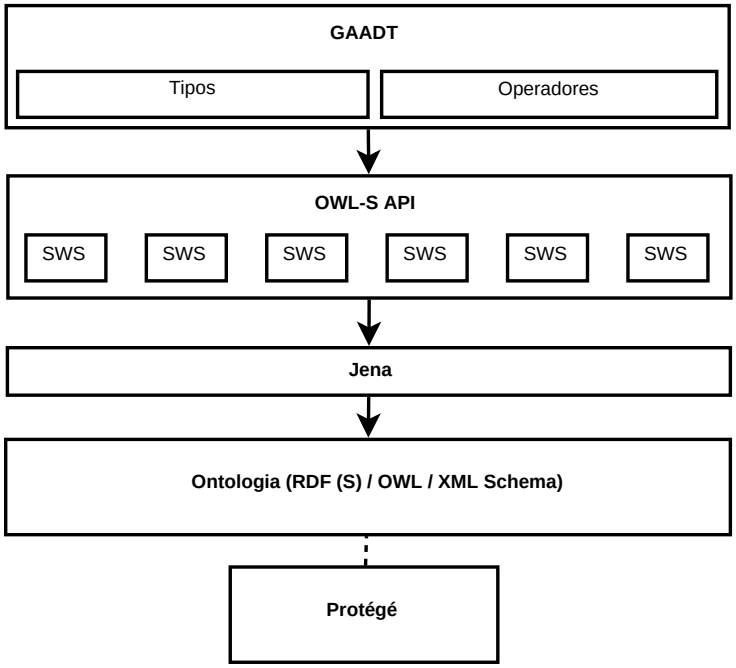

Figura 4.1 Arquitetura dos Componentes de Software Utilizados na Solução Implementada

1. *Protégé* [2]: Ferramenta para manipulação de ontologias descritas em RDF(S), OWL e XML *Schemas*. É utilizada diretamente na composição das ontologias e regras de inferência associadas.

2. *Ontologia*: Estrutura hierárquica utilizada para armazenar os conceitos referidos pelos SWS.

3. *Jena*: Framework desenvolvido pelo *HP Labs Semantic Web Research Group*[3] para a

[2]http://protege.stanford.edu
[3]http://www.hpl.hp.com/semweb

construção de aplicações semânticas. Capaz de interpretar as ontologias referenciadas em diversas linguagens de representação e dotado também de motores de inferência.

4. *OWL-S API* [4]: API que estende o Jena e fornece acesso à leitura, execução e gravação de serviços atômicos e compostos descritos em OWL-S.

5. *GAADT*: Algoritmo Genético Baseado em Tipos Abstratos de Dados. Utilizado para manipular e arranjar os serviços criados com a OWL-S API na busca de composições que atendam a determinada requisição de acordo com critérios pré-estabelecidos. Seus tipos e operadores específicos atuam nos serviços da camada anterior de forma a orquestrar os mesmos para a descoberta e composição.

4.2.1 OWL - *Web Ontology Language*

Martino(40) afirma que em termos computacionais, o que se espera de uma ontologia é uma especificação explícita e formal que conceitua um modelo abstrato de algum fenômeno do mundo em um conhecimento consensual, isto é, compartilhado por todos. Além disso, os conceitos, as propriedades, as funções e os axiomas devem ser especificados explicitamente e serem manipuláveis por computador. As ontologias devem ter capacidade de identificar contextos de um termo, compartilhar definições e dar suporte ao reuso. Quando elas são construídas levando-se em consideração esses aspectos, é possível ajudar as pessoas na busca, extração, interpretação e processamento da informação.

Para McIlraith e Martin(41), na construção de ontologias é fundamental uma linguagem com semântica bem definida e expressiva o suficiente para descrever inter-relacionamentos complexos e restrições entre objetos. A recomendação dada pelo W3C à linguagem OWL a constata como uma linguagem que atinge esses requisitos, cujo padrão é apresentado por McGuinness, Welty e Smith(42) e que deriva de um consenso entre duas propostas, a europeia OIL (Ontology Inference Layer) e a DAML (DARPA Agent Markup Language). A OWL é uma linguagem para definição e instanciação de ontologias Web cuja ideia central é permitir a representação eficiente de ontologias. Além disso, ela permite verificar uma ontologia para determinar se sua lógica é consistente ou se há algum conceito falho. Uma ontologia, em OWL, pode incluir:

- Relações de taxonomia entre classes;
- Propriedades dos tipos de dados e descrições dos atributos de elementos das classes;
- Propriedades do objeto e descrições das relações entre elementos das classes;
- Instâncias das classes e instâncias das propriedades.

A recomendação do W3C possui três versões da OWL, que depende do poder de expressividade requerido. As versões menos expressivas (*OWL Lite* e *DL*) estão contidas dentro das

[4] http://on.cs.unibas.ch/owls-api

mais expressivas (*OWL DL* e *Full*). Uma ontologia em uma linguagem menos expressiva é aceita por uma linguagem mais expressiva, contudo a recíproca não é aceita. A adoção da linguagem OWL vem aumentando a cada dia, no entanto, ainda se encontram iniciativas paralelas tais como o formato OBO[5], que tendem a desaparecer dada a quantidade limitada de ferramentas capazes de manipular tais objetos e também às ferramentas de conversão, cuja missão é converter ontologias de, e principalmente para o formato recomendado, ou seja, OWL.

A Figura 4.2 mostra o código-fonte de uma pequena ontologia descrita em OWL:

```
<rdfs:Class rdf:ID="MEAL-COURSE">
   <rdfs:subClassOf rdf:resource="#CONSUMABLE-THING"/>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:minCardinality>
            1
         </daml:minCardinality>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:toClass rdf:resource="#EDIBLE-THING"/>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#DRINK"/>
         <daml:minCardinality>
            1
         </daml:minCardinality>
      </daml:Restriction>
   </rdfs:subClassOf>
</rdfs:Class>
```

Figura 4.2 Exemplo de ontologia descrita em OWL.

4.2.2 OWL-S - *Ontology Web Language for Services*

OWL-S é uma ontologia para conceitos de serviços que fornecem um conjunto de construções, em linguagem de marcação, para descrever as propriedades e capacidades de um serviço de forma não-ambígua e interpretável por computador. A versão atual da OWL-S acompanha a versão recomendada pelo W3C da OWL, produzida pelo *Ontology Working Group* dessa instituição.

A OWL-S permite às ontologias descrever, por um lado, os conceitos dos domínios dos serviços (passagens aéreas, hotéis, turismo, comércio eletrônico e etc.), e por outro lado, conceitos genéricos que descrevem os próprios serviços (tais como controle de fluxo e controle de dados) e como esses se relacionam com as ontologias de domínio - através das entradas, saídas, precondições, efeitos e etc. Estas descrições semanticamente ricas habilitam a inferência

[5] *Open Biomedical Ontologies* - disponível em http://www.obofoundry.org

automática por máquinas sobre as descrições dos serviços e domínios, dando, assim, suporte à automação da descoberta, composição e execução de serviços e reduzindo tanto a configuração manual quanto maiores esforços de programação.

A OWL-S organiza a descrição de um serviço em quatro áreas conceituais, a saber:

- *Process*: Descreve como um serviço executa suas tarefas. Ele inclui informação sobre as entradas, saídas - incluindo uma especificação das condições nas quais várias saídas serão produzidas -, pré-condições - circunstâncias que devem ser satisfeitas antes que um serviço possa ser invocado - e resultados - mudanças realizadas por um serviço. O *Process Model* diferencia processos compostos, simples e atômicos. Em um processo composto, o modelo mostra como o serviço se divide em componentes de processo mais simples, e o fluxo de controle e dados entre esses componentes. Processos atômicos são essencialmente "caixas pretas" de funcionalidade, e processos simples são descrições abstratas de processos que podem se relacionar com outros processos compostos ou atômicos.

- *Profile*: Provê uma descrição geral do SWS de forma a facilitar a descoberta do serviço quando o mesmo é publicado e compartilhado. Esta descrição pode incluir ambas as propriedades funcionais (entradas, saídas, pré-condições e resultados) e não-funcionais (nome do serviço, descrição textual, informações de contato, categoria do serviço e parâmetros adicionais do serviço). As propriedades funcionais são derivadas do *Process Model*, mas não é necessário incluir todas as propriedades funcionais do Process Model no Profile. Uma visão simplificada pode ser fornecida para a descoberta de serviços, no pressuposto de que um consumidor de serviço olhará, eventualmente, para o *Process Model* para alcançar um entendimento completo sobre como o serviço funciona.

- *Grounding*: Especifica como um serviço é invocado através do detalhamento de como os processos atômicos do *Process Model* de um serviço mapeam um protocolo real de troca de mensagens. A OWL-S fornece tipos diferentes de *groundings*, no entanto, o único tipo desenvolvido até o momento é o WSDL[6] *grounding*, que permite que qualquer Serviço Web seja anotado como um Serviço Web Semântico usando OWL-S.

- *Service*: Um *service* simplesmente reúne todas as partes em uma unidade que pode ser publicada e invocada. É importante compreender que as diferentes partes de um serviço podem ser reutilizadas e conectadas de várias formas. Por exemplo, um provedor de serviço pode conectar seu *Proces Model* com vários *Profiles* de forma a fornecer propagandas direcionadas a diferentes comunidades de consumidores de serviço. Um provedor de serviço diferente, provendo um serviço similar, pode reutilizar o mesmo *Process Model*, possivelmente como parte de um processo composto maior, e conectá-lo a um *Grounding* diferente. As relações entre os componentes de um *Service* são modeladas utilizando propriedades tais como:

  – *presents*: Service - Profile;

[6] O WSDL é apresentado por Christensen et al.(6)

– *describedBy*: Service - Process Model;
– *supports*: Service - Grounding.

A Figura 4.3 mostra um exemplo de serviço descrito em OWL-S.

4.2.3 OWL-S API

Trata-se de uma API Java para acesso programático à criação, leitura, escrita e execução de serviços atômicos ou compostos descritos em OWL-S. A biblioteca fornece um motor de execução que pode executar processos atômicos (serviços atômicos) usando *Groundings* WSDL, Java ou UPnP, e processos compostos (serviços compostos) que utilizam construções de controle tais como *Choice*, *Sequence*, *AnyOrder*, *Split* e *Split-Join*. A execução de processos que consistem de construções de controle condicionais tais como *IfThenElse*, *RepeatUntil* ou *RepeatWhile* também é suportada. Além disso, também é fornecido um monitor de execução, responsável pela monitoria dos serviços envolvidos numa execução.

Ainda, são suportadas mais linguagens para expressar condições de controle e precondições de processo tais como SWRL[7] e SPARQL[8]. A API é projetada para ser extensível, de forma que formalismos(lógicos), assim como seus procedimentos de avaliação, podem ser utilizados. A API é distribuída em conjunto com um motor de inferência para OWL-DL chamado Pellet[9], porém esta também é uma parte extensível e pode ser substituída por outros motores de inferência (RDFS[10] por exemplo).

A Figura 4.4 mostra um pequeno exemplo de leitura de um SWS definido em OWL-S.

4.2.4 Jena

Jena é um arcabouço Java para a construção de aplicações semânticas através da manipulação de ontologias. Ele provê um ambiente programático para ontologias descritas em RDF, RDFS e OWL, SPARQL e inclui um motor de inferência baseado em regras.

Jena é de código livre e surgiu no HP Labs Semantic Web Programme. Como um arcabouço, ele inclui:

- Uma API para manipulação de arquivos RDF;
- Leitura e escrita de ontologia descritas em RDF em diversos formatos (incluindo OWL);
- Opções de armazenamento persistente ou em memória;
- Motor de execução de consultas SPARQL.

[7] *Semantic Web Rule Language* - http://www.w3.org/Submission/SWRL/
[8] *SPARQL Query Language for RDF* - http://www.w3.org/TR/rdf-sparql-query/
[9] http://clarkparsia.com/pellet
[10] http://jena.sourceforge.net/inference

```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF
 xmlns:owl = "http://www.w3.org/2002/07/owl#"
 xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
 xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:service = "http://www.daml.org/services/owl-s/1.1/Service.owl#"
 xmlns:process = "http://www.daml.org/services/owl-s/1.1/Process.owl#"
 xmlns:profile = "http://www.daml.org/services/owl-s/1.1/Profile.owl#"
 xmlns:grounding = "http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
 xmlns:expr = "http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#"
 xmlns:swrl = "http://www.w3.org/2003/11/swrl#"
 xml:base = "http://127.0.0.1/services/ISBNFinder.owl">

<owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://127.0.0.1/ontology/Concepts.owl"/>
</owl:Ontology>

<service:Service rdf:ID="ISBNFinder">
    <service:presents rdf:resource="#ISBNFinder-Profile"/>
    <service:describedBy rdf:resource="#ISBNFinder-Process-Model"/>
    <service:supports rdf:resource="#ISBNFinder-Grounding"/>
</service:Service>

<profile:Profile rdf:ID="ISBNFinder-Profile">
    <service:isPresentedBy rdf:resource="#ISBNFinder-Service"/>
         <profile:serviceName xml:lang="en">
           ISBN Finder
    </profile:serviceName>
    <profile:textDescription xml:lang="en">
           Receives a book and finds its corresponding ISBN.
    </profile:textDescription>
    <profile:hasInput rdf:resource="#Book"/>
    <profile:hasOutput rdf:resource="#ISBN"/>
</profile:Profile>

<process:ProcessModel rdf:ID="ISBNFinder-Process-Model">
     <service:describes rdf:resource="#ISBNFinder-Service"/>
     <process:hasProcess rdf:resource="#ISBNFinder-Process"/>
</process:ProcessModel>

<process:AtomicProcess rdf:ID="ISBNFinder-Process">
   <process:hasInput rdf:resource="#Book"/>
   <process:hasOutput rdf:resource="#ISBN"/>
</process:AtomicProcess>

<process:Input rdf:ID="Book">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
                http://127.0.0.1/ontology/Concepts.owl#Book
    </process:parameterType>
    <rdfs:label>Book Information</rdfs:label>
</process:Input>

<process:Output rdf:ID="ISBN">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
                http://127.0.0.1/ontology/Concepts.owl#ISBN
    </process:parameterType>
    <rdfs:label>ISBN Information</rdfs:label>
</process:Output>

<grounding:WsdlGrounding rdf:ID="ISBNFinder-Grounding">
    <service:supportedBy rdf:resource="#ISBNFinder-Service"/>
</grounding:WsdlGrounding>

</rdf:RDF>
```

Figura 4.3 Exemplo de SWS descrito em OWL-S.

```
// Cria um motor de execução
ProcessExecutionEngine exec = OWLSFactory.createExecutionEngine();

// Cria uma base de conhecimento
OWLKnowledgeBase kb = OWLFactory.createKB();

// lê a descrição de um serviço
Service aService = kb.readService("http://www.mindswap.org/2004/owl-s/1.1/Dictionary.owl");
// get the process for the server
Process aProcess = aService.getProcess();

// inicializa o mapeamento de entradas com o valor vazio
ValueMap aInputValueMap = new ValueMap();

// especifica um valor de entrada
String inValue = "hello";
// adiciona o valor ao mapeamento
aInputValueMap.setDataValue(aProcess.getInput("InputString"), inValue);

// executa o processo
ValueMap aOutputValueMap = exec.execute(aProcess, aInputValueMap);

// recebe a saída
OWLDataValue out = (OWLDataValue) aOutputValueMap.getValue(aProcess.getOutput());
```

Figura 4.4 Exemplo código para execução de um SWS utilizando a OWL-S API. Fonte: http://www.mindswap.org/2004/owl-s/api

4.3 Aspectos de Implementação e Dificuldades Encontradas

Esta seção apresenta a construção da aplicação com a estrutura evidenciada na Figura 4.1. O software que implementa a modelagem proposta neste trabalho foi construído em Java[11] no ambiente Eclipse[12].

Quanto ao conjunto de serviços, como não é objetivo deste trabalho criar Serviços Web Semânticos e sim descobri-los e compô-los quando necessário, optou-se por utilizar um conjunto de serviços de testes disponíveis na Web. Algumas propriedades e requisitos inerentes à utilização dos arcabouços e tecnologias descritas na seção anterior, bem como restrições para o desenvolvimento da aplicação, foram consideradas como segue:

- A utilização do Jena na manipulação de ontologias demanda o carregamento de todas as ontologias utilizadas nas descrições dos SWS utilizados. Isto implica que o motor de inferência utilizado pelo Jena, ao ser carregado, tentará inferir novas regras e ligações entre termos das ontologias utilizadas. Se o número de ontologias foi muito grande, este processo poderá demorar um tempo demasiadamente grande, o que causaria atraso e lentidão não na execução da aplicação em sua versão final (pois a ontologia já estaria carregada em memória), mas durante sua fase de desenvolvimento e testes.
- A possibilidade de indisponibilidade de serviços em tempo de execução na Web (pro-

[11] http://java.sun.com
[12] www.eclipse.org

blemas no servidor) era um risco no desenvolvimento da aplicação. Para mitigá-lo, era necessário que o conjunto de serviços fosse utilizável a partir do próprio computador, sem que para isso fosse necessária uma conexão externa à Web.

Dadas as características do desenvolvimento e suas restrições, optou-se por utilizar dois conjuntos de teste, o OWL-S TC[13] que é um conjunto de cerca de 1200 Serviços Web Semânticos de 10 domínios diferentes (médico, transportes, turismo, armamentos, biologia etc.) descritos em OWL-S, que são entregues juntamente com suas ontologias de domínio e demais regras de inferência SWRL; o segundo conjunto de testes foi o SWS TC[14], que é um conjunto de aproximadamente 240 serviços de domínios diferentes e entregues com o mesmo conteúdo que o pacote anterior.

Para a execução local dos serviços, foi utilizado o servidor Web XAMPP[15], que é de instalação simplificada e uso imediato. Porém, por conter 34 ontologias de tamanho considerável (algumas com mais de 500KB) que juntas somam mais de 8000 termos, o primeiro conjunto levou mais de 1 hora (utilizando computador *laptop* pessoal de configuração mediana) para ser lido e interpretado, o que tornava sua utilização, dados os critérios estabelecidos acima, impraticável. Assim, optou-se pela utilização do segundo conjunto de serviços de testes (241 serviços e ontologia com aproximadamente 600 termos), com redução considerável do tempo de carga para pouco mais de 10 segundos, no pior caso.

Ressalta-se que a escolha se deu por razões de performance da execução dos testes apenas, onde não se julgou que a aplicação deixaria de provar seu conceito. E, justamente por ser prova de conceito, não foi construída com fins de performance, embora se reconheçam inúmeras oportunidades de melhoria do código produzido em termos de otimização de execução.

4.4 Resultados Obtidos

4.4.1 Descrição dos Testes

A aplicação foi projetada para receber as URI dos termos de entrada e saída desejados. Assim, para o teste, foram idealizados quatro respostas possíveis para a execução:

1. Um *match* direto[16], com grau de adaptação de 100% entre requisição e resposta;
2. Um *match* indireto, com composição de serviços necessária.
3. Um *match* indireto, com *split* de serviços presente.

[13]Disponível em http://projects.semwebcentral.org/projects/owls-tc
[14]disponível em http://projects.semwebcentral.org/projects/sws-tc
[15]Disponível em http://www.apachefriends.org/pt_br/xampp.html
[16]Considera-se *match* direto uma requisição que tenha sido atendida por um serviço unitário, e *match* indireto uma resposta a uma requisição que envolva mais de um SWS (composição).

Para a execução do teste 1, foram utilizados os termos que descrevem o serviço "*Babylon Dictionary*", que são "*Word*" para a entrada e "*Definition*" para a saída. O resultado da execução se encontra a seguir:

4.4.2 Execuções

Resultado 1:

```
Criando base de Conhecimento... Ok.
Adicionando serviços à base de conhecimento... 241 serviços adicionados em 6.0 segundos.
Geração Inicial
    Tamanho da população inicial: 215
    Adaptação da população inicial: 33.48%
Execução:

Resposta Encontrada:
    Grau de adaptação da resposta: 100.0%
    Serviços:
        1: Babylon Dictionary | http://127.0.0.1/services/BabylonDictionary.owl#BabylonDictionary

Tempo total de execução: 0.0 segundos.
```

Segundo a modelagem dada no Capítulo 3, a população inicial é composta apenas por cromossomos formados por um único gene, e estes cromossomos encapsulam, individualmente, os serviços originalmente fornecidos pelo repositório. Como cada gene já nasce sabendo o seu grau de adaptação em relação à requisição, não foi necessário que o algoritmo genético operasse sobre a população, uma vez que a mesma já possuía um indivíduo com grau de adaptação maior que os 90% que fazem parte do critério de parada do mesmo.

O segundo teste também foi considerado satisfatório, pois apesar de as entradas descreverem uma composição entre os serviços "*Number2Words*" e "*Spelling Suggestor*", descritos em termos de "*Number*" e "*Text*" como entrada e saída do primeiro serviço, e "*Text*" e "*SpellingSuggestion*" como entrada e saída do segundo serviço, com *match* pretendido(e possível) de 100%, várias respostas foram encontradas além do resultado esperado.

Resultado 2:

```
...
    Geração: 3
        Cruzamento -> Quantidade de novos indivíduos: 5
        Seleção -> Quantidade de indivíduos removidos: 0
        Tamanho da população: 230
        Adaptação da população: 19.9%
        Indivíduo mais adaptado: 70.05%

    Geração: 4
        Cruzamento -> Quantidade de novos indivíduos: 3
        Seleção -> Quantidade de indivíduos removidos: 0
        Tamanho da população: 233
        Adaptação da população: 20.29%
        Indivíduo mais adaptado: 100.0%

Resposta Encontrada:
    Grau de adaptação da resposta: 100.0%
```

```
    Serviços:
        1: Number2Words | http://127.0.0.1/services/Number2Words.owl#Number2Words
        2: Spelling Suggestor | http://127.0.0.1/services/SpellingSuggestor.owl#SpellingSuggestor
              Entrada: http://127.0.0.1/ontology/Concepts.owl#Text
              Saída: http://127.0.0.1/ontology/Concepts.owl#Text
              Serviço: 1

Tempo total de execução: 1.4 segundos.
```

O resultado representa exatamente o que se estava tentando encontrar, ou seja, um *match* indireto com composição possuindo adaptação igual a 100%.

Resultado 3:

```
...
    Geração: 6
        Cruzamento -> Quantidade de novos indivíduos: 1
        Seleção -> Quantidade de indivíduos removidos: 0
        Tamanho da população: 243
        Adaptação da população: 20.06%
        Indivíduo mais adaptado: 70.05%

    Geração: 7
        Cruzamento -> Quantidade de novos indivíduos: 7
        Seleção -> Quantidade de indivíduos removidos: 0
        Tamanho da população: 250
        Adaptação da população: 20.29%
        Indivíduo mais adaptado: 96.77%

Resposta Encontrada:
    Grau de adaptação da resposta: 96.77%
    Serviços:
        1: Number2Words | http://127.0.0.1/services/Number2Words.owl#Number2Words
        2: CDYNE Spell Checker | http://127.0.0.1/services/CDYNESpellChecker.owl#CDYNESpellChecker
              Entrada: http://127.0.0.1/ontology/Concepts.owl#Text
              Saída: http://127.0.0.1/ontology/Concepts.owl#Text
              Serviço: 1

Tempo total de execução: 3.5 segundos.
```

A Figura 4.5 mostra uma representação gráfica do resultado 3.

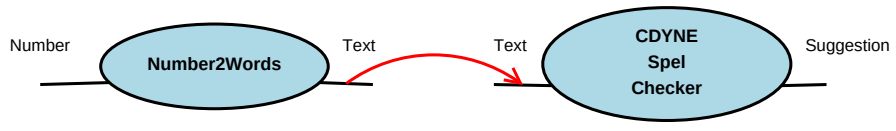

Figura 4.5 Representação gráfica de resultado encontrado na execução da ferramenta.

De fato, o grau de adaptação do cromossomo encontrado (96,77%) é válido, pois os termos *SpellingSuggestion* (informado na requisição) e *Suggestion* (encontrado na resposta) têm bastante proximidade, assim como evidencia a Figura 4.6. Esta composição encontrada representa, ainda, um *match* indireto com grau de adaptação inferior a 100%.

Resultado 4:

```
...
```

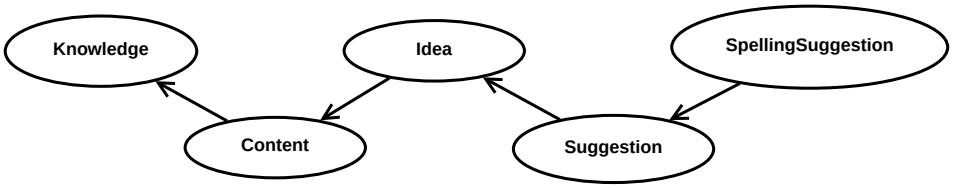

Figura 4.6 Representação gráfica de um ramo da ontologia envolvendo os conceitos *Suggestion* e *SpellingSuggestion*.

```
    Geração: 20
        Cruzamento -> Quantidade de novos indivíduos: 6
        Seleção -> Quantidade de indivíduos removidos: 6
        Tamanho da população: 280
        Adaptação da população: 27.65%
        Indivíduo mais adaptado: 73.14%

    Geração: 21
        Cruzamento -> Quantidade de novos indivíduos: 19
        Seleção -> Quantidade de indivíduos removidos: 19
        Tamanho da população: 280
        Adaptação da população: 28.45%
        Indivíduo mais adaptado: 100.0%

Resposta Encontrada:
    Grau de adaptação da resposta: 100.0%
    Serviços:
        1: Conference Information | http://127.0.0.1/services/ConferenceInformation.owl#
        ConferenceInformation
        2: Get Cheapest Hotel | http://127.0.0.1/services/GetCheapestHotel.owl#GetCheapestHotel
              Entrada: http://127.0.0.1/ontology/Concepts.owl#City | Saída: http://127.0.0.1/
              ontology/Concepts.owl#City | Serviço: 1
        3: SightSeeing Finder | http://127.0.0.1/services/SightSeeingFinder.owl#SightSeeingFinder
              Entrada: http://127.0.0.1/ontology/Concepts.owl#City | Saída: http://127.0.0.1/
              ontology/Concepts.owl#City | Serviço: 1
        4: Hotel Cost | http://127.0.0.1/services/HotelCost.owl#HotelCost
              Entrada: http://127.0.0.1/ontology/Concepts.owl#TimeInterval | Saída: http://127.0.0.1/
              ontology/Concepts.owl#TimeInterval | Serviço: 1
              Entrada: http://127.0.0.1/ontology/Concepts.owl#Place | Saída: http://127.0.0.1/
              ontology/Concepts.owl#Place | Serviço: 3

Tempo total de execução: 19.5 segundos.
```

O resultado 4 encontrado representa uma saída onde os serviços compostos se ramificam e essas ramificações terminam por habilitar o último serviço da composição (*Split*). Isso significa que o algoritmo genético foi capaz de gerar um encadeamento válido de serviços. A Figura 4.7 traz uma representação gráfica da resposta encontrada.

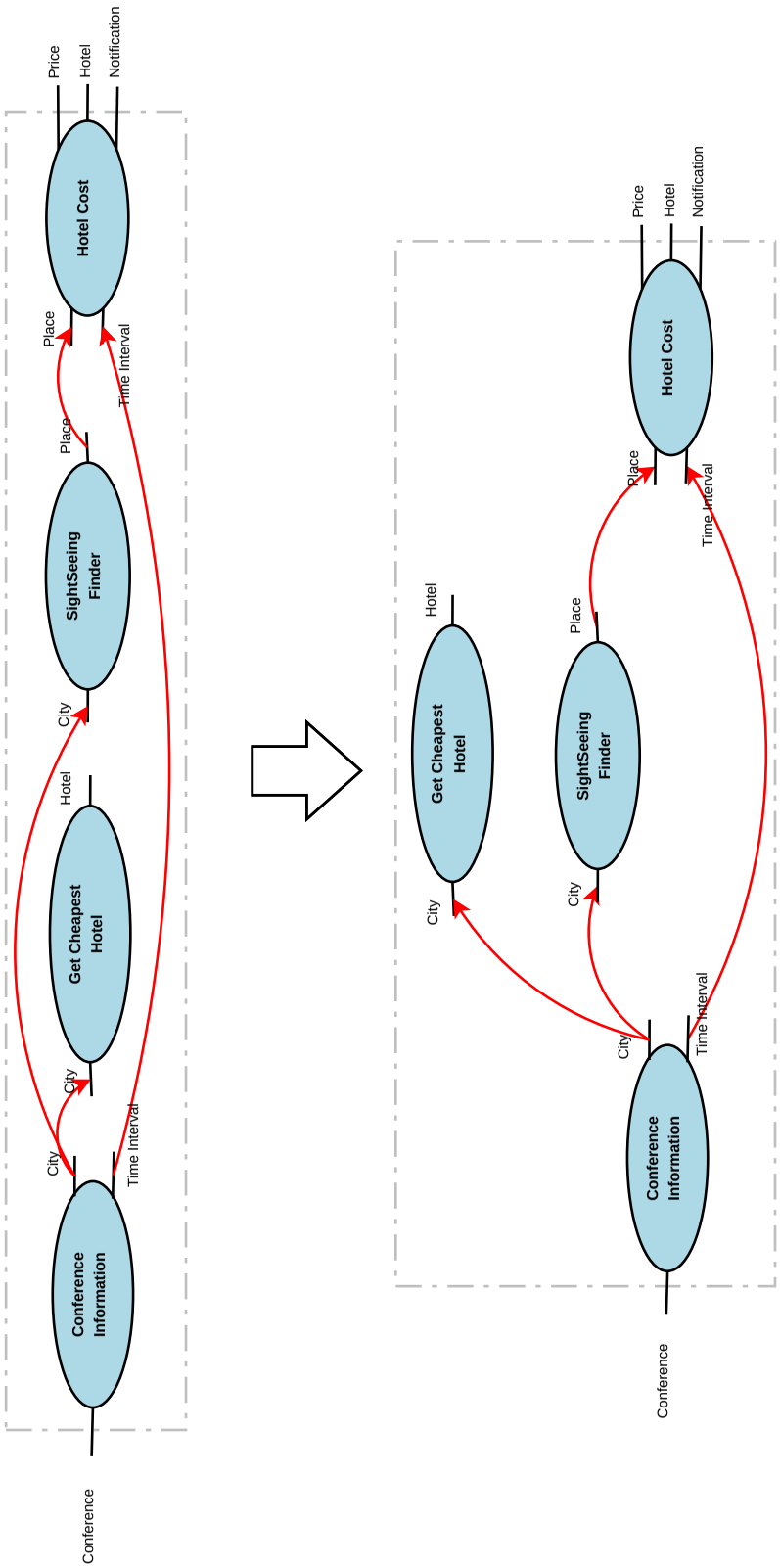

Figura 4.7 Representação gráfica de uma composição encontrada com *Split* de serviços.

CAPÍTULO 5

Conclusões

E agora estou perdido!
Devo parar?
- Não, se paras, estás perdido.
— GOETHE (Poemas)

Even Jeovah, after Moses had got the commandments committed to stone probably thought: "I always forget the things I really intend to say".
— CHRISTOPHER MORLEY

Neste capítulo, são apresentadas as conclusões em relação aos objetivos gerais e específicos apresentados no Capítulo 1, às necessidades inerentes à utilização de Serviços Web, apresentadas na introdução do Capítulo 2 e à contribuição deste trabalho.

5.1 Quanto aos Objetivos

Seguindo as orientações dos objetivos, este trabalho apresentou uma proposta de algoritmo genético para descoberta e composição de Serviços Web Semânticos sob a ótica do Algoritmo Genético Baseado em Tipos Abstratos de Dados, tal como definido no Capítulo 3. Ainda, utilizou-se Java para implementar a ferramenta que realiza a modelagem.

Quanto ao comportamento da ferramenta, este foi considerado satisfatório visto que os testes dispostos na Seção 4.4 mostraram que a mesma é capaz de executar buscas por serviços que atendam unitariamente a uma requisição, e também de compor serviços para atender a uma requisição quando não existe serviço unitário que o faça. Ainda, a ferramenta mostrou-se capaz de criar composições de serviços que envolvem o conceito de *split* de serviços, prova esta evidenciada também na Seção 4.4.

Por se tratar de um Algoritmo Genético, que tem por base a busca por uma solução em um em espaço de estados não necessariamente contínuo, não se garante que a solução encontrada será sempre ótima, tal como garantido pela busca não-informada definida na Seção 2.3. Porém o esforço computacional para grandes bases de serviços se torna menor, uma vez que a busca não-informada, geradora de soluções ótimas, é grande consumidora de memória.

As necessidades ao se utilizar serviços web apresentadas na introdução do Capítulo 2 foram atendidas por intermédio da utilização de tecnologias já consagradas e recomendadas tais como o OWL, OWL-S e a OWL-S API. Desafios tais como a monitoração da execução são vencidos com a criação de objetos específicos de tais arcabouços e a invocação automática são inerentes à descoberta de um serviço unitário, ou composição, que atenda a uma dada requisição.

5.2 Quanto à Contribuição

Como contribuições deste trabalho, destacam-se a modelagem de um algoritmo genético utilizando a orientação a objetos como abstração de fundo e a composição de serviços através de *split*.

Frente aos algoritmos evolucionários tradicionais, onde os indivíduos são tratados como sequências binárias ou cadeias de caracteres, destaca-se a facilidade da modelagem usando o GAADT para problemas com alto grau de abstração tais como o da descoberta e composição de SWS. Seria demasiadamente complexo e detalhista o trabalho de representar todas as relações entre conceitos semânticos utilizados para descrever serviços, o seu grau de aproximação com uma dada pesquisa e seu papel numa composição candidata a cada operação do algoritmo (mutação, troca, cruzamento etc.). O autor considera a abstração provida pelo GAADT peça chave para a resolução deste problema na abordagem dos algoritmos evolucionários, bem como para a rápida implementação de melhorias de código (com relação à otimização de sua execução).

Quanto à possibilidade de compor serviços utilizando *split*, este não é um fato tão abordado na literatura. O levantamento bibliográfico não menciona fortemente tais capacidades em estratégias baseadas em sistemas multi-agentes, buscas informadas e não-informadas. Dado o algoritmo deste trabalho, tal problema pôde ser modelado e implementado de forma mais simplificada do que o seria numa busca em largura ou numa estratégia de encadeamento (para frente ou para trás), visto que as possibilidades infinitas de se organizar os serviços de uma base numa estrutura de grafo levaria tais estratégias ao consumo excessivo de memória e processamento.

5.3 Trabalhos Futuros

Quanto aos trabalhos futuros, podemos destacar:

- Utilização e análise de desempenho entre as diversas estratégias citadas no Capítulo Capítulo 2, a fim de obter dados mais precisos quanto aos tempos de resposta (eficiência das estratégias) e a qualidade das soluções encontradas;
- Testes comparativos com implementações de outras abordagens (não informada, informada e SMA) para verificar desempenho e qualidade das soluções encontradas;

- Refatoramento do código criado com fins de incremento de performance, uma vez que o objetivo da concepção do mesmo foi o de prova de conceito;
- Adequação da base de descrições de serviços à versão 1.2 da linguagem OWL-S para inclusão de parâmetros de pré-condições e efeitos;
- Utilização de parâmetros de pré-condições e efeitos na descoberta e composição para melhores filtragem e acurácia do algoritmo em suas respostas, evitando que sejam considerados dois serviços de propósitos diferentes mas que tenham os mesmos tipos de dados de entrada e saída.

APÊNDICE A

Algoritmo Genético Baseado em Tipos Abstratos de Dados - GAADT

A.1 Introdução

Este capítulo apresenta o GAADT em termos do problema do caixeiro viajante e demonstra seu formalismo. Maiores detalhes poderão ser encontrados em Vieira(38).

A.2 O Problema do Caixeiro Viajante

O problema do caixeiro viajante pode ser resumido da seguinte forma: Dados um conjunto N de cidades e uma matriz de distâncias formada pelas ligações de cada par de cidades deste conjunto, construa a menor rota para o caixeiro percorrer as cidades de N, que parta de uma cidade n ($n \in N$), que passe por todas as outras cidades uma única vez e que termine na cidade n de origem.

Para um conjunto N de cardinalidade φ, a rota a ser percorrida pelo caixeiro seria composta por $\varphi + 1$ pontos de passagem. O primeiro e o último ponto de passagem seriam preenchidos com o mesmo nome de cidade, neste caso n. Isto, por ser uma ligação comum a qualquer roteiro, não influencia no cálculo do número de rotas possíveis. O segundo ponto de passagem da rota pode ser preenchido com qualquer uma das cidades ainda não visitadas, correspondendo a $\varphi - 1$ alternativas. O terceiro ponto de passagem da rota pode ser preenchido com qualquer uma das cidades ainda não visitadas, correspondendo a $\varphi - 2$ alternativas. Prosseguindo com este raciocínio, tem-se que, no momento do preenchimento do φ-ésimo ponto de passagem da rota, só existirá uma única cidade ainda não visitada para ocupá-lo. Pode-se concluir, então, que o número de rotas analisadas para um conjunto N de cardinalidade φ é $(\varphi - 1) \times (\varphi - 2) \times \ldots \times 1 = (\varphi - 1)!$.

A.3 Tipos Básicos

Definição A.1. (Base) - Uma base B é o conjunto de todas as unidades genéticas elementares que podem ser usadas na formação do material genético dos cromossomos de uma população.

No caso do problema do caixeiro viajante, envolvendo quatro cidades denominadas de *CidadeA*, *CidadeB*, *CidadeC* e *CidadeD*), o conjunto base pode ser instanciado como $B = \{CidadeA, CidadeB, CidadeC, CidadeD\}$.

O *AFG* deveria estabelecer que as sequências válidas são aquelas com tamanho igual a 2, tal que a base da primeira posição desta sequência deve ser diferente da base que ocupa a segunda posição. A semântica associada a cada sequência $\langle b_1, b_2 \rangle$ assim descrita é que existe um caminho da cidade b_1 para a cidade b_2. $AFG = \{(\forall g \in AFG \bullet \#g = 2), (\forall g = \langle b_1, b_2 \rangle \in AFG \bullet (b_1 \neq b_2)\}$

Definição A.2. (Gene) - Um gene *g* é uma sequência formada pelos elementos da base que pertence ao conjunto *AFG*.

Os genes são agrupados em conjuntos para formar os cromossomos da população e as características apresentadas pelos cromossomos de uma população servem também para classificá-los em grupos taxonômicos (espécies e famílias) em função do grau de similaridade das características compartilhadas pelos mesmos. Este fato conduziu alguns pesquisadores a concluir erroneamente que todos os cromossomos com alto grau de similaridade genética (e morfológica) pertenceriam ao mesmo grupo taxonômico. O contra exemplo mais forte a esta conclusão é o alto grau de similaridade existente entre o homem e o chimpanzé, os quais não pertencem ao mesmo grupo taxonômico.

Após muitas tentativas, mal sucedidas, de diferenciar estas espécies pela característica *inteligência* atribuída ao homem, King e Wilson concluíram que a principal diferença entre o homem e o chimpanzé não está em suas características genéticas e morfológicas, mas sim no seu desenvolvimento embrionário. Segundo King e Wilson, existe um *relógio biológico* que ativa o início e fim da formação de uma característica em cada uma destas espécies em tempos embrionários diferentes. Tal *relógio biológico* deve também garantir que o cruzamento de dois cromossomos de uma mesma espécie resulte em um cromossomo da mesma espécie.

Neste trabalho, o *relógio biológico* do cromossomo será representado pelo conjunto de *Axiomas de Formação de Cromossomos* (*AFC*), o qual deverá ser definido para cada situação de acordo com a semântica adotada para o cromossomo.

Por exemplo, um cromossomo para o problema do caixeiro viajante é qualquer conjunto de genes, tal que para todo elemento *b* do tipo base o número de ocorrências de *b* como cidade de partida e o número de ocorrências de *b* como cidade de chegada são ambos iguais a 1.

$$AFC = \forall c \in AFC \forall b \in B \bullet (\#g \in c \mid \exists a \in B \bullet g = \langle b, a \rangle = \#g \in c \mid \exists a \in B \bullet g = \langle a, b \rangle = 1)$$

Definição A.3. (Cromossomo) - Um cromossomo *c* é um conjunto de genes que obedece às condições estabelecidas pelo *AFC*.

Definição A.4. (População) - Uma população P_j é um conjunto de cromossomos construídos conforme des-crito na Definição A.3

O tipo população é o conjunto formado por todos os conjuntos formados por objetos do tipo cromossomo (ou seja, $P = \mathbb{P}(\mathbb{P}(C))$), que são possíveis resultados para o problema em

foco segundo a interpretação adotada para os tipos *C*, *G* e *B*. Deve-se ressaltar que a geração da população vazia pelo algoritmo indica que a interpretação adotada para o problema está errada.

A.4 Operadores Genéticos

O GAADT trabalha com dois tipos de operadores genéticos: o de reprodução e o de mutação. O operador genético de reprodução caracteriza-se por combinar os genes de dois cromossomos (cromossomos-pai) para formar outros cromossomos (cromossomos-filho), enquanto que o operador genético de mutação caracteriza-se por alterar a identidade de um cromossomo para formar um outro cromossomo (cromossomo-mutante).

O gene dos cromossomos-pai para uma dada característica que fará parte dos cromossomos-filho é aquele que melhor satisfaz as restrições do problema sobre a característica expressa por este gene, o qual será denominado de gene-dominante. Este gene não apresenta o mesmo nível epistemológico do gene dominante proposto por Mendel em seu trabalho com ervilhas, já que existe uma grande diferença entre dizer que um gene g_1 satisfaz melhor as restrições do problema do que um gene g_2 e dizer que o fator hereditário de um gene g_1 é superior ao fator hereditário de um gene g_2.

O grau de adaptação de um gene é dado por uma função *grau* e será considerada a existência de um gene g_λ que será usado para representar um gene que não expressa qualquer característica, de forma que a sua presença ou ausência não altera a identidade do cromossomo, o qual satisfaz as restrições impostas pelo conjunto de axiomas de formação de genes. Seu grau de adaptação é menor que o grau de adaptação de qualquer outro elemento do tipo gene. Tal gene será denominado de gene-inócuo.

Definição A.5. (Grau) - O grau de adaptação de um gene é uma função *grau* do seguinte tipo: $grau : G \rightarrow K$ tal que, a cada gene g, $g \in G$, é associado um único número k, $k \in K$ (K é um corpo ordenado[1]), chamado de $grau(g)$ e que reflete, segundo a interpretação adotada para o problema, uma estratificação comparativa entre a adaptação dos genes.

O gene inócuo será uma constante do sistema, cujo valor deverá ser definido no momento da instanciação do algoritmo a um dado problema. Para o problema do caixeiro viajante, será preciso redefinir o tipo base e o *AFG*, para que o gene inócuo possa ser definido. O tipo base é acrescido do elemento que irá formar a sequência do gene inócuo, $B = \{CidadeA, CidadeB, CidadeC, CidadeD, CidadeInocua\}$.

O conjunto de axiomas de formação de gene irá conter mais um predicado, para informar a composição do gene inócuo, que neste caso é a sequência $\langle CidadeInocua, CidadeInocua \rangle$.

Para o problema do caixeiro viajante, a função *grau* deverá fornecer uma medida sobre a distância entra as cidades consideradas, de modo que a soma destas medidas informem a

[1]Corpo ordenado - é uma estrutura algébrica, com duas operações, sem divisores próprios de zero e munido de uma ordem. Ex: $\langle \mathbb{R}, \leq, +, \times, 0, 1 \rangle$.

menor distância a ser percorrida pelo caixeiro. A distância entre as cidades *CidadeA*, *CidadeB*, *CidadeC* e *CidadeD* está contida na Tabela A.1. Assim a função $grau(g = \langle b_1, b_2 \rangle) = \frac{1}{d(b_1,b_2)}$ para todo gene diferente do gene inócuo, e $grau(g_\lambda)$ é zero.

	CidadeA	*CidadeB*	*CidadeC*	*CidadeD*
CidadeA	0	3	7	5
CidadeB	3	0	2	1
CidadeC	7	2	0	4
CidadeD	5	1	4	0

Tabela A.1 Distância entre as cidades

$$grau(g) = \begin{cases} \frac{0}{1} & \text{se } g = g_\lambda. \\ \frac{1}{3} & \text{se } g = \{CidadeA, CidadeB\}. \\ \frac{1}{7} & \text{se } g = \{CidadeA, CidadeC\}. \\ \frac{1}{5} & \text{se } g = \{CidadeA, CidadeD\}. \\ \frac{1}{2} & \text{se } g = \{CidadeB, CidadeC\}. \\ \frac{1}{1} & \text{se } g = \{CidadeB, CidadeD\}. \\ \frac{1}{4} & \text{se } g = \{CidadeC, CidadeD\}. \end{cases}$$

Lema A.1. $\forall g : G \mid g \neq g_\lambda \bullet (grau(g), grau(g_\lambda)) \in maior\mathbb{Q}$

Por uma questão de nomenclatura, será também exibida a relação de equivalência cromossômica, denotada por $\equiv_C$, definida da seguinte maneira: $\equiv_C$: $C \leftrightarrow C$, tal que $c_1 \equiv_C c_2$, se e somente se, $c_1 - \{g_\lambda\} = c_2 - \{g_\lambda\}$.

O gene dominante é identificado pela função *domi* que recebe um par de genes, um de cada um dos cromossomos-pai, e retorna o gene de maior grau de adaptação se os genes fornecidos expressarem uma mesma característica. Caso os genes fornecidos não expressem uma mesma característica, então a função *domi* retornará g_λ.

Dados dois genes, diz-se que eles expressam uma mesma característica se existe um atributo relevante para o problema em foco que seja satisfeito pelos genes fornecidos. No caso do problema do caixeiro viajante, o conjunto *atributoRelevante* exige que as cidades que compõem um dos genes analisados sejam as mesmas que compõem o outro gene, como ilustra a definição axiomática abaixo:

$\forall g_1, g_2 \in G(\text{ran}\, g_1 = \text{ran}\, g_2 \Leftrightarrow (\exists G_1 \in atributoRelevante(\{g_1, g_2\} \subseteq G_1)))$

A relação *mesma* é especificada pela seguinte definição axiomática:

$\forall g_1, g_2 \in G((g_1, g_2) \in mesma \Leftrightarrow (\exists G_1 \in atributoRelevante(\{g_1, g_2\} \subseteq G_1)))$

Lema A.2. $\forall g \in G \bullet (g, g) \in mesma.$

Lema A.3. $\forall g_1, g_2 \in G \mid (g_1, g_2) \in mesma \bullet (g_2, g_1) \in mesma.$

Lema A.4. $\forall g_1, g_2, g_3 \in G \mid (g_1, g_2) \in mesma \wedge (g_2, g_3) \in mesma \bullet (g_1, g_3) \in mesma$.

Definição A.6. (Dominante) - O gene dominante é uma função *domi* do seguinte tipo:
$domi : G \times G \rightarrow G$

$$domi(g_1, g_2) = \left\{ \begin{array}{ll} g_\lambda & \text{se } (g_1, g_2) \notin mesma, \\ g_1 & \text{se } (g_1, g_2) \in mesma \wedge grau(g_1) \geq grau(g_2), \\ g_2 & \text{se } (g_1, g_2) \in mesma \wedge grau(g_1) < grau(g_2). \end{array} \right\}$$

A especificação da função *domi* é feita através de uma definição axiomática, na qual cada rótulo representa o predicado que descreve cada uma das alternativas do sistema que define esta função.

$$domi(g_1, g_2) = \left\{ \begin{array}{ll} g_1 & \text{se } ((g_1, g_2) \in mesma) \wedge (grau(g_1) = grau(g_2)), \\ g_1 & \text{se } ((g_1, g_2) \in mesma) \wedge (grau(g_1) \langle grau(g_2)), \\ g_2 & \text{se } ((g_1, g_2) \in mesma) \wedge (grau(g_1) \rangle grau(g_2)), \\ g_\lambda & \text{se } ((g_1, g_2) \notin mesma). \end{array} \right.$$

Lema A.5. $\forall g \in G((g, g) \in mesma \Leftrightarrow domi(g, g) = g)$

Lema A.6. $\forall g_1, g_2, g_3 \in G \mid (g_1, g_2) \in mesma \wedge (g_2, g_3) \in mesma$
$(domi(g_1, domi(g_2, g_3)) = domi(domi(g_1, g_2), g_3))$

A produção de novos cromossomos durante o processo evolutivo de uma população serve para direcionar a busca por cromossomos mais adaptados através da transmissão das características de maior grau de adaptação presentes nos cromossomos da população atual. A adaptação de um cromossomo é dada pela função *adapt*.

Definição A.7. (Adaptação) - A adaptação de um cromossomo é uma função *adapt* do seguinte tipo:
$adapt : C \rightarrow K$
$adapt(c) = \sum_{g \in c} \Theta_{c,g} \times grau(g)$
onde $\Theta_{c,g}$ é o peso com o qual o gene g contribui para a adaptação do cromossomo c.

A assinatura da função Θ é o mapeamento do par cromossomo e gene em um valor racional positivo, a ser dado pelo problema ao qual o GAADT será aplicado. O conjunto dos números racionais positivos é especificado pela seguinte definição abreviada $\mathbb{Q}$.

Por exemplo, para o problema do caixeiro viajante, o peso de qualquer gene no cromossomo será igual à $\frac{0}{1}$, se este gene não pertencer ao cromossomo, caso contrário ele será $\frac{1}{1}$.

$$\Theta(c, g) = \left\{ \begin{array}{ll} \frac{0}{1} & \text{se } (g \notin c), \\ \frac{1}{1} & \text{se } (g \in c). \end{array} \right\}$$

A operação de cruzamento recebe dois cromossomos-pai, aptos a cruzarem, e retorna uma população cujos cromossomos são formados somente pelos genes dominantes dos cromossomos fornecidos. Logo, para se definir esta função precisa-se antes definir uma função para

selecionar os cromossomos aptos a cruzarem (seleção) e uma função para retornar o conjunto de genes dominantes para todas as características existentes nos cromossomos-pai (fecundação).

A função de seleção recebe uma população P_1 e retorna a subpopulação de P_1 formada pelos cromossomos que satisfazem um requisito do problema r, descrito por uma fórmula em lógica de primeira ordem, o qual indica quando um dado cromossomo é considerado apto a cruzar.

Definição A.8. (Seleção) - A seleção dos cromossomos que satisfazem um predicado r é uma função *sel* do seguinte tipo:
$sel : \mathbb{P}(P) \times \mathbb{P}(P) \rightarrow \mathbb{P}(P)$
$sel(P_1, r) = P_1 \cap r$.

A função fecundação recebe dois cromossomos e retorna o conjunto de genes dominantes entre todos os genes dos cromossomos fornecidos.

Definição A.9. (Fecundação) - A fecundação é uma função *fec* do seguinte tipo:
$fec : C \times C \rightarrow \mathbb{P}(G)$
$fec(c_1, c_2) = \{g \mid \forall g_1 \in c_1 \, \forall g_2 \in c_2(g = domi(g_1, g_2))\}$

Lema A.7. $\forall c \in C((fec(c,c), c) \in \equiv_C)$

Os cromossomos-pai aptos a cruzarem são representados pelo conjunto *MACHO* e *FEMEA*, formados da seguinte forma: $MACHO = sel(P_1, M)$ e $FEMEA = sel(P_1, F)$, onde P_1 é uma subpopulação da população atual formada por cromossomos adaptados ao ambiente e, M e F são dois predicados sobre o tipo população pertencentes ao conjunto de requisitos do ambiente *Rq*, escritos em uma linguagem de primeira ordem.

Note que, dependendo da especificação dos requisitos do ambiente M e F, a reprodução gerada pode ser sexuada, assexuada ou mista. A reprodução sexuada ocorre quando $M \cap F = \emptyset$, a assexuada quando $M = F$, e a mista quando $M \cap F \neq \emptyset$ e $M \neq F$.

Por exemplo, para o problema do caixeiro viajante, o requisito M pode exigir que as cidades *CidadeA* e *CidadeB* façam parte da rota representada pelos seus cromossomos, enquanto F exige que as cidades *CidadeC* e *CidadeD* façam parte da rota representada pelos seus cromossomos. Assim, o cromossomo resultante deste cruzamento deve representar uma rota que passa pelas cidades *CidadeA*, *CidadeB*, *CidadeC*, e *CidadeD*.

$M = \{c \in C \mid \exists b_1, b_2 \in B(\forall g_1 = (b_1, b_2) \in c(b_1 \in \{CidadeA, CidadeB\}) \vee$
$(b_2 \in \{CidadeA, CidadeB\}))\}$

$F = \{c \in C \mid \exists b_1, b_2 \in B(\forall g_1 = (b_1, b_2) \in c(b_1 \in \{CidadeC, CidadeD\}) \vee$
$(b_2 \in \{CidadeC, CidadeD\}))\}$

Definição A.10. (Cruzamento) - O cruzamento é uma função *cruz* do seguinte tipo:
$cruz : MACHO \times FEMEA \rightarrow P$
$cruz(c_1, c_2) = \{c \mid c \subseteq fec(c_1, c_2)\}$

O operador genético de mutação, definido para o GAADT, é composto pelas funções de inserção, supressão e troca. A operação de inserção *ins* adiciona um conjunto de genes ao cromossomo de origem.

Definição A.11. (Inserção) - A inserção é uma função *ins* do seguinte tipo:
$ins : C \times \mathbb{P}(G) \to C$
$$ins(c, G_1) = \begin{cases} c \cup G_1 & \text{se } c \cup G_1 \in AFC, \\ c & \text{caso contrário.} \end{cases}$$

A operação de supressão *del* remove um conjunto de genes do cromossomo de origem.

Definição A.12. (Supressão) - A supressão é uma função *del* do seguinte tipo:
$del : C \times \mathbb{P}(G) \to C$
$$del(c, G_1) = \begin{cases} c - G_1 & \text{se } c - G_1 \in AFC, \\ c & \text{caso contrário.} \end{cases}$$

A operação de troca *troc* remove um conjunto de genes do cromossomo de origem e lhe adiciona outro conjunto de genes.

Definição A.13. (Troca) - A troca é uma função *troc* do seguinte tipo:
$troc : C \times \mathbb{P}(G) \times \mathbb{P}(G) \to C$
$$troc(c, G_1, G_2) = \begin{cases} (c \cup G_1) - G_2 & \text{se } c \cup G_1 \in AFC \wedge (c \cup G_1) - G_2 \in AFC, \\ c \cup G_1 & \text{se } c \cup G_1 \in AFC \wedge (c \cup G_1) - G_2 \notin AFC, \\ c - G_2 & \text{se } c \cup G_1 \notin AFC \wedge c - G_2 \in AFC, \\ c & \text{se } c \cup G_1 \notin AFC \wedge c - G_2 \notin AFC. \end{cases}$$

Corolário A.1. $\forall c \in C;\ G_1, G_2 \in \mathbb{P}(G)(troc(c, G_1, G_2) = del(ins(c, G_1), G_2))$

Este corolário estabelece que as ações da função de inserção e supressão podem ser vistas como casos particulares da ação da função de troca.

Definição A.14. (Mutação) - A mutação é um predicado $mut \subseteq \mathbb{P}(P)$, tal que:
$mut(c_1) = \{c_2 \mid \exists G_1, G_2 : \mathbb{P}(G)\,((\#G_1 \leq \#c_1 \operatorname{div} 2) \wedge (\#G_2 \leq \#c_1 \operatorname{div} 2) \wedge (troc(c_1, G_1, G_2) = c_2)$
$\wedge (adapt(c_2), adapt(c_1)) \in maior\mathbb{Q}\}$

A restrição $(adapt(c_2), adapt(c_1)) \in maior\mathbb{Q}$ garante que todo cromossomo-mutante é mais adaptado do que o cromossomo que lhe deu origem. E a limitação do número de genes que podem ser alterados do cromossomo no cromossomo-mutante em cinquenta por cento do tamanho do cromossomo fornecido deve-se ao fato de que se as mutações ocorridas em um cromossomo de uma dada espécie forem muito grandes, então este cromossomo seria repelido pelos cromossomos da sua espécie, por não ser considerado mais um igual a estes.

A.5 Ambiente

Um algoritmo genético opera sobre populações de cromossomos que evoluem de acordo com as características de um ambiente *A*. Um ambiente *A* é uma 8-tupla $\langle P, \mathbb{P}(P), Rq, AFG, AGC, Tx, \Sigma, P_0 \rangle$, onde:

- P é a população,
- $\mathbb{P}(P)$ é o conjunto potência de P,
- Rq é o conjunto dos requisitos (características expressas através de fórmulas numa linguagem de primeira ordem) do problema que influenciam a genealogia da população P,
- AFG é o conjunto de axiomas de formação dos genes dos cromossomos da população P,
- AFC é o conjunto de axiomas de formação dos cromossomos da população P e
- Tx é o conjunto de pares de cromossomos (x, y), onde x é um cromossomo construído a partir do cromossomo y, pela ação da operação de cruzamento ou mutação, registrando desta forma a genealogia dos cromossomos pertencentes às populações geradas pelo GA-ADT durante a sua execução,
- Σ é o conjunto de operadores genealógicos que atuam sobre a população P,
- P_0 é uma sub-população pertencente a $\mathbb{P}(P)$, chamada de população inicial, com no mínimo um cromossomo.

O processo de evolução darwinista, segundo o qual todas as espécies desenvolveram-se a partir de outras espécies, pela transmissão hereditária de pequenas variações, em sucessivas gerações, resultando na sobrevivência das espécies que melhor adaptaram-se ao ambiente, é induzido pelas alterações ambientais produzidas pela natureza. Este papel desempenhado pela natureza, na visão de evolução darwinista, aqui será representado pelo GAADT, que é quem submete os cromossomos de uma população à ação dos requisitos do problema Rq, resultando assim na geração de novos cromossomos a partir daqueles já existentes.

A.6 Algoritmo

O GAADT é uma função *GAADT* que recebe a população P_0 e, depois de submetê-la à simulação de um processo evolutivo, devolve uma população P_t. Os cromossomos da população P_t são os cromossomos das populações $P_0, P_1, \ldots, P_{t-1}$ que ainda satisfazem os requisitos do problema Rq, ou então são novos cromossomos resultantes da ação genealógica das operações

de cruzamento e mutação sobre os cromossomos da população P_{t-1} que apresentam adaptação maior do que a adaptação dos cromossomos que lhes deram origem. Diz-se então que a população P_t evoluiu da população P_0.

Os cromossomos das populações P_0,P_1,...,P_{t-1} que não mais satisfaçam os requisitos do problema *Rq* não participarão da construção da população P_t, podendo ser assim entendidos como fazendo parte da população de cromossomos "mortos", que não figurarão entre os cromossomos da população P_t e das populações seguintes manipuladas pela função *GAADT*. Não obstante, tais cromossomos serão recuperados pela análise da taxonomia *Tx* dos cromossomos da população atual para evitar que eles apareçam novamente nas próximas iterações da função *GAADT*. Esta restrição atende ao entendimento do processo de evolução darwinista, que não contempla a possibilidade de uma espécie extinta voltar a aparecer num outro momento futuro.

Antes de ser apresentada uma definição para a função *GAADT*, deve-se observar a necessidade de se estabelecer um critério de preservação sobre a população atual P_t, para orientar o corte dos cromossomos que não devem figurar nas populações P_{t+1},P_{t+2},.... Na definição da função *GAADT*, este ponto de corte será representado por um predicado unário p_{corte} pertencente ao conjunto de requisitos do problema *Rq*, que atua sobre os cromossomos de P_t.

Por exemplo, para o problema do caixeiro viajante, o ponto de corte poderia selecionar somente os cromossomos com grau de adaptação maior ou igual ao valor da adaptação média da população. A adaptação média da população é obtida dividindo-se a soma da adaptação de todos os cromossomos da população pelo número de cromossomos desta população.

Os critérios de parada adotados pela função *GAADT* são: a) o número máximo de iterações desejadas, b) valor da adaptação dos cromossomos considerado satisfatório para o resultado do problema em análise. Estes critérios também fazem parte do conjunto de requisitos do problema *Rq*.

Definição A.15. (GAADT) - O GAADT é uma função *GAADT* do seguinte tipo:
$GAADT : A \rightarrow A$

$$GAADT(P_t) = \begin{cases} P_{otm} & \text{se } P_{otm} = \{c \mid \forall c : P_t\,(adapt(c) \geq k)\} \neq \emptyset, \\ P_{t+1} & \text{se } t+2 = T, \\ GAADT(P_{t+1}) & \text{caso contrário.} \end{cases}$$

onde $P_{t+1} = cruz(a,b) \cup mut(c) \cup p_{corte}(P_t)$ com e $a,b,c \in P_t$, P_0 é a população inicial considerada, $k \in K$ é um valor imposto pelo ambiente *A*, como critério de aceitação de cromossomos em P_t que satisfazem o problema e $T \in \mathbb{N}$ é um número dado como critério de satisfação do número de iterações.

No processo acima, a primeira e a segunda opções de saída são condições de parada. A segunda ocorrerá garantidamente, se eventualmente a primeira não ocorrer. Também deve ser observado que para qualquer entrada P_t, o processo *GAADT* dá uma saída bem determinada. Isto significa que o *GAADT* é um algoritmo e desta forma um procedimento correto.

APÊNDICE B

Recuperação de Informação Através do Modelo de Espaço Vetorial

B.1 Informação Representada no Espaço Vetorial

No modelo de Recuperação de Informação - RI - de espaço vetorial, um vetor é usado para representar cada item ou *documento* de uma coleção. Cada componente do vetor reflete um conceito, palavra-chave ou *termo* particular associado ao documento dado. O valor dado a esse componente reflete a importância ao representar a semântica do documento. Tipicamente, o valor é uma função da frequência com que o termo ocorre no documento ou na coleção de documentos como um todo, tal como afirmam Baeza-Yates e Ribeiro-Neto(43). Suponha um documento descrito, para fins de indexação, por três termos: T_1, T_2 e T_3. Ele pode ser representado por um vetor nas três dimensões correspondentes. A Figura B.1 mostra o vetor quando os termos tem pesos 0.5, 2.5 e 5, respectivamente. Neste caso, o termo T_3 é o termo mais significante no documento, com T_2 com importância secundária e T_1 com importância ainda menor.

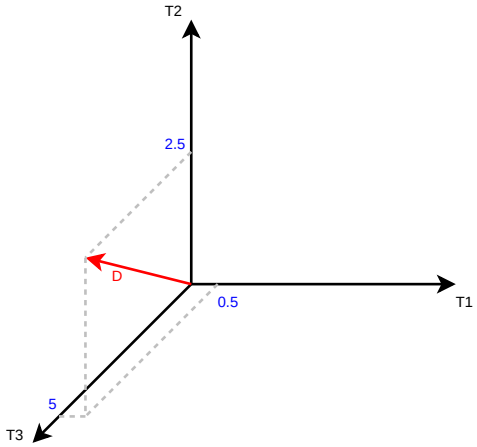

Figura B.1 Representação de um documento no Modelo Vetorial

Um banco de dados contendo um total de d documentos descritos por t termos é representado como uma *matriz A de termos*$\times$*documentos* $t \times d$. Os vetores d representando os d documentos das colunas da matriz. Assim, o elemento da matriz a_{ij} é a frequência com a qual o termo i ocorre no documento j, como em Manning, Raghavan e Schtze(44). Na linguagem

do modelo vetorial, as colunas de A são os *vetores de documentos*, e as linhas de A são os *vetores de termos*. O conteúdo semântico da base de dados é contido integralmente no espaço das colunas de A, significando que os vetores de documentos abrangem esse conteúdo. Nem todo vetor representado no espaço das colunas de A tem uma interpretação específica em termos de da coleção de documentos propriamente dita, ou seja, uma combinação linear de vetores correspondendo a dois títulos de documentos pode não se traduzir diretamente em um título de documento que tenha um significado válido. O que é importante de uma perspectiva de RI, no entanto, é a possibilidade de se explorar relações geométricas entre vetores de documentos para modelar similaridades e diferenças de utilização de termos.

Há uma variedade de esquemas disponíveis para ponderar os elementos da matriz. Aos elementos a_{ij} da matriz termo-por-documento A são frequentemente atribuídos dois fatores $a_{ij} = l_{ij}g_i$. Neste caso, o fator g_i, também conhecido como *Inverse Document Frequency (IdF)* Manning, Raghavan e Schtze(44), é um *peso global* que reflete o valor geral do termo i como um termo indexador para a coleção inteira. Como um exemplo, considere um termo muito comum como **computador** em uma coleção de artigos de computadores pessoais. Não é importante incluir este termo na descrição de um documento justamente porque é sabido que todos os documentos são sobre computadores (independentemente deles usarem ou não o termo **computador** em suas descrições), então um valor pequeno para o peso g_i é apropriado.

O fator l_{ij}, conhecido como *Term Frequency (TF)* Manning, Raghavan e Schtze(44), é um *peso local* que reflete a importância do termo i dentro do próprio documento j. Há uma gama de valores para pesos locais, que variam desde simples valores binários (0 ou 1) a funções envolvendo logaritmos de frequências de termos. Estas últimas tem um efeito de suavização na qual termos de alta frequência que tem valor discriminatório limitado ganham pesos baixos. Esquemas de pesos globais variam desde simples normalizações a avançadas abordagens baseadas em estatística. Maiores detalhes sobre estratégias de pesos locais e globais podem ser vistas em Baeza-Yates e Ribeiro-Neto(43) e Manning, Raghavan e Schtze(44).

Para coleções de texto abrangendo muitos contextos, tais como uma enciclopédia, o número de termos é frequentemente muito maior que o número de documentos: $t \gg d$. No caso da Web, a situação é revertida. Uma matriz de termos-por-documentos utilizando o conteúdo do maior dicionário de inglês como termos e o conteúdo de todas as páginas em inglês como documentos deve ter um tamanho por volta de $300,000 \times 300,000,000$ Berry, Drmac e Jessup(45). Como um documento geralmente usa apenas um pequeno subconjunto do dicionário inteiro de termos gerado para um dado banco de dados, a maioria dos termos de uma matriz termos-por-documentos é zero.

Em um esquema de RI vetorial, um usuário executa uma consulta no banco de dados para encontrar documentos relevantes utilizando alguma forma de representação vetorial para estes documentos. A consulta é um conjunto de termos, possivelmente com pesos atribuídos, representados da mesma forma como o seria um documento. Novamente, é provável que muitos dos termos do banco de dados não apareçam na consulta, significando que muitos dos componentes do vetor de consulta são zero. Executar uma consulta significa encontrar os documentos mais similares à consulta na utilização e peso dos termos. No modelo vetorial, os documentos

selecionados são aqueles mais próximos geometricamente à consulta de acordo com alguma medição.

Uma medição comum de similaridade é o cosseno do ângulo entre os vetores da consulta e do documento. Se a matriz termos-por-documentos A tem colunas $a_j, j = 1, ..., d$, os cossenos de d são computados de acordo com a fórmula

$$cos\theta_j = \frac{a_j^T q}{|| a_j ||_2 || q ||_2} = \frac{\sum_{i=1}^{t} a_{ij} q_i}{\sqrt{\sum_{i=1}^{t} a_{ij}^2} \sqrt{\sum_{i=1}^{t} q_i^2}} \tag{B.1}$$

para $j = 1, ..., d$ onde a norma Euclidiana $|| x ||_2$ é definido por $|| x ||_2 = \sqrt{x^T x} = \sum_{i=1}^{T} x_i^2$ para qualquer vetor t-dimensional real x. Devido aos vetores de consulta e documento serem tipicamente esparsos, os produtos internos e normas na equação (B.1) geralmente não tem alto custo computacional. Além disso, a norma do vetor de documentos $|| a_j ||_2$ necessita ser computada apenas uma vez para qualquer matriz termo-por-documento. Perceba que multiplicando tanto a_j ou q por uma constante não muda o valor do cosseno. Então, podemos dimensionar os vetores de documentos ou consultas para qualquer valor conveniente.

B.2 Um Exemplo

A Figura B.2 demonstra como uma simples coleção de cinco títulos descrita por seis termos leva a uma matriz termos-por-documentos de tamanho 6×5, apresentada por Berry, Drmac e Jessup(45). Devido ao conteúdo de um documento ser determinado pelas frequências relativas dos termos e não pelo número total de vezes que um termo aparece, os elementos da matriz neste exemplo são dimensionados de forma que a norma Euclidiana de cada coluna é 1. Ou seja, $|| a_j ||_2 = 1$ para as colunas $a_j, j = 1, ..., 5$. Desta forma, utilizamos a *frequência de termos* como o peso local l_{ij} e não aplicamos peso global - neste caso, $g_i = 1$.

A escolha dos termos utilizados para descrever o banco de dados determina não somente o seu tamanho, mas também sua utilidade. Neste exemplo, foram utilizados apenas os termos diretamente relacionados a cozinha, significando que o interesse particular do leitor em *cozinha francesa* não resultaria na recuperação de documentos relevantes. Neste caso, adicionar os termos **francesa** e **italiana** para descrever as nacionalidades cobertas ampliaria a representação da semântica do banco de dados de forma útil. Por outro lado, incluir termos muito comuns como **de** e **para** faria muito pouco para aumentar a qualidade da matriz termos$\times$documentos.

Ao construir uma matriz termos$\times$documentos, os termos são geralmente identificados por suas palavras-tronco. No exemplo citado, a palavra *doces* conta como **doce** e a palavra *assando* conta como o termo **assar**. O uso de derivações em RI remonta aos anos 60 e a derivação reduz a necessidade de armazenamento através da redução do número mantido de palavras Berry, Drmac e Jessup(45).

Os $T = 6$ termos:

T_1: ass(ar, ado, ando)
T_2: receitas
T_3: p(ão, ães)
T_4: bolo
T_5: sobremesa(s)
T_6: torta

Os $D = 5$ documentos:

D_1: Como assar pães sem receitas
D_2: A Arte Clássica das Sobremesas de Viena
D_3: Receitas Numéricas: A Arte da Computação Científica
D_4: Pães, Sobremesas, Tortas e Bolos: Receitas de Assados em Quantidade
D_5: Sobremesas: O Livro das Melhores Receitas Francesas

A matriz 6×5 (termos$\times$documentos) antes da normalização, onde o elemento $\hat{a}_{ij}$ representa o número de vezes que o termo i aparece no título do documento j:

$$\hat{A} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

A matriz 6×5 (termos$\times$documentos) com colunas unitárias:

$$A = \begin{pmatrix} 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0.5774 & 0 & 1.0000 & 0.4082 & 0.7071 \\ 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0 & 0 & 0 & 0.4082 & 0 \\ 0 & 1.0000 & 0 & 0.4082 & 0.7071 \\ 0 & 0 & 0 & 0.4082 & 0 \end{pmatrix}$$

Figura B.2 Construção de uma matriz termos$\times$documentos A.

B.3 Execução de Consultas

Usando a pequena coleção de títulos da Figura B.2, é possível demonstrar a execução de uma consulta baseada nos ângulos de um espaço vetorial de dimensão 6. Suponha que

um usuário na busca por informação sobre cozinha inicia uma busca por livros sobre *assando pães*. A consulta correspondente, no modelo vetorial, seria escrita como o vetor $q^{(1)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T$ com entradas diferentes de zero para **assando** e **pães**. A pesquisa por documentos relevantes é conduzida através da computação dos cossenos dos ângulos θ_j entre o vetor de consulta $q^{(1)}$ e os vetores de documentos a_j pela equação (B.1). Um documento é dado como relevante somente se o cosseno do ângulo que ele faz com o vetor de consultas seja maior que um valor mínimo ou ponto de corte. Uma implementação prática poderia usar um ponto de corte como 0.9, mas no pequeno exemplo dado, foi utilizado um valor mínimo de cosseno de 0.5.

Para a consulta $q^{(1)}$, os únicos cossenos diferentes de zero foram $cos\theta_1 = 0.8165$ e $cos\theta_4 = 0.5774$. Dessa forma, todos os documentos sobre assar pães (o primeiro e o quarto) são retornados como relevantes. O segundo, terceiro e quinto documentos, que não contém esses tópicos, são corretamente ignorados.

Se o usuário tivesse simplesmente pedido livros sobre *assar*, no entanto, os resultados teriam sido notadamente diferentes. Neste caso, o vetor de consulta seria dado por $q^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$, e os cossenos dos ângulos entre a consulta e os cinco vetores de documentos seriam, respectivamente, 0.5774, 0, 0, 0.4082 e 0. Somente o primeiro documento, um livro sobre assar pães, atingiria o ponto de corte do cosseno. O quarto documento, que é de fato uma referência mais compreensiva sobre assados, não é retornado como relevante.

A comunidade de Recuperação de Informação desenvolveu uma variedade de abordagens para responder a tais falhas do modelo vetorial básico. Estas técnicas tipicamente afetam como os dados são representados na matriz termos$\times$documentos. Vários exemplos e refinamentos do modelo vetorial podem ser encontrados em Manning, Raghavan e Schtze(44).

Referências Bibliográficas

1 BERNERS-LEE; HENDLER, T.; LASSILA, J. The semantic web. *Scientific American*, maio 2001.

2 SHADBOLT, N.; BERNERS-LEE, T.; HALL, W. The semantic web revisited. *IEEE Intelligent Systems*, IEEE Computer Society, Los Alamitos, CA, USA, v. 21, n. 3, p. 96–101, 2006. ISSN 1541-1672.

3 MAHMOUD, T.; GOMEZ, J. M. Integration of semantic web services principles in soa to solve eai and erp scenarios; towards semantic service oriented architecture. In: *Proc. 3rd International Conference on Information and Communication Technologies: From Theory to Applications ICTTA 2008*. [S.l.: s.n.], 2008. p. 1–6.

4 BREHM, N.; GOMEZ, J. M.; RAUTENSTRAUCH, C. An ERP solution based on web services and peer-to-peer networks for small and medium enterprises. Inderscience Publishers, v. 1, p. 99–111, nov. 29 2005. ISSN 1479-313X. Disponível em: <http://www.inderscience.com/link.php?id=8288>.

5 MA, J.; ZHANG, Y.; HE, J. Web services discovery based on latent semantic approach. In: *ICWS*. IEEE Computer Society, 2008. p. 740–747. ISBN 978-0-7695-3310-0. Disponível em: <http://dx.doi.org/10.1109/ICWS.2008.135>.

6 CHRISTENSEN, E. et al. Web services description language (WSDL) 1.1. Http://www.w3.org/TR/wsdl. 2001.

7 MA, J.; ZHANG, Y.; HE, J. Efficiently finding web services using a clustering semantic approach. In: SHENG, Q. Z. et al. (Ed.). *CSSSIA*. ACM, 2008. (ACM International Conference Proceeding Series, v. 292), p. 5. ISBN 978-1-60558-107-1. Disponível em: <http://doi.acm.org/10.1145/1361482.1361487>.

8 HICKS, J.; GOVINDARAJU, M.; MENG, W. Search algorithms for discovery of web services. In: *Proc. IEEE International Conference on Web Services ICWS 2007*. [S.l.: s.n.], 2007. p. 1172–1173.

9 VASCONCELOS, J. B. de et al. A knowledge-engine architecture for a competence management information system. In: *Proceedings of 14th UKAIS Conference, Oxford*. [S.l.: s.n.], 2009.

10 2008 IEEE International Conference on Web Services (ICWS 2008), September 23-26, 2008, Beijing, China. [S.l.]: IEEE Computer Society, 2008. ISBN 978-0-7695-3310-0.

11 MEDITSKOS, G.; BASSILIADES, N. Object-oriented similarity measures for semantic web service matchmaking. In: *ECOWS*. IEEE Computer Society, 2007. p. 57–66. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/ECOWS.2007.17>.

12 KLUSCH, M.; FRIES, B.; SYCARA, K. P. OWLS-MX: A hybrid semantic web service matchmaker for OWL-S services. *J. Web Sem*, v. 7, n. 2, p. 121–133, 2009. Disponível em: <http://dx.doi.org/10.1016/j.websem.2008.10.001>.

13 YU, L. *Introduction to the Semantic Web and Semantic Web Services*. [S.l.]: Chapman & Hall/CRC, 2007. ISBN 1584889330.

14 LIU, C.; PENG, Y.; CHEN, J. Web services description ontology-based service discovery model. In: *Proc. IEEE/WIC/ACM International Conference on Web Intelligence WI 2006*. [S.l.: s.n.], 2006. p. 633–636.

15 CARDOSO, J. *Semantic Web Services: Theory, Tools and Applications*. [S.l.]: IGI Global, 2007. ISBN 159904045X, 9781599040455.

16 MENEZES, P. B. *Matemática Discreta para Computação e Informática*. [S.l.]: Sagra Luzzatto, 2004.

17 YAN, Y.; ZHENG, X. A planning graph based algorithm for semantic web service composition. In: *CEC/EEE*. IEEE, 2008. p. 339–342. Disponível em: <http://dx.doi.org/10.1109/CECandEEE.2008.135>.

18 SHESHAGIRI, M.; DESJARDINS, M.; FININ, T. *A Planner for Composing Services Described in DAML-S*. maio 05 2003. Disponível em: <http://citeseer.ist.psu.edu/579497.html; http://umbc.edu/ finin/papers/icaps03.pdf>.

19 WU, D. et al. Automating DAML-S web services composition using SHOP2. In: FENSEL, D.; SYCARA, K. P.; MYLOPOULOS, J. (Ed.). *International Semantic Web Conference*. [S.l.]: Springer, 2003. (Lecture Notes in Computer Science, v. 2870), p. 195–210. ISBN 3-540-20362-1.

20 HULL, R.; SU, J. Tools for composite web services: a short overview. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 34, n. 2, p. 86–95, 2005. ISSN 0163-5808.

21 WEISE, T. et al. Different approaches to semantic web service composition. In: IEEE. *Proceedings of The Third International Conference on Internet and Web Applications and Services, ICIW 2008*. Athens, Greece: IEEE Computer Society Press, 2008. Disponível em: <http://www.it-weise.de/documents/files/WBCG2008ICIW.pdf>.

22 XU, B. et al. Swsds: Quick web service discovery and composition in sewsip. In: *CEC-EEE '06: Proceedings of the The 8th IEEE International Conference on E-Commerce*

Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services. Washington, DC, USA: IEEE Computer Society, 2006. p. 71. ISBN 0-7695-2511-3.

23 PALIWAL, A. V.; ADAM, N. R.; BORNHOVD, C. Web service discovery: Adding semantics through service request expansion and latent semantic indexing. In: *Proc. IEEE International Conference on Services Computing SCC 2007*. [S.l.: s.n.], 2007. p. 106–113.

24 PIETRO, I. D. et al. Semantic web service selection at the process-level: The ebay/amazon/paypal case study. In: *Proc. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology WI-IAT '08*. [S.l.: s.n.], 2008. v. 1, p. 605–611.

25 RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach (Second Edition)*. [S.l.]: Prentice Hall, 2003.

26 ZHANG, Y. et al. Strategies for efficient syntactical and semantic web services. In: . Los Alamitos, CA, USA: IEEE Computer Society, 2006. v. 0, p. 72. ISBN 0-7695-2511-3.

27 CHAHOUD, J. J. *Planejamento para Serviços Web Semânticos*. Tese (Doutorado) — Universidade de São Paulo, 2006.

28 FERBER, J. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. [S.l.]: Addison-Wesley Professional, 1999. Paperback. ISBN 0201360489.

29 FERNáNDEZ, A.; OSSOWSKI, S. Filters for semantic service composition in service-oriented multiagent systems. In: *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2007. p. 1–3. ISBN 978-81-904262-7-5.

30 KUMAR, S.; MISHRA, R. B. Multi-agent based semantic web service composition models. *INFOCOMP Journal of Computer Science*, v. 7, n. 3, p. 42–51, 2008.

31 ALMEIDA M., B. M. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Ciência da Informação*, v. 32, n. 3, 2004. Disponível em: <http://revista.ibict.br/index.php/ciinf/article/view/17>.

32 BENER, A. B.; OZADALI, V.; ILHAN, E. S. Semantic matchmaker with precondition and effect matching using swrl. *Expert Syst. Appl.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 36, n. 5, p. 9371–9377, 2009. ISSN 0957-4174.

33 BUHLER, P. A.; GREENWOOD, D.; WEICHHART, G. A multiagent web service composition engine, revisited. *E-Commerce Technology, IEEE International Conference on, and Enterprise Computing, E-Commerce, and E-Services, IEEE International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 529–532, 2007.

34 BäCK, T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford, UK, 1996.

35 XU, J.; REIFF-MARGANIEC, S. Towards heuristic web services composition using immune algorithm. *Web Services, IEEE International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 238–245, 2008.

36 CLARO, D. B.; ALBERS, P.; HAO, J. kao. J.k.: Selecting web services for optimal composition. In: *In Proceedings of the 2nd International Workshop on Semantic and Dynamic Web Processes (SDWP 2005*. [S.l.: s.n.], 2005. p. 32–45.

37 DIBERNARDO, M.; POTTINGER, R.; WILKINSON, M. Semi-automatic web service composition for the life sciences using the biomoby semantic web framework. *Journal of Biomedical Informatics*, v. 41, n. 5, p. 837–847, 2008. Disponível em: <http://dx.doi.org/10.1016/j.jbi.2008.02.005>.

38 VIEIRA, R. V. *Um Algoritmo Genético Baseado em Tipos Abstratos de Dados e sua Especificação em Z*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2003.

39 SANTOS, F. C.; CARVALHO, C. L. de. *Aplicações de Suporte àWeb Semântica*. [S.l.], 2007.

40 MARTINO, B. D. Semantic web services discovery based on structural ontology matching. *Int. J. of Web and Grid Services*, Inderscience Publishers, v. 5, p. 46–65, mar. 17 2009. ISSN 1741-1114. Disponível em: <http://www.inderscience.com/link.php?id=23868>.

41 MCILRAITH, S. A.; MARTIN, D. L. Bringing semantics to web services. *IEEE Intelligent Systems*, v. 18, n. 1, p. 90–93, 2003. Disponível em: <http://csdl2.computer.org/dl/mags/ex/2003/01/x1090.pdf>.

42 MCGUINNESS, D. L.; WELTY, C.; SMITH, M. K. *OWL Web Ontology Language Guide*. [S.l.], fev. 2004. Http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

43 BAEZA-YATES, R. A.; RIBEIRO-NETO, B. *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 020139829X.

44 MANNING, C. D.; RAGHAVAN, P.; SCHTZE, H. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN 0521865719, 9780521865715.

45 BERRY, M. W.; DRMAC, Z.; JESSUP, E. R. Matrices, vector spaces, and information retrieval. *SIAM Rev.*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 41, n. 2, p. 335–362, 1999. ISSN 0036-1445.