UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

JÚNIO CÉSAR DE LIMA

Seleção de Serviços Sensível à QoS e à Capacidade para Implantação Eficiente de Múltiplas Coreografias de Serviços

Goiânia
2018

sistema de bibliotecas ufg

PRPG
PRÓ-REITORIA DE PÓS-GRADUAÇÃO

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[ ] Dissertação** **[x] Tese**

2. Identificação da Tese ou Dissertação:

Nome completo do autor: Júnio César de Lima

Título do trabalho: Seleção de Serviços Sensível à QoS e à Capacidade para Implantação Eficiente de Múltiplas Coreografias de Serviços

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Júnio César de Lima[2]

Ciente e de acordo:

Fábio Moreira Costa[2]

Data: 20 / 12 / 2018

[1]Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.

Casos de embargo:
- Solicitação de registro de patente
- Submissão de artigo em revista científica
- Publicação como capítulo de livro
- Publicação da dissertação/tese em livro

[2]A assinatura deve ser escaneada.

JÚNIO CÉSAR DE LIMA

Seleção de Serviços Sensível à QoS e à Capacidade para Implantação Eficiente de Múltiplas Coreografias de Serviços

Tese apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Fábio Moreira Costa

Co-Orientador: Prof. Ricardo Couto Antunes da Rocha

Goiânia
2018

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

César de Lima, Júnio
Seleção de Serviços Sensível à QoS e à Capacidade para Implantação Eficiente de Múltiplas Coreografias de Serviços [manuscrito] / Júnio César de Lima. - 2018.
CXCI, 191 f.

Orientador: Prof. Fábio Moreira Costa; co-orientador Ricardo Couto Antunes da Rocha.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), , Goiânia, 2018.
Bibliografia. Anexos.
Inclui símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Seleção de Serviços . 2. Coreografia de Serviços. 3. Qualidade de Serviços. 4. Compartilhamento de Serviços. 5. Capacidade. I. Moreira Costa, Fábio, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

UFMS

Ata de Defesa de Tese de Doutorado

Aos vinte e três dias do mês de novembro de dois mil e dezoito, no horário das nove horas, foi realizada, nas dependências do Instituto de Informática da UFG, a defesa pública da Tese de Doutorado do aluno Júnio César de Lima, matrícula no. 2013 0350 , intitulada **"Seleção de Serviços Sensível a QoS e à Capacidade para Implantação Eficiente de Múltiplas Coreografias de Serviços"**.

A Banca Examinadora, constituída pelos professores:

Prof. Dr. Fábio Moreira Costa – INF/UFG - orientador

Prof. Dr. Ricardo Couto Antunes da Rocha – DCC-Catalão/UFG - coorientador

Prof. Dr. Edmundo Roberto Mauro Madeira – IC/Unicamp

Prof. Dr. Nelson Souto Rosa – CIn/UFPE

Prof. Dr. Humberto José Longo – INF/UFG

Prof. Dr. Wellington Santos Martins – INF/UFG

emitiu o resultado:

(X) Aprovado

( ) Aprovado com revisão

(A Banca Examinadora deve definir as exigências a serem cumpridas pelo aluno na revisão, ficando o orientador responsável pela verificação do cumprimento das mesmas.)

( ) Reprovado

com o seguinte parecer: o candidato deverá fazer os devidos ajustes no texto conforme apontados pela banca.

Prof. Dr. Fábio Moreira Costa

Prof. Dr. Ricardo Couto Antunes da Rocha

Prof. Dr. Edmundo Roberto Mauro Madeira

Prof. Dr. Nelson Souto Rosa

Prof. Dr. Humberto José Longo

Prof. Dr. Wellington Santos Martins

Júnio César de Lima

Possui graduação (2002) e mestrado (2006) em Ciência da Computação pelo Instituto de Informática (INF) da Universidade Federal de Goiás (UFG). Foi bolsista da Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG) durante o doutorado. Desde 2009 é professor efetivo do Instituto Federal de Educação, Ciência e Tecnologia Goiano – Câmpus Urutaí. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas Distribuídos, atuando principalmente nos seguintes temas: composição de serviços, seleção de serviços sensível à QoS e sistemas adaptativos.

À Deus e aos meus pais que foram os primeiros a acreditar que eu me tornaria um Doutor.

Agradecimentos

Acima de tudo, agradeço a Deus por mais esta conquista e por todas às oportunidades que me foram dadas, graças a sua benção.

À minha família, que em todos os momentos, me deram apoio incondicional. Em especial, agradeço meus pais, Luiz e Maria, pelas longas horas de oração, inclusive com lágrimas. Certamente só cheguei até aqui porque vocês foram os primeiros a acreditarem em mim. Agradeço também aos meus irmãos, Ézio e Waulene, e cunhados Luiz Cláudio e Viviane, por terem aberto mão em vários momentos da minha companhia para que eu pudesse me dedicar aos meus estudos, e a meus sobrinhos, Ítalo, Aline, Fernanda e João Pedro, que com certeza representam uma das grandes motivações de eu querer sempre melhorar.

A meus colegas do IF, Allan Kardec, Vanessa França, Paulo Mansur, Cristiane Santos, Mônica Sakuray, João Lopes, Júlio César, que me incentivaram a persistir nesta longa jornada. A meus amigos, Marciel, primo Leandro, tio Arcílio, tio Francisco, amigos do Pedra do Mar, que também sempre me apoiaram e que me ajudaram a lembrar que existe vida além do doutorado.

Aos meus orientadores, Dr. Fábio Costa e Dr. Ricardo da Rocha, pelos ensinamentos e direção nesta trajetória. Ao professor Dr. Humberto Longo pelas nossas longas conversas.

A todos que colaboraram para que este projeto fosse desenvolvido. A todos os colegas do Instituto de Informática da UFG, em especial Raphael Gomes, Leandro Alexandre e Thiago Borges, que vivenciaram as mesmas dificuldades que eu e com quem compartilho os bônus e ônus desta experiência.

Ao Instituto Federal de Goiano pela concessão do afastamento que, embora não tenha sido por todo o período do meu curso, permitiu que eu me dedicasse integralmente ao doutorado.

À Fundação de de Amparo à Pesquisa do Estado de Goiás (FAPEG) pela concessão de bolsa (Edital 003/2013, Processo 201310267000282), o que foi fundamental para que o trabalho fosse desenvolvido de maneira efetiva.

A todos que direta ou indiretamente contribuíram para a conclusão desta tese.

O sucesso é a soma de pequenos esforços repetidos dia após dia.

Robert Collier,
1885-1950.

Resumo

de Lima, Júnio César. **Seleção de Serviços Sensível à QoS e à Capacidade para Implantação Eficiente de Múltiplas Coreografias de Serviços**. Goiânia, 2018. 191p. Tese de Doutorado . Instituto de Informática, Universidade Federal de Goiás.

Coreografias representam uma abordagem para composição de serviços em que a coordenação é realizada de forma distribuída. Para implantação de uma coreografia deve ser selecionado um conjunto de serviços para desempenharem as funcionalidades requeridas na especificação da coreografia, inclusive assegurando os requisitos de QoS. No entanto, abordagens existentes para seleção de serviços sensível à QoS não consideram explicitamente o compartilhamento de serviços, pois lidam com cada coreografia isoladamente. Ao considerar coreografias isoladas, o processo de seleção de serviços pode tornar-se pouco eficiente em cenários reais, onde várias coreografias que concorrem pelo mesmo conjunto de serviços devem ser implantadas em conjunto. Nesse caso, um determinado serviço que atenda um papel em mais de uma coreografia pode ser compartilhado. O compartilhamento de serviços não supervisionado, no entanto, pode degradar a QoS global fornecida para as coreografias, já que a capacidade máxima dos serviços compartilhados pode ser excedida. Além disso, tais abordagens tendem a selecionar serviços com QoS mais alta do que o necessário, levando ao desperdício de recursos. Esta tese propõe uma abordagem para a seleção de serviços sensível à QoS e à capacidade para a implantação de múltiplas coreografias. Esta abordagem garante a satisfação dos requisitos de QoS, mesmo diante da possibilidade de compartilhamento de serviços. Para este fim, é proposto um modelo para representação combinada de um conjunto de coreografias. Tal modelo é utilizado como entrada para a seleção de serviços, que é resolvida buscando um emparelhamento entre os papéis das coreografias e os serviços candidatos de forma a minimizar os custos dos serviços selecionados associados à utilização de recursos. Para isso, é proposta uma função de utilidade para avaliar a QoS dos serviços, juntamente com a extensão do algoritmo de emparelhamento. A tese apresenta uma arquitetura que combina todos os elementos da abordagem proposta. Uma implementação do protótipo da arquitetura foi desenvolvida para possibilitar sua avaliação. Os resultados da avaliação indicam eficácia e desempenho superior da abordagem proposta em relação aos trabalhos relacionados utilizados para comparação.

Palavras–chave

Seleção de Serviços, Coreografia de Serviços, QoS, Compartilhamento de Serviços, Capacidade

Abstract

de Lima, Júnio César. **QoS- and Capacity-Aware Service Selection to Efficient Deployment of Multiple Services Choreographies**. Goiânia, 2018. 191p. Ph.D. Thesis . Instituto de Informática, Universidade Federal de Goiás.

Choreographies are an approach for service composition in which coordination is performed in a decentralized way. To deploy a choreography, a set of services must be selected to perform the functionalities required in its specification, including ensuring its QoS requirements. However, existing approaches for QoS-aware service selection fail to explicitly consider service sharing, as they deal with each choreography in isolation. By dealing with a single choreography at a time, the service selection process may become less feasible in real scenarios, in which several choreographies, competing for the same set of services, must be deployed together. In this case, a given service that suits a role in more than one choreography may be shared. Unsupervised service sharing, however, may degrade the overall QoS provided for the choreographies, as the maximum capacity of the shared services may be exceeded. In addition, such approaches tend to select services with higher QoS than necessary, leading to waste of resources. This thesis proposes an approach for QoS- and capacity-aware service selection for the combined deployment of multiple choreographies. This approach ensures the satisfaction of QoS requirements, even in the face of possible service sharing. To this end, we propose a model for the combined representation of multiple choreographies. This model is used as input for the service selection, which is solved by seeking a matching between of the choreographies roles and the candidate services to minimize the costs of the selected services in terms of resource usage. For this, a utility function is proposed to evaluate the QoS of the services, along with the extension of the matching algorithm. The thesis presents an architecture that combines all the elements of the proposed approach. A prototype implementation of the architecture was developed to enable its evaluation. The results of the evaluation indicate superior effectiveness and performance of the proposed approach as compared to related work.

Keywords

Service Selection, Service Choreography, QoS, Sharing of Services, Capacity

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Códigos de Programas

CAPÍTULO 1

Introdução

Coreografias de serviços são composições de serviços que implementam aplicações distribuídas sem a necessidade de um coordenador centralizado, reduzindo assim o número de mensagens de controle trocadas e simplificando a lógica de distribuição [10]. Geralmente, os serviços que compõem uma coreografia podem ser descritos de maneira abstrata, devido ao uso do papel esperado pelo serviço na composição. Estes papéis podem então ser atribuídos a responsáveis usando entidades concretas, isto é, identificando uma implementação compatível para cada serviço.

O processo de desenvolvimento de uma coreografia compreende duas etapas [152]. Inicialmente, são identificadas as funcionalidades exigidas para cada papel (tarefas a serem realizadas pela coreografia) e seu inter-relacionamento com os demais papéis. Em seguida, serviços que são capazes de fornecer as funcionalidades requisitadas são selecionados e vinculados a cada papel na coreografia. Um papel pode ser desempenhado por diversos serviços, desde que estes implementem os requisitos funcionais esperados. Quando um usuário envia uma mensagem para uma coreografia, os serviços selecionados iniciam o processo de troca de mensagens a fim de desempenhar a lógica de negócio descrita pela coreografia. Nesse caso, diz-se que a coreografia é encenada [47].

A seleção de serviços para os papéis de uma coreografia pode ser orientada também pelos requisitos não-funcionais, como qualidade de serviço (*Quality of Service* – QoS). Esses requisitos de QoS podem ser aplicados à coreografia como um todo ou a serviços específicos da coreografia. De fato, os requisitos de QoS dos usuários desempenham um papel importante no processo de seleção de serviços [126].

Os requisitos de QoS são usualmente negociados entre os provedores de serviços e os usuários, com base em termos e condições relacionados à QoS que um serviço deve entregar. Usualmente, essa negociação produz um Acordo de Nível de Serviço (*Service Level Agreement* – SLA), que também define responsabilidades e condições em caso de falhas no acordo. O SLA especifica os níveis mínimos de desempenho que um provedor de serviços deverá manter e as penalidades, estipuladas contratualmente caso o não cumprimento do acordo.

Muitas abordagens existentes utilizam SLAs como especificação dos requisitos de QoS [67, 124, 91]. Tais abordagens definem SLAs com base nos requisitos dos usuários, juntamente com as penalidades caso ocorram violações de QoS. Por outro lado, a seleção de serviços pode ser utilizada de forma a garantir os requisitos, selecionando serviços que evitam violações de QoS. Caso a QoS oferecida por um serviço não satisfaça a QoS requisitada pelo usuário, uma nova seleção de serviços pode ser realizada, evitando o uso de penalidades e a insatisfação do usuário.

Em cenários onde várias coreografias devem ser encenadas em um mesmo ambiente de execução, diferentes coreografias podem conter papéis em comum. Comércio eletrônico e plataformas de *streaming* são exemplos de cenários onde coregrafias de serviços podem ser utilizadas para implementar processos de negócios complexos. Neste caso, um mesmo serviço pode ser compartilhado quando desempenha um papel presente em mais de uma coreografia. Nessa situação, abordagens que sempre selecionam serviços com os melhores valores para os requisitos de QoS podem levar ao excessivo compartilhamento de um subconjunto de serviços.

O compartilhamento de serviços pode causar a degradação da QoS. A degradação da QoS ocorre porque na implementação de cada serviço é realizada a alocação de recursos computacionais que consigam garantir a QoS ofertada de acordo com uma capacidade de processamento. Caso a carga agregada das coreografias supere a capacidade de um serviço compartilhado, haverá um aumento do número de violações de QoS. Por este motivo, uma abordagem de seleção de serviços também deve ser sensível à capacidade dos serviços.

Os usuários geralmente não precisam de serviços com a melhor QoS possível, uma vez que exceder a demanda normalmente não melhora a satisfação dos usuários [139]. Por exemplo, um usuário que deseja reservar um táxi para ir a um centro de compras, não necessita de um serviço de reserva que tenha o melhor tempo de resposta, ou seja, o usuário pode ficar satisfeito com serviços de reserva com um tempo um pouco superior. Sendo assim, serviços com um alto nível de QoS devem ser oferecidos para usuários que tenham requisitos de QoS mais severos.

Além disso, serviços com a melhor QoS tendem a necessitar de mais recursos computacionais, gerando um maior custo pela sua utilização. A seleção de serviços sensível à QoS contribui para a minimização do desperdício de recursos ao evitar a subutilização dos serviços selecionados. Essa subutilização pode ocorrer devido ao superdimensionamento dos recursos utilizados pelos serviços selecionados, uma vez que serviços com alto consumo computacional podem ser oferecidos para coreografias com requisitos de QoS que não demandem tanto.

Um dos problemas tratados nesta tese é a escolha de uma estratégia de seleção de serviços que torne o uso dos serviços mais eficiente. Uso eficiente de serviços

implica em uma melhor utilização dos recursos onde os serviços são executados, uma vez que os recursos alocados para execução de cada serviço são realmente utilizados. Sendo assim, em um ambiente onde múltiplas coreografias, com possível compartilhamento de serviços, podem ser encenadas ao mesmo tempo, o sucesso da encenação de cada coreografia depende de uma estratégia eficiente de seleção de serviços. Para isso, a seleção de serviços deve considerar o impacto da carga agregada resultante na QoS fornecida pelos serviços compartilhados. Esta tese investiga como selecionar serviços sensíveis à QoS e à capacidade para implantação de múltiplas coreografias. O objetivo da seleção é a satisfação dos requisitos dos usuários ao mesmo tempo que minimiza o custo em relação ao uso de recursos sobre os quais os serviços são implantados, uma vez que o custo é diretamente relacionado a QoS fornecida por um serviço.

1.1 Escopo da tese

A Figura 1.1 ilustra o processo de construção de aplicações utilizando coreografias de serviços.

O Passo 1 a ser realizado é a especificação das coreografias de serviços utilizando uma linguagem apropriada, como BPMN2 [99] e WS-CDL [53], que deve permitir descrever os papéis que compõem a coreografia e o modo com que cada papel interage com os outros. Além disso, informações sobre requisitos não-funcionais, como QoS, também podem ser especificadas para os papéis individualmente ou para a coreografia como um todo. Neste caso, uma linguagem de especificação de coreografia deve ser capaz de representar os requisitos não-funcionais para uma coreografia, bem como informações sobre a carga estimada para a mesma. Essa atividade pode ser realizada por um especialista do domínio de negócio, que assume o papel do provedor de coreografias. No final, a especificação da coreografia é submetida para um *motor de execução* capaz de interpretar a linguagem utilizada.

O Passo 2 é a seleção dos serviços que irão desempenhar os papéis em cada coreografia. A seleção de serviços é o componente principal deste processo e provê suporte para garantir tanto a satisfação dos provedores de coreografias quanto dos usuários. O motor de execução ao receber uma especificação de coreografia utiliza algum mecanismo interno ou externo para realizar a seleção. Este mecanismo deve realizar a seleção dos serviços para cada papel da coreografia, considerando um conjunto conhecido de serviços, juntamente com informações de suas propriedades funcionais e não-funcionais. Ao final da seleção, o motor de execução recebe as informações dos serviços selecionados para cada papel e implanta a lógica de execução da coreografia de acordo com os serviços selecionados. Já no passo 3, os usuários podem requisitar a

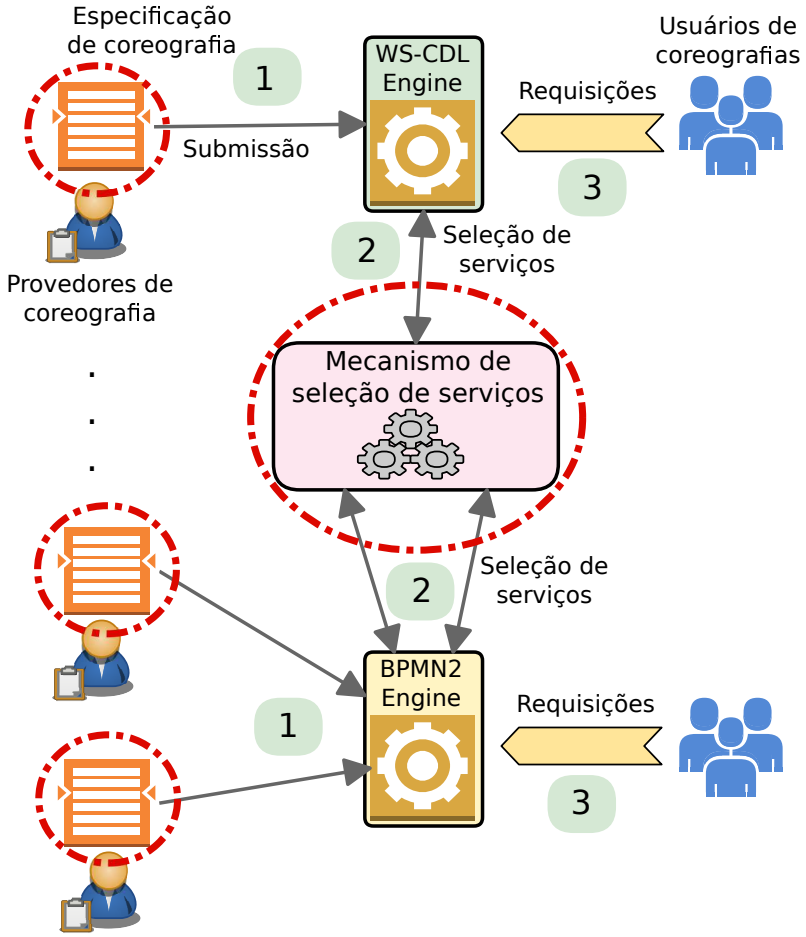


Figura 1.1: *Processo de construção de aplicações utilizando coreografias de serviços. As caixas em destaque indicam o escopo desta tese.*

execução de uma coreografia de serviços, o que resulta em sua encenação pelo motor de execução.

O foco desta tese é a seleção de serviços, onde apresentamos modelos para representação interna das especificações das coreografias anotadas com requisitos de QoS e métodos para serem utilizados na seleção de serviços para um grupo de coreografias. Argumentamos que a seleção de serviços sensível à QoS deve levar em consideração que várias coreografias podem ter papéis em comum, assim como o possível compartilhamento de serviços entre elas, de forma a garantir a QoS requisitada e minimizar o custo em relação ao uso de recursos sobre os quais os serviços são implantados. Já a implantação dos serviços selecionados e posterior encenação de coreografias de serviços pelo motor de execução estão fora do escopo desta tese.

1.2 Definição do problema

Esta tese investiga o seguinte problema:

> *Dado um conjunto de especificações de coreografias de serviços cujos papéis possuam requisitos de QoS e que possam ser comuns a diferentes coreografias, como selecionar um conjunto de serviços capaz de satisfazer os requisitos de QoS ao mesmo tempo que se busca minimizar os custos relacionados ao uso de recursos por tais serviços para garantir a QoS oferecida?*

Há diversos trabalhos que propõem seleção de serviços sensível à QoS para implantação de coreografias de serviços [153, 6, 7, 139, 80]. Entretanto, tais propostas não levam em consideração situações em que a seleção de serviços envolve múltiplas coreografias e nem lidam com a eficiência de utilização dos recursos associados aos serviços.

Nesses trabalhos, a seleção de serviços é realizada isoladamente para cada coreografia. Nesse caso, um mesmo serviço que satisfaz um papel comum a várias coreografias pode ser selecionado para atender a diversas coreografias, causando degradação da QoS oferecida pelo serviço. Ao mesmo tempo, serviços que oferecem a melhor QoS podem ser selecionados para coreografias com requisitos de QoS menos exigentes, levando à subutilização dos serviços e, por consequência, ao desperdício de recursos.

Ao desenvolver esse tema de pesquisa, objetivamos propor uma solução que seja capaz de lidar, no mesmo processo de seleção, com a minimização do custo para os provedores de coreografias em relação aos recursos associados às instâncias de serviços ao mesmo tempo que a QoS requisitada é garantida. A solução proposta nesta tese considera desafios adicionais relacionados à seleção de serviços para múltiplas coreografias de serviços, como a necessidade de satisfazer simultaneamente uma quantidade maior de requisitos de QoS e a lidar com questões relacionadas ao possível compartilhamento de serviços entre elas.

1.3 Exemplo de cenário da tese

Para ilustrar os principais conceitos envolvidos na seleção de serviços sensível à QoS para aplicações baseadas em coreografias de serviços, apresentamos o cenário considerado nesta tese. Este cenário consiste de aplicações utilizadas no domínio de Cidade Inteligente.

Para atingir seus objetivos, um ambiente de Cidade Inteligente deve fornecer um grande conjunto de serviços públicos e/ou privados que funcionem de forma integrada, acessível e sustentável, como tratamento e distribuição de água, transporte público, controle de tráfego e distribuição de energia [70]. Os serviços oferecidos podem ser modelados como processos de negócio e implementados como coreografias de serviços, permitindo a interação estruturada e distribuída entre diferentes serviços.

Nesse sentido, o contexto de Cidade Inteligente pode ser visto como um exemplo típico de ambiente onde múltiplas coreografias de serviços devem fornecer funcionalidades personalizadas para diversos usuários [18]. Cada etapa de uma coreografia corresponde a um papel esperado com seus respectivos requisitos. Por exemplo, como parte de uma coreografia para reserva de táxi são necessários diferentes papéis, tais como: serviço de tráfego, serviço de localização e serviço de pagamento. Além disso, podem ser definidos requisitos de QoS sobre a coreografia ou papéis individuais, como o tempo de resposta máximo para uma coreografia, que deve ser menor do que um certo valor alvo, ou a vazão mínima para um serviço, em termos do número mínimo de requisições por segundo que o serviço deve ser capaz de processar.

O cenário é apresentado a seguir.

"*Na próxima semana, vai ocorrer um congresso na cidade de Amsterdã. Os diversos participantes necessitam realizar a reserva do hotel e comprar a passagem para os dias da reserva. Ao chegar na cidade, os participantes necessitam realizar diversas atividades, como procurar por restaurante e locomover-se entre o hotel e o local do congresso ou algum ponto turístico.*"

A seguir, usamos essas atividades para enfatizar os principais conceitos relacionados à seleção de serviços sensível à QoS para coreografias de serviços.

Quando um participante vai organizar sua ida ao congresso, ele necessita contactar vários hotéis da região do evento e verificar se há disponibilidade de reserva na data requerida com preços e serviços de acordo com o orçamento, o que representa um processo longo e difícil. Em um cenário em que Cidade Inteligente é considerado, existe um ambiente próprio que oferece vários serviços (por exemplo, reserva de hotel, reserva de transporte, calendário e pagamento) para os turistas organizarem suas atividades.

Dado o grande número de turistas que Amsterdã recebe anualmente, o ambiente de Cidade Inteligente deve ser capaz de oferecer serviços para atender aos variados perfis de turistas. Por exemplo, durante sua estadia, os turistas desejam buscar por restaurantes próximos do hotel, onde é necessários coreografar vários serviços, como serviços de localização, restaurante, rota, transporte e pagamento.

Ao final da participação no congresso, os participantes podem querer buscar as melhores rotas entre o hotel e os pontos turísticos. Para tanto, o ambiente de Cidade Inteligente disponibiliza coreografias de serviços específicas para serviço de tráfego, permitindo que os usuários adicionem novas funcionalidades de forma simples, como condições meteorológicas do trajeto e informações em tempo real da infraestrutura da cidade.

Nesse cenário podem existir vários turistas que requisitam a coreografia de tráfego, de acordo com as preferências de cada um. Além disso, os moradores da cidade precisam acessar o serviço de tráfego para ir ao trabalho, escola ou compras. Ao mesmo tempo, serviços essenciais à cidade, como polícia, socorrista e corpo de bombeiros, também precisam desse serviço com maior prioridade em relação à população em geral. Com base nas necessidades de cada perfil de usuário, o ambiente de Cidade Inteligente deve realizar a seleção de serviços para cada papel da coreografia ao mesmo tempo em que garante os requisitos de qualidade solicitado.

As especificações dessas coreografias de serviços são criadas pelos provedores de coreografia (por exemplo, a equipe de TI de uma organização), que também são responsáveis pela sua implantação de forma a torná-las disponíveis para os usuários. Como parte do processo de implantação, um serviço apropriado deve ser selecionado para realizar cada papel em cada coreografia.

Como podem existir vários serviços de tráfego urbano operando simultaneamente com diferentes ofertas de QoS, o mecanismo de seleção de serviços deve selecionar um serviço que atenda aos requisitos de QoS dos usuários. Por exemplo, selecionar um serviço com tempo de resposta menor do que 10 milissegundos e com um certo nível de autenticação. A quantidade de serviços similares, de distintos provedores e com diferentes níveis de QoS, varia de acordo com o tipo de serviço. Por exemplo, Mazza *et al.* [88] descreveu um cenário que oferece coreografias de serviços para pessoas em centros de compras contendo 100 serviços de mapa, 1000 serviços de meteorologia e 1000 serviços de relatórios de condições de tráfego. Além disso, o processo de seleção deve ser capaz de analisar a carga agregada estimada para serviços compartilhados por várias coreografias, uma vez que coreografias que possuem papéis em comum podem selecionar um mesmo serviço para desempenhar um mesmo papel.

Os serviços candidatos podem ser oferecidos por diversos provedores, podendo existir, inclusive, concorrência entre os provedores, que podem oferecer serviços diferenciados em relação à qualidade e ao custo. Para cada serviço candidato é informada a descrição de suas funcionalidades, a QoS ofertada e a capacidade máxima de processamento de requisições que o serviço consegue garantir com a QoS ofertada. A oferta de QoS pode ser informada diretamente pelo provedor do serviço. Já a informação da capacidade de um serviço candidato pode ser obtida a partir de algum

mecanismo de *profiling* que mede a capacidade à medida que o serviço é usado ou através de testes de desempenho extensivos [41].

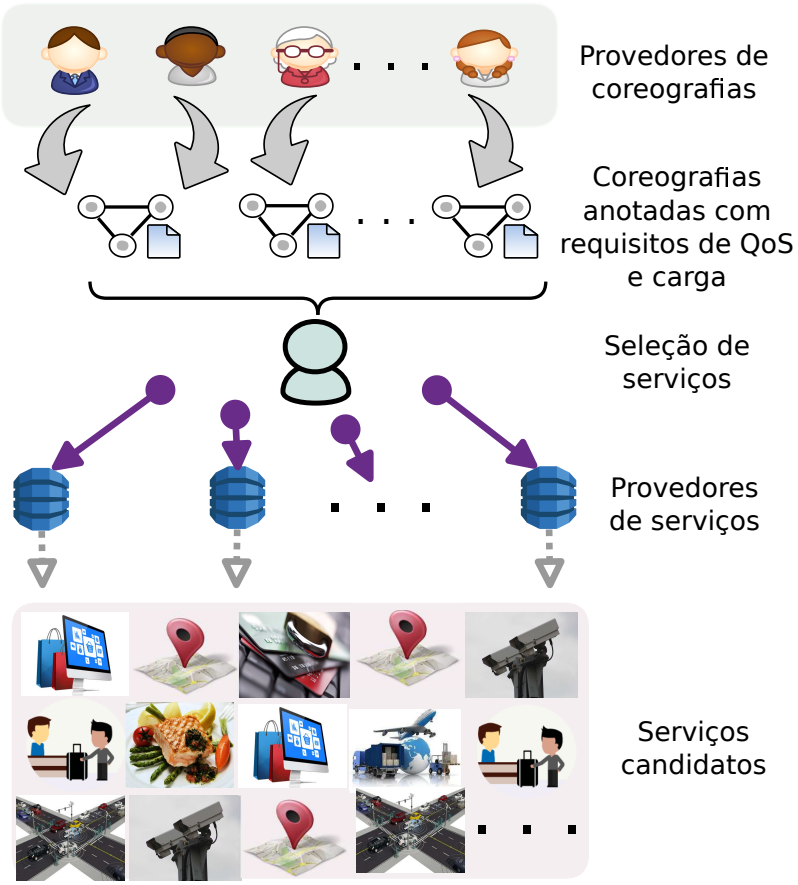


Figura 1.2: *Cenário considerado na seleção de serviços sensível à QoS e à capacidade.*

A Figura 1.2 ilustra um cenário onde a seleção de serviços deve ser capaz de fornecer os serviços adequados para vários provedores de coreografias. A seleção de serviços deve considerar a existência de papéis comuns entre coreografias e o impacto da carga agregada resultante na QoS que é de fato fornecida pelos serviços. Como a seleção de serviços deve garantir a QoS requisitada por cada provedor de coreografias, o principal desafio desse cenário refere-se ao fato de lidar com o possível compartilhamento de serviços. Como efeito colateral desse compartilhamento, a carga agregada resultante pode causar degradação da QoS, levando ao aumento no número de violações da QoS requisitada.

Os provedores de coreografias, além de garantir a satisfação dos requisitos de QoS dos usuários, também desejam minimizar os custos relacionados aos recursos onde os serviços são executados. Um serviço possui um custo baseado na QoS ofertada e na capacidade máxima contratada, uma vez que o provedor deve garantir a QoS acordada até o limite da capacidade [132]. Por exemplo, dado dois serviços de pagamento, serviço A e serviço B, que possuem o tempo de resposta em *10ms*, mas o serviço

A possui uma capacidade de 10 requisições por segundo, enquanto o serviço B possui uma capacidade de 30 requisições por segundo, logo o serviço B possui um custo mais elevado do que o serviço A. Neste sentido, o mecanismo de seleção de serviços, além de analisar o tempo de resposta, deve levar em consideração a capacidade dos serviços diante da carga estimada para cada coreografia.

Como no cenário é oferecido um conjunto de coreografias de forma a atender um grande número de usuários, os provedores de coreografias podem se juntar para negociar melhores condições junto aos provedores de serviços. Nesse modelo de negócio de economia compartilhada, quanto mais coreografias são disponibilizadas, mais usuários são atraídos e, por consequência, aumenta a chance de compartilhamento de serviços. Tal modelo é amplamente adotado em diversos segmentos como, por exemplo, compartilhamento de transporte [26, 32]. Esse modelo é condizente com o cenário descrito acima, uma vez que organizações públicas ou privadas podem oferecer várias coreografias às pessoas que circulam por uma Cidade Inteligente. O objetivo desse tipo de organizações é oferecer serviços de acordo com as expectativas dos usuários ao mesmo tempo que são reduzidos os custos.

A seleção de serviços deve garantir a satisfação dos requisitos dos usuários e a redução dos custos para os provedores de coreografias na implantação conjunta de múltiplas coreografias de serviços. A partir dessa observação, surge a motivação para esta pesquisa, que visa estabelecer uma abordagem para seleção de serviços sensível à QoS e à capacidade que atenda as necessidades dos usuários enquanto cumpre os requisitos de QoS e minimiza o desperdício de recursos associados aos serviços selecionados.

1.4 Abordagem proposta

O processo geral de seleção de serviços para implantação conjunta de múltiplas coreografias, conforme proposto nesta tese, é ilustrado na Figura 1.3. Inicialmente, o provedor de coreografias especifica uma ou mais coreografias de serviços para atender diferentes processos de negócio. Um processo de negócio baseado em serviços pode descrever, por exemplo, os passos para reservar uma passagem aérea ou as etapas para buscar uma rota entre dois pontos. Tais processos são tipicamente criados para atender perfis de usuários em domínios particulares.

Em seguida, o provedor de coreografias submete uma ou mais especificações de coreografias de serviços anotadas com informações de requisitos de QoS, com base nas expectativas dos usuários. Além dos requisitos de QoS, informações de carga também são anotadas. A carga é estimada com base na quantidade prevista de usuários. Essas especificações podem ser representadas em qualquer linguagem para modela-

gem de coreografias de serviços. Vale ressaltar que diferentes provedores de coreografias podem enviar suas especificações ao mesmo tempo. Isto significa que a seleção de serviços deve ser capaz de fornecer os serviços adequados para vários provedores de coreografias.

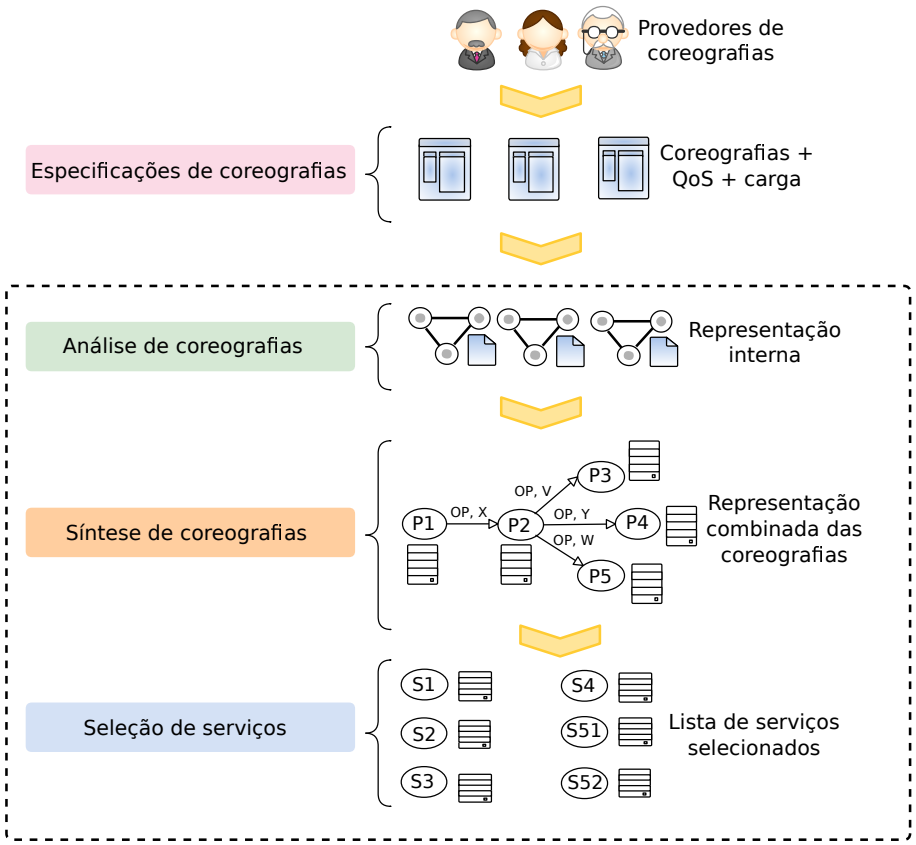


Figura 1.3: *Visão geral da abordagem proposta.*

Considerando o ciclo de vida de uma coreografia de serviços, conforme proposição de Sheng *et al.* [114], a abordagem proposta nesta tese lida com a fase de seleção de serviços. O primeiro passo da abordagem, ao receber especificações de coreografias de serviços, é realizar a *análise de coreografias*. Inicialmente cada coreografia é convertida em uma representação interna, denominada de *grafo de processo*, que será formalizada no Capítulo 5. Além disso, cada grafo de processo é avaliado para decompor os requisitos de QoS e carga estimada sobre uma coreografia entre os papéis que dela participam. Sendo assim, cada coreografia é representada internamente como um grafo de processo, onde para cada papel é anotado o requisito de QoS e a carga estimada.

O segundo passo da abordagem consiste em realizar a *síntese de coreografias*. Neste passo, as coreografias representadas como grafos de processo são sintetizadas em um único modelo denominado de *grafo de dependências*, que será formalizado no Capítulo 5. Esta representação combinada visa permitir uma análise global dos

requisitos de várias coreografias, a fim de realizar a seleção de serviços sensível à QoS de forma a analisar o possível compartilhamento de serviços.

O terceiro e último passo da abordagem refere-se à *seleção de serviços*, que é descrita em detalhes no Capítulo 6. A abordagem proposta considera o problema de seleção de serviços como um problema de emparelhamento [93], no qual o objetivo é emparelhar os papéis que compõem as coreografias com os serviços candidatos, analisando os requisitos de QoS e a carga dos papéis, bem como a oferta de QoS e capacidade dos serviços candidatos.

1.5 Contribuições

As principais contribuições desta tese são:

i) Um modelo de coreografia de serviços que permite sintetizar a representação combinada de múltiplas coreografias;

ii) Uma estratégia para avaliar e ranquear serviços candidatos de acordo com a QoS ofertada. Essa estratégia leva em consideração os requisitos de QoS específicos de cada provedor de coreografias buscando tornar o uso dos serviços mais eficiente;

iii) Uma abordagem para seleção de serviços sensível à QoS e ao custo. Essa abordagem leva em consideração os requisitos de QoS dos usuários, bem como a carga de trabalho sobre papéis comuns e a capacidade dos serviços candidatos de forma a garantir a QoS ofertada e minimizar os custos relacionados aos recursos alocados para os serviços;

iv) Uma arquitetura que permite integrar, de maneira conjunta e coordenada, as abordagens e os modelos propostos para seleção de serviços. A arquitetura possibilita definir diferentes níveis de abstração para as coreografias de serviços, permitindo que a seleção de serviços seja realizada através da análise combinada de um conjunto de coreografias.

1.6 Estrutura da tese

A seguir é apresentada uma visão geral dos Capítulos que compõem as partes desta tese.

Capítulo 2: Conceitos fundamentais para seleção de serviços para coreografias. O capítulo apresenta os conceitos que fundamentam o trabalho desenvolvido

e que são essenciais para o seu entendimento. Particularmente, os conceitos de coreografias de serviços, requisitos não-funcionais sobre serviços e seleção de serviços sensível à QoS.

Capítulo 3: Trabalhos relacionados. O capítulo descreve as abordagens mais relevantes que foram propostas na literatura para seleção de serviços sensível à QoS. Por fim, o capítulo identifica as principais lacunas existentes na literatura sobre seleção de serviços sensível à QoS para implantação de múltiplas coreografias de serviços.

Capítulo 4: Modelo de QoS e custo de serviços. O capítulo apresenta a formalização do modelo de representação dos serviços juntamente com os critérios de QoS. Além disso, este capítulo formaliza a função de utilidade, para classificar os serviços candidatos, e o modelo de custo de serviços adotado nesta teste.

Capítulo 5: Representação de múltiplas coreografias de serviços com requisitos de QoS. O capítulo apresenta a notação para a representação interna de coreografias de serviços anotadas com requisitos de QoS e carga. O capítulo também define a estrutura para a representação combinada de múltiplas coreografias de serviços, juntamente com os requisitos de QoS e carga estimada.

Capítulo 6: Síntese de serviços. O capítulo apresenta a abordagem proposta para o problema de seleção de serviços sensível à QoS e à capacidade para a implantação eficiente de múltiplas coreografias de serviços. Além disso, o capítulo apresenta estratégias para lidar com os dois objetivos conflitantes da síntese de serviços: satisfazer os requisitos de QoS dos usuários e minimizar os custos dos serviços selecionados para as coreografias.

Capítulo 7: Arquitetura e implementação. O capítulo propõe uma arquitetura para implementação das estratégias propostas para seleção de serviços sensível à QoS e à capacidade. Além disso, o capítulo também discute a implementação de um protótipo usado para avaliar a abordagem proposta.

Capítulo 8: Avaliação. O capítulo apresenta uma avaliação quantitativa da abordagem proposta nesta tese, com o objetivo de demonstrar sua qualidade e eficiência, bem como compará-la com outras estratégias.

Capítulo 9: Conclusões. Este capítulo descreve em perspectiva a tese como todo, as principais conclusões e contribuições do trabalho, e apresenta perspectivas

futuras para o aprimoramento da abordagem proposta nessa tese, levando em consideração suas principais limitações e problemas em aberto.

CAPÍTULO 2

Conceitos fundamentais para seleção de serviços para coreografias

Este capítulo apresenta os fundamentos conceituais utilizados nesta tese e está organizado da seguinte forma. Na Seção 2.1, são apresentadas as principais características e desafios de composições de serviços, com particular ênfase em coreografias de serviços. Em seguida, é apresentada a terminologia de coreografias de serviços adotada nesta tese. Esta terminologia é uma extensão daquela apresentada em [52]. Em complemento, são abordados o ciclo de vida e a modelagem de coreografias de serviços. Por fim, é apresentado um exemplo de modelagem de coreografias de serviços com BPMN2.

A Seção 2.2 discute algumas definições relacionadas com requisitos não-funcionais, destacando as particularidades dessas definições no contexto de coreografias de serviços. Por fim, na Seção 2.3 são descritas as principais técnicas utilizadas no processo de seleção de serviços para implantação de coreografias, destacando os conceitos de qualidade de serviço no contexto de seleção de serviços sensível à QoS.

2.1 Coreografia de serviços

Processos de negócios complexos exigem múltiplas funcionalidades, que podem ser realizadas por grupos de serviços na forma de composições de serviços [114]. Composição de serviços tende a reduzir os custos e os riscos na construção de novas aplicações através da reutilização de serviços. De forma geral, uma composição de serviços é uma combinação de múltiplos serviços, que podem ser oferecidos por provedores diferentes e em locais diferentes na rede, para executar um conjunto de atividades que satisfaçam um objetivo específico [14].

Papazoglou e Dubray argumentam que o principal objetivo da *computação orientada a serviços* é permitir composição de aplicações distribuídas personalizadas de acordo com as demandas dos negócios e requisitos dos usuários [102]. Por este motivo, a construção de uma composição de serviços é essencialmente uma atividade de

programação distribuída [31], onde o desenvolvimento da aplicação envolve atividades de identificação, seleção e composição dos serviços oferecidos por terceiros [22].

Aplicações construídas como composições de serviços são especificadas como processos abstratos compostos, os quais, por sua vez, são constituídas por um conjunto de papéis, que definem as atividades que devem ser desempenhadas pelos serviços. Em tempo de execução, cada papel definido em uma coreografia deve ser desempenhado por um serviço previamente selecionado, assegurando os requisitos dos usuários [21]. Uma composição de serviços é tipicamente opaca para o usuário, que a considera como um serviço único e não um conjunto de serviços distintos [49].

Há dois caminhos para descrever a sequência de atividades de uma composição de serviços: centralizado ou distribuído. No modelo centralizado, chamado de orquestração, a composição de serviços tem um coordenador central que organiza o fluxo de controle da composição, conforme ilustrado na Figura 2.1 (a) [95].

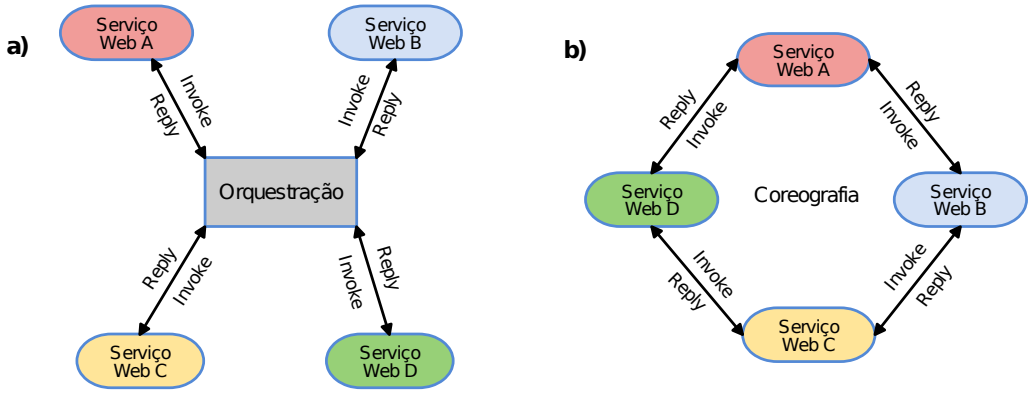


Figura 2.1: *Orquestração de serviços (a) e coreografia de serviços (b) [114].*

Por outro lado, no modelo distribuído de composição, chamado de coreografia de serviços, não há um coordenador central, como ilustrado na Figura 2.1 (b). Neste modelo, o conhecimento do fluxo de controle é distribuído entre os serviços participantes. Deste modo, uma coreografia de serviços fornece uma perspectiva global de colaborações existentes, significando que todos os participantes são tratados igualmente, de forma *peer-to-peer*, diferente da perspectiva local fornecida pela orquestração, em que uma única parte controla o fluxo de mensagens [10].

Uma coreografia representa uma descrição global do comportamento observável de cada um dos serviços que participam da interação, que é definido pela troca pública de mensagens e regras de interação [114]. Assim, cada serviço envolvido na composição executa a operação necessária ao receber uma mensagem do serviço responsável pela operação anterior, repassando o resultado para o próximo serviço da interação, sem a necessidade de controle centralizado [10].

Coreografias de serviços estabelecem os protocolos de coordenação para troca de mensagens entre os vários serviços, sendo que nenhuma das partes tem controle completo das mensagens [147]. Como as coreografias não tem a necessidade de um coordenador centralizado, consegue-se reduzir o número de mensagens de controle trocadas, simplificando a lógica de distribuição [10].

Sendo assim, coreografias descrevem as interações colaborativas entre vários serviços, enquanto a orquestração representa um controle centralizado da perspectiva de um dos serviços. Isso significa que uma coreografia difere de uma orquestração com relação a implantação da lógica que controla as interações entre os serviços envolvidos [143].

2.1.1 Terminologia

Serviços *Web*, ou simplesmente ***serviços***, são entidades de software autocontidas que executam uma ou mais operações, cujas interfaces são publicadas, e que oferecem suporte para interações diretas com outras aplicações [127]. Uma ***operação*** estabelece um tipo de requisição que pode ser feita por um usuário a um serviço, as ações executadas pelo respectivo serviço e a resposta gerada. Cada operação requer uma quantidade de poder computacional para ser processada.

A combinação de vários serviços por meio de suas operações forma uma ***composição de serviços***. Uma ***coreografia de serviços*** é uma forma de composição de serviços onde o protocolo de interação entre os serviços é definido de forma distribuída. Uma coreografia é descrita por meio dos papéis que se espera que cada serviço desempenhe na composição. Um ***papel*** define uma atividade de uma coreografia, especificando o comportamento esperado de cada participante que irá executá-lo em termos da sequência e periodicidade das mensagens que ele pode produzir e consumir [119].

Cada papel usado na definição de uma coreografia é desempenhado por um ***serviço concreto***, previamente selecionado dentre um conjunto de ***serviços candidatos***. Um serviço candidato é uma implementação de um serviço capaz de satisfazer os requisitos funcionais de um papel. Serviços candidatos funcionalmente equivalentes compartilham as mesmas interfaces para oferecer suas operações. Serviços candidatos que compartilham a mesma interface são descritos em classes de equivalência, chamadas de ***tipo de serviço***. Os serviços candidatos pertencentes a um tipo de serviço, embora implementem a mesma interface podem fazê-lo com diferentes ofertas de QoS.

Assumimos que todos os serviços candidatos são *stateless*, ou seja, não exibem estado observável. Serviços *stateless* proveem uma abstração funcional, onde um mesmo serviço fornece resultados idênticos para requisições com os mesmos argumentos. Logo, o comportamento funcional das operações pode ser especificado por

um contrato e o comportamento em tempo de execução pode ser monitorado isoladamente [40]. Em contraste, em um serviço *stateful* a resposta a uma operação depende não apenas dos argumentos de entrada, mas também do estado interno do serviço, necessitando manter o histórico das requisições [12]. Essa decisão de coreografar apenas serviços *stateless* é baseada em vários trabalhos na literatura, como em [1, 48, 84].

O termo ***provedor de coreografias*** refere-se às entidades responsáveis pela definição da lógica da aplicação e requisitos de QoS com base nas expectativas dos usuários. ***Usuário*** é uma entidade que requisita a execução de uma coreografia. Ao requisitar a execução de uma coreografia, os serviços participantes devem desempenhar seus respectivos papéis. Nesse momento, diz-se que a coreografia é ***encenada***.

Uma ***aplicação*** é um software desenvolvido com o propósito de auxiliar um usuário a desempenhar uma tarefa específica, como por exemplo, agendamento de passagens áreas. Nesta tese, assumimos que as aplicações são coreografias de serviços fornecidas no contexto de software como serviço (*Software as a Service* – SaaS), onde um usuário requisita uma aplicação via Internet, pagando um valor pelo serviço [138]. O termo ***recurso*** utilizado na tese se refere aos recursos computacionais alocados em provedores de computação em nuvem necessários para executar cada serviço candidato fornecido por um provedor de serviço.

2.1.2 Ciclo de vida de coreografias de serviços

O ciclo de vida de uma coreografia de serviços sensível à QoS, como ilustrado na Figura 2.2, pode incluir quatro fases:

- **Especificação**: o provedor de coreografias especifica a estrutura (ordem em que os serviços interagem) e os requisitos funcionais e não-funcionais da coreografia de serviços. O resultado desta fase é um modelo de processo abstrato que especifica um conjunto de papéis, o fluxo de controle e de dados e os requisitos de QoS [94].
- **Seleção de serviços**: nesta fase são identificados os serviços capazes de satisfazer funcionalmente os papéis especificados na coreografia. Os serviços candidatos são localizados pesquisando-se o registro de serviços, com base nas informações fornecidas pelos provedores de serviços. Uma vez que vários serviços candidatos podem desempenhar os requisitos funcionais de um papel, nesta fase é selecionado um serviço concreto para cada papel, de forma que os requisitos de QoS dos usuários também sejam garantidos. Como resultado, esta fase produz uma coreografia de serviços concreta. A seleção de serviços é o problema de pesquisa central tratado nesta tese.

- **Implantação**: nesta fase, dado o conjunto de serviços selecionados, é solicitada a implantação da coreografia de serviços por uma máquina de execução típica, como aquela proposta em [74]. A implantação é responsável por adicionar a lógica de coordenação da coreografia dado o conjunto de serviços selecionados.
- **Encenação**: nesta fase a lógica da coreografia é encenada (executada) pelos serviços para atender uma requisição de um usuário. Além disso, esta fase também pode ser responsável pela adaptação da coreografia de serviços em tempo de execução. O processo de adaptação pode ser ativado para ajustar dinamicamente o comportamento de uma coreografia para atender às mudanças nos requisitos do usuário ou na QoS ofertada [94]. Este processo pode exigir uma nova seleção de serviços. O processo de adaptação de coreografias de serviços está fora do escopo da tese.

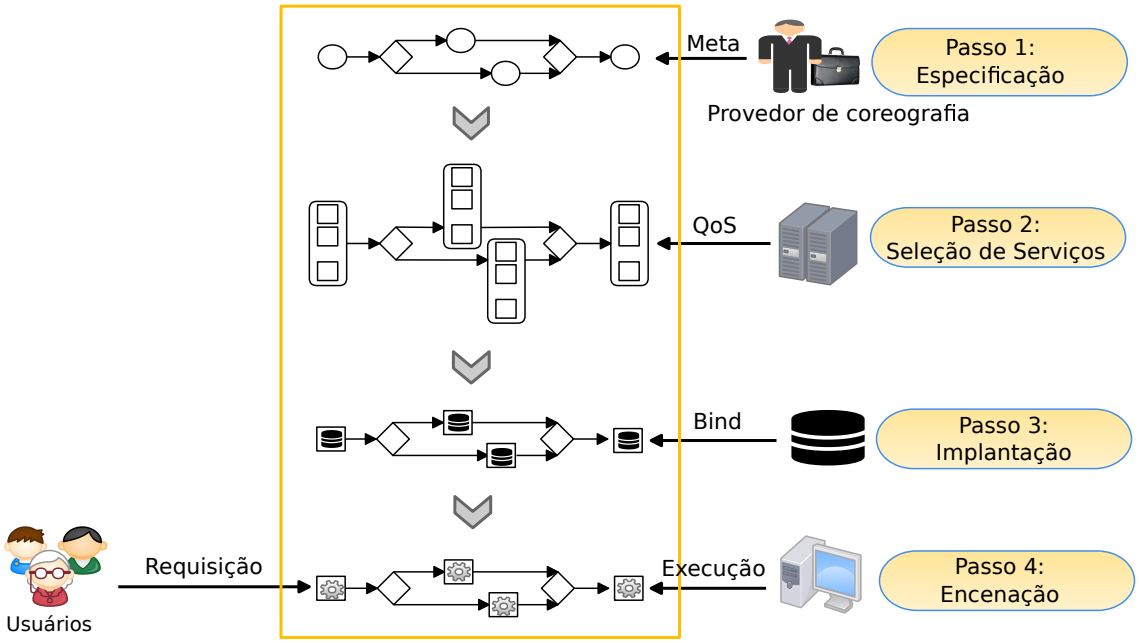


Figura 2.2: *O ciclo de vida de uma coreografia de serviços (adaptado de [114]).*

As atividades realizadas no ciclo de vida de coreografias de serviços são orientadas tanto pelo objetivo de atender aos requisitos funcionais, de forma a garantir a lógica de negócio, quanto por garantir os requisitos de QoS [24]. Como pode existir um conjunto de serviços candidatos com funcionalidades semelhantes, esse processo deve ser capaz de selecionar os serviços mais adequados para desempenhar cada papel de acordo com os requisitos de QoS. Desta forma, a QoS tem um papel central em todo o ciclo de vida de uma coreografia de serviços [91].

2.1.3 Modelagem de coreografias de serviços

Vários modelos e linguagens foram propostos para modelar coreografias de serviços em diferentes níveis conceituais. As linguagens para modelagem de coreografia de serviços podem ser categorizadas utilizando dois critérios [35]: linguagens independentes de implementação e linguagens específicas de implementação. As linguagens independentes de implementação são utilizadas principalmente para descrever processos a partir de uma perspectiva de negócios. Por outro lado, as linguagens específicas de implementação definem os protocolos de comunicação utilizados, o que inclui os formatos das mensagens trocadas.

As linguagens de coreografia de serviços também podem ser classificadas com base no estilo de modelagem para o qual elas oferecem suporte, o qual pode ser baseado em modelos de interação ou em modelos de interconexão [10]. No estilo baseado em modelos de interação, o bloco de construção básico é uma interação, sendo que os fluxos de dados e de controle são baseados nas dependências entre essas interações, consideradas sob uma perspectiva global. Modelos de interação podem representar restrições que precisam ser satisfeitas com o uso de comunicação adicional que não corresponde ao modelo de interação original [151]. Por exemplo, se o serviço A recebe uma mensagem e o modelo de interação exige que o serviço B subsequentemente envie uma mensagem para o serviço C, o serviço B tem que ter conhecimento de quando o serviço A recebe uma mensagem.

Por outro lado, no estilo baseado em modelos de interconexão o fluxo de controle é definido por papel de cada participante, e as respectivas atividades de envio e recepção de mensagens representam as interações. Esse tipo de modelo permite a modelagem de processos para os quais, dado um determinado serviço, não existe parceiro adequado para interação [136].

A Figura 2.3 mostra uma categorização das principais linguagens para modelagem de coreografias segundo os critérios acima descritos [44]. A seguir, essas linguagens são descritas resumidamente.

A linguagem *Web Services Flow Language* (WSFL) [75], proposta pela IBM, define dois tipos de descrições. O primeiro, modelo de fluxo (*flow model*), visa descrever fluxos de trabalho que interagem com outros fluxos, sendo as interações modeladas como serviços *Web*, cuja implementação é especificada usando uma descrição WSDL [30]. O outro tipo, modelo global (*global model*), permite indicar ligações entre as operações de processos que sejam definidos por WSDL. Dessa forma, WSFL pode ser usada apenas para definir a estrutura da coreografia.

BPEL4Chor é uma extensão da *Web Services Business Process Execution Language* (WS-BPEL) para representação de coreografias [36]. Nesta linguagem uma coreografia descreve a troca de mensagens entre serviços a partir da perspectiva de um

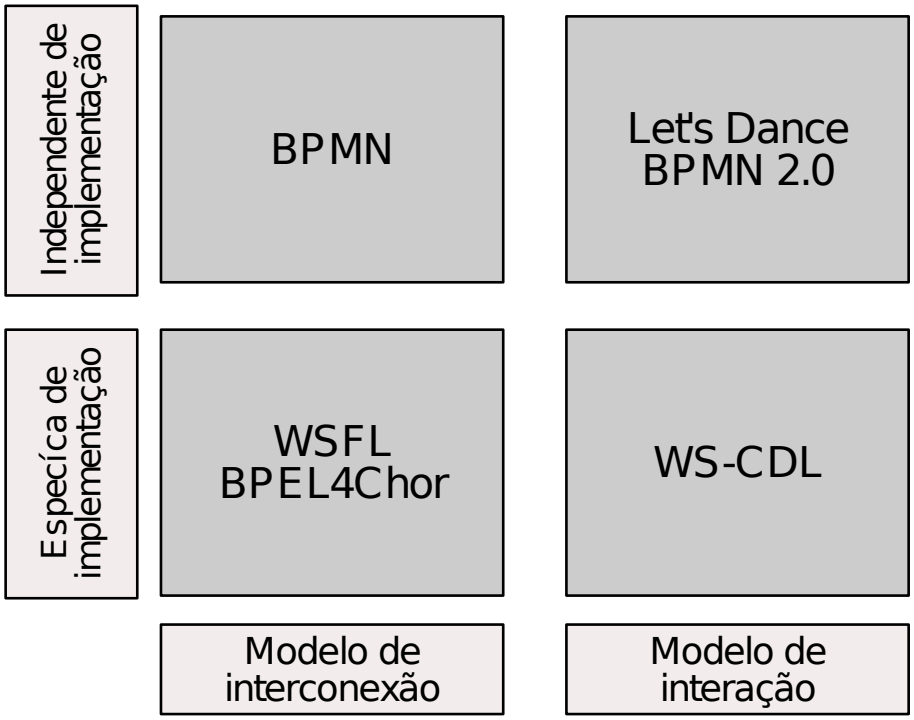


Figura 2.3: *Categorização de linguagens para modelagem de coreografias de serviços (Adaptada de [44]).*

observador que é capaz de observar todas as interações e o fluxo de dependências. Em particular, *BPEL4Chor* introduz uma camada sobre WS-BPEL onde o desenvolvedor define apenas as atividades de envio e recepção de mensagens. Em seguida, dada uma especificação abstrata em *BPEL4Chor*, o desenvolvedor pode implementar os componentes da coreografia na forma de serviços WS-BPEL.

A linguagem *Web Services Choreography Description Language* (WS-CDL) [53] é uma recomendação do W3C baseada em XML que define uma coreografia a partir de um ponto de vista global com relação às várias entidades participantes. Com base nesse ponto de vista global, pode-se extrair as definições dos processos que cada participante deve implementar. Nesta linguagem, o desenvolvedor especifica o protocolo de interações observáveis entre serviços de um ponto de vista global. Desta forma, pode-se dizer que WS-CDL é uma linguagem formal para especificar protocolos. Essa visão global tem a vantagem de permitir a separação entre o processo a ser seguido por uma organização individual, e a definição da sequência segundo a qual essa organização trocará mensagens com as demais. Contudo, WS-CDL é criticada por não separar de maneira apropriada o metamodelo da sintaxe, por oferecer suporte insuficiente a algumas categorias de cenários e por ser de difícil compreensão [11].

A linguagem *Let's Dance* [150] foi criada visando capturar a interação entre serviços sob uma perspectiva comportamental nas fases de análise e projeto de sistemas. Em uma visão global, as interações são descritas sob a perspectiva de uma cole-

ção de serviços abstraídos como papéis. De forma semelhante, a *Business Process Modeling Notation 2* (BPMN2) [99] é uma linguagem gráfica proposta pelo *Object Management Group* (OMG) para modelar processos. Ela oferece uma notação em alto nível que permite focar no papel dos serviços em uma coreografia, sem ter que se preocupar em como esse papel é implementado. Esta linguagem realiza a distinção explícita entre o fluxo de controle e o fluxo de mensagens. Segundo a especificação de BPMN2, uma coreografia é definida como uma sequência ordenada de troca de mensagens B2B entre dois ou mais participantes, sem um controlador central ou entidade responsável ou mesmo um observador do processo. Essa linguagem é, atualmente, o padrão *de facto* para modelagem gráfica de processos de negócio [35]. Na Seção 2.1.4 descrevemos os principais elementos da BPMN2.

2.1.4 Modelagem de coreografias de serviços com BPMN2: Exemplo

A partir da versão BPMN2, lançada em 2010 [99], a BPMN introduziu a especificação de coreografias de serviços. Em um diagrama de coreografia o foco está na coordenação da troca de mensagens entre os serviços participantes, estabelecendo um contrato processual entre eles. A Figura 2.4 apresenta um subconjunto dos principais elementos em BPMN2 utilizados na construção de diagramas de coreografias de serviços.

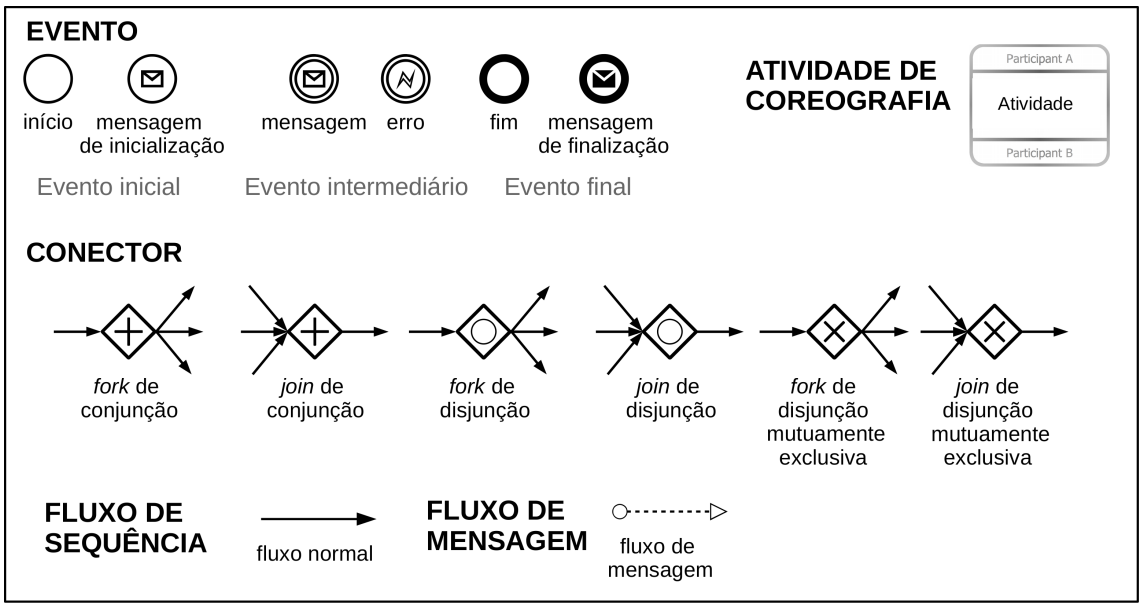


Figura 2.4: *Subconjunto dos principais elementos BPMN2 para modelagem de coreografias de serviços.*

Eventos podem representar o início de um processo (evento de início), o fim de um processo (evento de fim) ou algo que deve acontecer durante o processo (evento intermediário). Na Figura 2.4, apresentamos eventos sem especial conotação (usados

quando o modelador não especifica o tipo), eventos de mensagem (usados quando mensagens são usadas como gatilho do processo, para indicar envio ou recebimento de mensagens durante o processo, ou para indicar o fim do processo) e eventos de erro (podendo indicar que o fluxo dispara ou captura um erro).

O principal elemento de um diagrama de coreografia refere-se às atividades de coreografia, também chamadas de tarefas de coreografia, que consiste na troca de mensagens entre participantes, frequentemente representados por papéis. Uma atividade de coreografia é representada como um componente que apresenta três divisões. A divisão superior representa o participante que inicia a interação, a divisão intermediária é usada para representar o nome da tarefa que deve ser realizada pela atividade e a divisão inferior representa o participante que é o dono da tarefa. Quando dois serviços desempenham o mesmo papel é dito que ambos fornecem funcionalidades equivalentes.

As atividades de uma coreografia são conectadas por um fluxo de sequência, que é utilizado para demonstrar o fluxo lógico que representa toda a troca de informações que acontece naquele processo de negócio. As atividades só podem se conectar a outros objetos do fluxo como eventos, conectores (*gateway*) e outras atividades.

Por sua vez, conectores controlam fluxos de sequência. Um conector *fork* de conjunção é utilizado para criar fluxos paralelos, ao passo que um conector *join* de conjunção é usado para sincronizar fluxos paralelos. Um conector *fork* de disjunção divide o fluxo de sequência e ativa uma ou mais alternativas de acordo com uma condição. O conector *join* de disjunção aguarda que todos os ramos de entrada ativos sejam concluídos antes de acionar o fluxo de saída. O conector *fork* de disjunção mutuamente exclusiva divide o fluxo de sequência usando como referência o valor de dados ou a ocorrência de eventos, encaminhando o fluxo para exatamente um dos ramos de saída. Por outro lado, o conector equivalente *join* aguarda a conclusão de um de seus ramos de entrada antes de acionar o fluxo de saída.

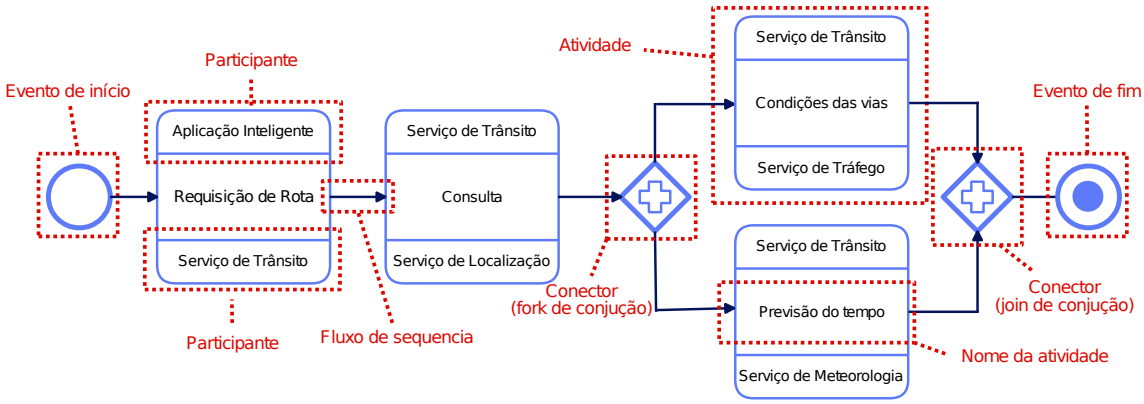


Figura 2.5: *Exemplo de uma coreografia de serviços em notação BPMN2 para busca da melhor rota entre dois pontos em uma cidade.*

A Figura 2.5 apresenta um exemplo de coreografia de serviços para busca pela melhor rota entre uma origem e destino, seguindo o cenário descrito no Capítulo 1. Essa coreografia é modelada usando a notação BPMN2 e ilustra os principais elementos de diagrama de coreografia de serviços. Neste exemplo, em uma atividade de *consulta de trajeto urbano*, pode-se representar a troca de mensagens entre dois participantes da seguinte forma: um participante representa o *Serviço de Trânsito*, que pode ser substituído pelo papel *trânsito*, e o outro participante representa o *Serviço de Localização*, que pode ser substituído pelo papel *mapa*. Além disso, o conector *join* de conjunção é usado para sincronizar fluxos paralelos, ou seja, buscar a *previsão do tempo* e *condições da via*, como ilustrado na Figura 2.5.

Ainda de acordo com o cenário descrito no Capítulo 1, a Figura 2.6 exemplifica a especificação de coreografias de serviços utilizando a notação BPMN2 para dois exemplos. A Figura 2.6(a) representa uma coreografia para reserva de hotel descrevendo o conjunto de papéis a serem utilizados para reservar um quarto de hotel e transporte em uma determinada data para um dado destino. A Figura 2.6(b) representa uma coreografia de serviços para consulta por restaurantes, incluindo os papéis utilizados para consulta por restaurantes de acordo com o perfil do usuário. Nesse exemplo, são acessados serviços do tipo de localização, restaurante, rota, transporte, pedido de comida e pagamento. Vale ressaltar que os papéis em destaque (tracejados em vermelho) são papéis comuns em mais de uma coreografia.

O padrão BPMN2 permite estabelecer apenas o contrato procedural entre os serviços participantes, tratando apenas dos aspectos funcionais de tais contratos. Não existem, portanto, mecanismos para estabelecer acordos contratuais em relação aos níveis de QoS exigidos pelo provedor de coreografias [13]. Por esse motivo, essa linguagem não é suficiente para representar os requisitos de QoS associados a uma coreografia como um todo ou a seus papéis individualmente. Nesta tese, propomos uma representação para contornar essa limitação, que será apresentada no Capítulo 5.

2.2 Requisitos não-funcionais sobre serviços

Um produto de software é desenvolvido para atender a um conjunto de funcionalidades e requisitos relacionados com as restrições ou qualidades com as quais essas funcionalidades devem ser realizadas, como desempenho, confiabilidade, portabilidade e custo. Esses requisitos, chamados de requisitos não-funcionais ou propriedades não-funcionais, especificam restrições globais que devem ser atendidas pelo software [107].

Os requisitos não-funcionais são chamados, geralmente, de atributos de qualidade na literatura de engenharia de software [116]. Na computação orientada a ser-

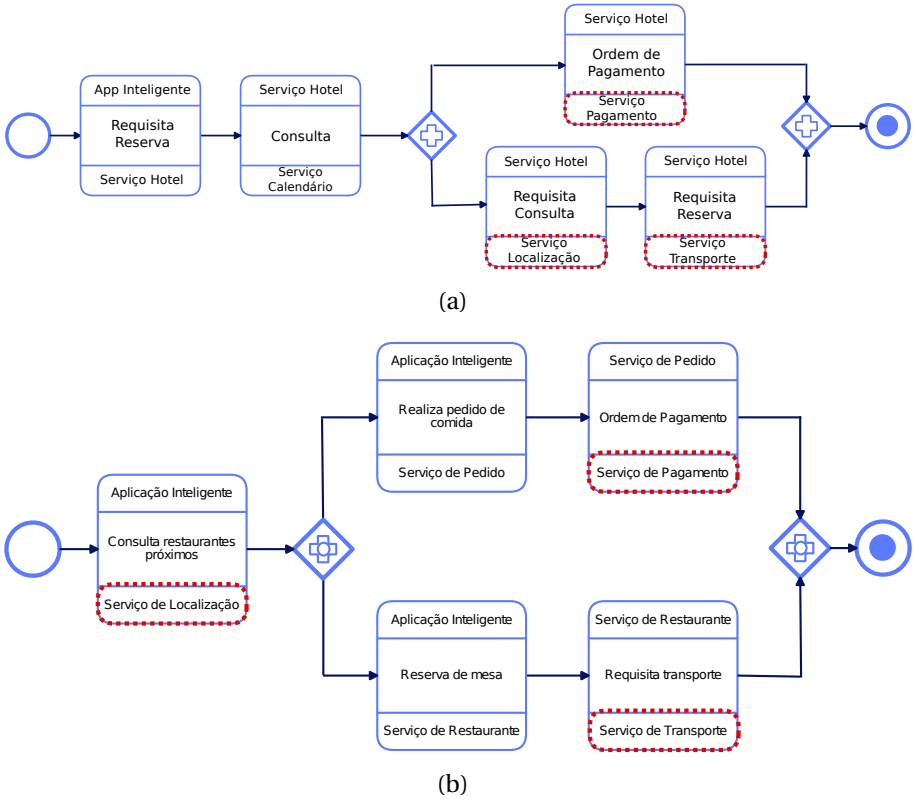


Figura 2.6: *Exemplos de coreografia de serviços em notação BPMN2 com base no cenário discutido nesta tese: (a) Coreografia para reserva de hotel usando a notação BPMN2. (b) Coreografia de serviços para consultar por restaurantes usando a notação BPMN2.*

viços, requisitos não-funcionais são referenciados pelo termo qualidade de serviço, ou apenas QoS [89, 111, 65].

A *International Telecommunication Union (ITU)* define QoS como um conjunto de características de um serviço que afetam sua capacidade de atender às necessidades declaradas pelo usuário do serviço [60]. Kritikos e Plexousakise definem QoS como um conjunto de atributos que têm impacto na qualidade oferecida por um serviço, onde qualidade é sinônimo de atendimento das especificações fornecidas pelos provedores de coreografias [71]. Desta forma, ao se definir a QoS para um serviço busca-se estabelecer um limiar para os valores dos atributos de QoS que devem ser garantidos de forma a atender as expectativas dos usuários.

Cada atributo de QoS possui uma métrica para sua quantificação, a qual é usada para avaliar o desempenho de um serviço [65]. Uma métrica de QoS descreve em detalhes como a medição é conduzida, ou seja, define as unidades de medidas utilizadas, os tipos de valores, o instante ou periodicidade de medição e o responsável pela medição [71]. Por exemplo, a métrica de QoS *tempo de resposta*, que pode ser associada a um serviço de localização, deve ser medida pelo provedor de coreografias a

cada requisição que utiliza o serviço, sendo que o valor, dado em milissegundos, deve ser menor que *10 milissegundos*.

Esta tese classifica QoS em duas categorias:

- **QoS requisitada:** é o nível mínimo de QoS especificada como requisito na definição de um papel individual, chamado de requisito de QoS local, ou na definição da coreografia como um todo, chamado de requisito de QoS global. A QoS requisitada estabelece o nível mínimo de QoS que o provedor de coreografias espera obter ao requisitar um serviço junto aos provedores de serviços utilizados em uma coreografia. Por exemplo, o provedor de coreografias, ao especificar uma composição de serviços, pode definir que a reputação exigida pelos serviços seja maior que 95%.
- **QoS oferecida:** é o valor mínimo de uma métrica de QoS garantida pelo provedor de um serviço a todos os provedores de coreografia, firmado como resultado de uma negociação prévia entre o provedor de serviços e os provedores de coreografias, usando a QoS requisitada como parâmetro. Por exemplo, um provedor de serviços informa que o tempo de resposta máximo do *serviço de localização* é de *8 milissegundos*.

A QoS requisitada estabelece restrições sobre os atributos de QoS dos serviços contratados, especificando um contrato entre as partes envolvidas. Isto é importante, uma vez que os serviços são oferecidos por terceiros e hospedados em ambientes que podem estar fora do controle dos usuários ou mesmo do provedor de coreografias [33].

Esta tese assume que informações sobre serviços são gerenciadas por um ou mais catálogos de serviços, comumente conhecidos como *brokers* de serviços [79]. Neste caso, um catálogo de serviços contém a descrição de um conjunto de serviços disponibilizados com informações sobre a QoS oferecida pelo provedor de serviços.

Os provedores de serviços e os provedores de coreografias podem negociar os termos e condições relacionados à QoS que um serviço deve entregar. O resultado dessa negociação pode ser um Acordo de Nível de Serviço (*Service Level Agreement* – SLA) que também define responsabilidades e condições em caso de não atendimento do acordo [89, 91, 109].

Para assegurar a QoS requisitada pelos provedores de coreografia, esta tese utiliza uma abordagem que utiliza um sistema de *broker*. Nesse cenário, um sistema de *broker* atua em nome do provedor de coreografias para selecionar serviços mais adequados e negociar com os provedores de serviços para alcançar melhores garantias dos contratos de serviços [19]. Nesse caso, o uso de abordagens de seleção de serviços busca minimizar violações de QoS ao utilizarem um processo mais racional, permitindo selecionar serviços que garantam os requisitos de QoS.

A garantia da QoS requisitada é um dos principais desafios em abordagens de coreografias de serviços. Este aspecto é ainda mais crítico ao implantar múltiplas coreografias de serviços em ambientes dinâmicos (por exemplo, ambientes de nuvem), devido à existência de múltiplos serviços que cooperam entre si, e ao possível compartilhamento destes serviços entre as coreografias. Nesses casos, a preocupação de garantir os requisitos de QoS dos serviços participantes vem do fato de que a QoS global de uma coreografia depende diretamente da QoS local de cada serviço individual [38].

2.3 Estratégias de seleção de serviços sensível à QoS

A garantia dos requisitos de QoS é um dos principais desafios no contexto de coreografias de serviços, uma vez que a QoS oferecida tem um papel crucial na seleção de serviços [91]. Isso ocorre porque diferentes atributos de QoS, como disponibilidade, tempo de resposta, capacidade e segurança, podem ser usados na seleção de serviços como uma forma de diferenciação na preferência dos provedores de coreografias em relação aos serviços candidatos para um mesmo papel [103]. Em geral, nenhum serviço candidato pode exceder todos os outros serviços em relação a todos os atributos de QoS. Por exemplo, um serviço pode fornecer o melhor tempo de resposta, mas pode ter alto preço e baixa disponibilidade. Uma forma de lidar com vários atributos de QoS é o emprego de funções de utilidade [5].

A finalidade de uma função de utilidade, no contexto de seleção de serviços, é tratar a multidimensionalidade da QoS dos serviços, mapeando os valores de múltiplos atributos de QoS em um único valor chamado de *QoS agregada* de um serviço [4]. Desta maneira, uma função de utilidade permite classificar serviços que tenham as mesmas funcionalidades, de acordo com as diversas dimensões de QoS [128]. O uso de uma função de utilidade permite ainda conferir pesos aos atributos para quantificar a prioridade de um requisito em relação a outro requisito. Para isso, uma função de utilidade deve acrescentar pesos aos atributos de QoS [153].

Uma estratégia de avaliação de QoS, dado um conjunto de atributos de QoS, provê funções de utilidade que classifica os serviços de acordo com a melhor QoS [122], chamada de ***best-QoS***, ou de acordo com a proximidade da QoS ofertada em relação à QoS requisitada [66], chamada de ***best-fit***.

A estratégia *best-QoS* seleciona os serviços candidatos com os melhores valores de QoS [2]. Nesse caso, as funções de utilidade geralmente usam a abordagem Aditiva Ponderada Simples (*Simple Additive Weighting* – SAW) sobre os atributos de QoS [5, 153]. A aplicação dessa abordagem no contexto de seleção de serviços sensível à QoS requer que os intervalos de todos os atributos de QoS sejam normalizados e que os valores normalizados sejam convertidos em um único valor de QoS agregada

a ser maximizado [118]. Em outras palavras, no processo de normalização, cada valor de atributo de QoS de um serviço candidato é transformado em um valor entre 0 e 1, comparando-o com o valor mínimo e máximo possível de acordo com as informações de QoS disponíveis dos serviços candidatos. Em seguida, os valores normalizados de cada atributo de QoS são somados, gerando a QoS agregada do serviço candidato.

Na estratégia *best-fit*, um serviço candidato é selecionado quando sua QoS oferecida é mais próxima da QoS requisitada, calculando a utilidade como a medição de distância Euclidiana tradicional entre a QoS oferecida pelos serviços candidatos e a QoS requisitada pelo provedor de coreografias [66, 64].

Em um processo de seleção de serviços sensível à QoS, um determinado serviço pode ainda ser selecionado por mais de uma coreografia, o que, significa que uma mesma operação de um serviço pode receber requisições simultaneamente de diferentes aplicações coreográficas, inclusive com diferentes requisitos de QoS. Logo, a seleção de serviços deve garantir a QoS requisitada considerando um ***fluxo de requisições*** para um mesmo serviço [23]. Isso ocorre com maior frequência quando coreografias competem pelo mesmo grupo de serviços candidatos.

Se a seleção de serviços é realizada por coreografia isoladamente, logo um serviço selecionado pode receber um fluxo de requisições que podem exceder a capacidade máxima de processamento do serviço, levando a uma sobrecarga e fazendo com que o serviço não possa mais garantir a QoS ofertada. Por este motivo, a seleção de serviços sensível à QoS também deve ser ***sensível à capacidade***, levando em consideração a carga agregada [1] sobre o serviço juntamente com sua capacidade.

Para resolver o problema de seleção de serviços sensível à QoS, a maioria dos trabalhos o tratam como um problema de otimização, também chamado de problema de seleção de serviços sensível à QoS para composições de serviços [153]. Esses trabalhos tentam encontrar o conjunto ideal de serviços para uma coreografia considerando a definição dos requisitos de QoS.

Existem duas abordagens propostas com base na definição de requisitos de QoS local ou requisitos de QoS global, como discutido na Seção 2.2: abordagem de seleção local e uma abordagem de seleção global. A abordagem de seleção local seleciona um serviço para cada papel individual de uma coreografia sem considerar requisitos de QoS que abrangem múltiplos papéis e sem necessariamente lidar com requisitos de QoS global [153].

As abordagens de seleção global resolvem o problema de seleção de serviços considerando a composição de serviços como um todo [4]. A seleção global tenta oti-

[1] Carga agregada refere-se ao somatório das cargas individuais de diferentes usuários para um mesmo serviço.

mizar todo o conjunto de serviços selecionados para todos os papéis, levando em consideração a estrutura da composição de serviços que constitui a coreografia. Os valores de QoS agregados de todas as combinações de serviços possíveis são computados e a combinação de serviços que maximiza o valor da QoS agregada, ao mesmo tempo que satisfaz os requisitos de QoS global, é selecionada [153].

A maioria das soluções propostas na literatura para tratar a seleção de serviços para coreografias se concentra nos requisitos de QoS global [153, 8, 6, 105]. Contudo, essa abordagem é mais complexa do que a seleção local, sendo definida como um problema *NP-Difícil* e adotando soluções que seguem a mesma linha dos problemas de otimização [94].

Algoritmos de programação linear inteira têm sido utilizados com frequência para resolver a seleção de serviços com requisitos de QoS globais [69]. A programação linear inteira é uma técnica para otimização de uma função objetivo linear que visa determinar um vetor de variáveis que maximiza ou minimiza uma função, dado um conjunto de restrições [152]. Ao utilizar uma solução baseada em programação linear inteira pode-se garantir uma solução ótima ao comparar todas as soluções possíveis.

No entanto, essa abordagem é limitada pelo tempo computacional, que aumenta exponencialmente quando o número de serviços candidatos e o número de atributos de QoS aumentam [5]. Para superar essas limitações, principalmente em ambientes de computação altamente dinâmicos, algumas propostas [20, 61, 120] exploram o uso de heurísticas para lidar com a seleção de serviços com requisitos de QoS globais.

Na abordagem heurística, as coreografias de serviços também devem garantir os requisitos de QoS global, ao mesmo tempo que buscam maximizar a QoS entregue ao usuário. Em geral, esta abordagem não examina todas as composições de serviços possíveis, como as soluções que utilizam programação linear inteira. Ao contrário, elas tentam aplicar diferentes heurísticas para explorar um subconjunto de composições de serviços que provavelmente podem levar a uma solução satisfatória.

Por exemplo, no trabalho de Yu e Lin, o problema de composição de serviços sensível à QoS é modelado como um problema de mochila com múltipla escolha (*multi-choice knapsack problem* – MCKP) e também como um problema baseado em grafos [148]. Ambas as abordagens são resolvidas usando heurísticas de forma a reduzir a complexidade do tempo [149].

Outra abordagem heurística é a aplicação de algoritmos genéticos. Um algoritmo genético lida recursivamente com um conjunto de serviços e realiza comparações para encontrar a melhor solução, usando uma função de *fitness* projetada para maximizar (minimizar) alguns atributos de QoS, com base no ponto de vista do provedor de coreografias. Comumente, esse processo termina quando um número máximo

de iterações foi executado ou um nível satisfatório da função de *fitness* foi atingido [94]. Entretanto, algoritmos genéticos não garantem que seja encontrado um conjunto de serviços que atenda aos requisitos de QoS globais, uma vez que esta abordagem não considera todas as combinações de serviços possíveis [69].

De forma semelhante, em [63] os autores transformam o problema de seleção de serviços em um problema de atribuição de peso máximo e empregam o método húngaro para realizar uma seleção global, de forma a atender todas as requisições. Já outras abordagens heurísticas utilizam seleção global e seleção local em conjunto para reduzir a complexidade de tempo e permitir a implementação descentralizada da seleção de serviços [76].

2.4 Considerações finais

Este capítulo apresenta alguns conceitos importantes para o desenvolvimento desta tese, como coreografia de serviços e requisitos não-funcionais sobre serviços. Em seguida, para evidenciar os principais conceitos existentes na seleção de serviços sensível à QoS para múltiplas coreografias, apresentamos uma discussão sobre os principais desafios nas diferentes estratégias de seleção de serviços. Sendo assim, os tópicos relacionados ao espaço do problema foram discutidos, bem como aqueles relacionados à solução.

Nesta tese, propomos uma abordagem para seleção de serviços sensível à QoS e à capacidade para múltiplas coreografias. A abordagem proposta tem como objetivo realizar a seleção de serviços que assegure a satisfação dos requisitos de QoS dos provedores de coreografias e que minimize os custos relacionados aos recursos onde os serviços que compõem as coreografias são executados.

Argumentamos que a seleção de serviços analisando múltiplas coreografias de serviços ao mesmo tempo expressa uma perspectiva mais realista em cenários reais. Apesar disso, as soluções para seleção de serviços encontradas na literatura ainda são bastante incipientes para lidar com múltiplas coreografias que possuem papéis em comum. No próximo capítulo discutimos algumas dessas abordagens, destacando suas vantagens e desvantagens.

CAPÍTULO 3

Trabalhos relacionados

De acordo com o paradigma SOC *Service-Oriented Computing*, aplicações construídas como coreografias de serviços são especificadas como processos abstratos, os quais, por sua vez são compostos de um conjunto de papéis. Em seguida, um serviço concreto é selecionado para cada papel, assegurando flexibilidade na construção de coreografias de serviços [21]. Contudo, devido à extensa quantidade de serviços candidatos disponíveis com diferentes ofertas de QoS, a seleção dos serviços mais adequados para cada papel passa a ser um grande desafio. Este problema torna-se ainda mais complexo ao lidar com a seleção de serviços sensível à QoS para implantar múltiplas coreografias, com seus respectivos requisitos de QoS, que competem pelo mesmo conjunto de serviços candidatos.

Este capítulo realiza uma revisão da literatura para examinar abordagens que lidam com seleção de serviços sensível à QoS para composições de serviços em geral, em especial abordagens que lidam com orquestração e coregrafia de serviços. Vale ressaltar que trabalhos que não lidam diretamente com composições de serviços, mas empregam técnicas de seleção de serviços sensível à QoS e são relacionados com nossa abordagem também são analisados. A análise das abordagens discutidas foi guiada pelas seguintes de questões de pesquisa organizadas em três aspectos:

- ***Seleção de serviços sensível à QoS***: Como os requisitos de QoS são considerados na seleção de serviços? Qual estratégia de avaliação de QoS é utilizada para avaliar a qualidade dos serviços candidatos em relação aos requisitos de QoS? Como a seleção de serviços sensível à QoS realiza a classificação dos serviços candidatos?
- ***Compartilhamento de serviços entre coreografias***: A capacidade dos serviços candidatos é tratada durante o processo de seleção? A abordagem de seleção considera o compartilhamento de serviços de maneira a evitar a degradação da QoS? A abordagem seleção de serviços maximiza o número de coreografias implantadas?
- ***Seleção de serviços sensível ao custo***: Quais abordagens e estratégias de seleção de serviços visam minimizar o custo dos serviços selecionados? Quais métodos

são utilizados na seleção de serviços para garantir o uso eficiente de recursos pelos serviços selecionados?

As Seções 3.1, 3.2 e 3.3 apresentam os trabalhos de seleção de serviços, de acordo com os três aspectos mencionados. Vale ressaltar que alguns trabalhos lidam com mais de um dos aspectos discutido acima, logo tais trabalhos podem aparecer em mais de uma seção.

3.1 Seleção de serviços sensível à QoS

Esta seção descreve os trabalhos que tratam da seleção eficiente de serviços para composições objetivando garantir os requisitos de QoS. Esses trabalhos são cobertos devido à sua relevância no estado da arte de seleção de serviços sensível à QoS e seu relacionamento próximo com a abordagem desta tese.

Os algoritmos de seleção de serviços sensível à QoS podem ser classificados em algoritmos de seleção local, de seleção global ou híbridos [94]. Essa classificação é baseada no escopo do requisito QoS, ou seja, como os requisitos de QoS são tratados durante o processo de seleção.

Nos trabalhos propostos por Menascé [89] e Zeng *et al.* [153] a abordagem de seleção local é usada para selecionar serviços para composições com requisitos de QoS definidos para cada papel específico. Já Al-Masri e Mahmoud [2] e Wang *et al.* [131] aplicam a abordagem de seleção local para lidar com serviços individuais.

Esta abordagem é especialmente útil para ambientes distribuídos onde grupos de serviços candidatos são gerenciados e selecionados de forma independente para uma coreografia [5]. Além disso, a seleção local pode ser resolvida de forma eficiente, sendo que a sua complexidade de tempo no pior caso é $O(n)$, onde n é o número de serviços candidatos [144].

Entretanto, a seleção local é incapaz de lidar com requisitos de QoS global. Outra desvantagem desta abordagem é que o provedor de coreografias precisa especificar os requisitos de QoS local para cada papel presente em uma coreografia [7]. Por outro lado, devido à eficiência em termos computacionais, a seleção local tem siso usada em combinação com outras abordagens, tais como [4], [81] e [85].

Atualmente, a maioria das abordagens de seleção de serviços propostas na literatura tratam de requisitos de QoS global [8, 61, 29, 84, 123]. As coreografias de serviços com requisitos de QoS global são mais pragmáticas em ambientes reais, onde os requisitos são definidos para a coreografia como um todo, desobrigando o usuário em conhecer a estrutura de uma coreografia. Nesta situação, a seleção de serviços é um problema multicritério pertencente à classe de problemas NP-completo. Quando

o número de papéis definidos em uma coreografia e o número de serviços candidatos crescem, a seleção demora mais tempo para convergir em uma solução ótima.

Zeng *et al.* [153] apresentam uma abordagem para seleção de serviços sensível à QoS global utilizando técnicas de programação linear. Os serviços são avaliados por meio de um modelo de QoS multidimensional, buscando selecionar os serviços candidatos com a melhor QoS possível, ou seja, usando a estratégia *best-QoS*. Além do mais, uma composição pode ser adaptada frente às mudanças que ocorrem durante a execução. Por exemplo, algum serviço se torna indisponível ou os valores da QoS diminui abaixo dos valores requisitados. Para este fim, a abordagem revisa o plano de execução para otimizar a QoS dado o conjunto de requisitos do provedor de coreografias e o conjunto de serviços candidatos.

Semelhante a essa abordagem, Ardagna e Pernici [8] estendem o modelo de programação linear para tornar a execução de processos de negócio baseados em composições de serviços mais flexível e adaptativa. O objetivo é descobrir o mapeamento ótimo entre cada papel de uma composição e um serviço candidato que seja capaz de satisfazer os requisitos do papel, de modo que a QoS global percebida pelo provedor de coreografias seja maximizada sob requisitos de QoS severos. Nessa abordagem proposta, a seleção de serviços pode ser executada tanto para requisitos de QoS local quanto global. Em complemento, foram implementadas técnicas para otimização de iterações de *loop* e técnicas de negociação para identificar uma nova solução viável do problema, quando se lida com restrições de QoS severas, buscando reduzir a sobrecarga do processo de adaptação dinâmica.

Os trabalhos Zeng *et al.* [153] e Ardagna e Pernici [8] utilizam com base de resolução métodos de programação linear que são ineficientes quando o tamanho do problema cresce, uma vez que possuem complexidade de tempo [1] exponencial [86]. Para lidar com esta questão em problemas de seleção de serviços utilizando métodos de programação linear, Alrifai, Skoutas, e Risse [6] propõem o uso da técnica *skyline* como um passo de otimização adicional na fase de pré-processamento. Para isso, eles consideram relações de dominância entre os serviços candidatos com base em seus atributos de QoS, onde apenas os serviços que não são dominados por nenhum outro serviço funcionalmente equivalente, são candidatos válidos para uma coreografia. Por exemplo, o serviço A domina o serviço B se os atributos de QoS do serviço A é igual ou melhor do que todos os atributos de QoS do serviço B e melhor em pelo menos um atributo de QoS. A técnica *skyline* tem como objetivo remover um subconjunto de serviços candidatos do espaço de busca e, como consequência, reduzir o tempo de

[1] A complexidade de tempo de um algoritmo quantifica a porção de tempo tomada por um algoritmo para executar em função do tamanho da entrada do problema.

seleção exigido. No entanto, embora esse passo forneça uma poda inicial do número de serviços candidatos, o tamanho do conjunto ainda pode ser grande, dependendo da distribuição dos valores de QoS.

Essas abordagens sofrem de baixa escalabilidade devido à complexidade de tempo exponencial com o incremento do número de serviços candidatos e papéis presentes em uma coreografia. Para alcançar maior escalabilidade algumas abordagens baseadas em heurísticas foram propostas [94, 27]. Por exemplo, Canfora *et al.* [20] discutem uma abordagem para tratar o problema de seleção de serviços sensível à QoS usando uma solução baseada em algoritmos genéticos. Os resultados revelam que os algoritmos genéticos mostram um melhor desempenho e escalabilidade do que a programação linear com o incremento do número de serviços candidatos e de papéis. Além disso, o uso de algoritmos genéticos possibilita lidar com problemas de seleção de serviços que apresentem fórmulas de agregação de QoS arbitrárias e não lineares, enquanto que as abordagens que utilizam a programação linear requerem linearização [145].

Yu *at al.* [148] discutem o problema de composições de serviços com requisitos de QoS global e propõem heurísticas a serem utilizados na busca por soluções de forma mais eficiente do que soluções exatas. Os autores propõem duas abordagens de solução: modelagem do problema como o problema da mochila multidimensional de múltipla escolha (*Multiple choice Multi-dimension Knapsack Problem* – MMKP) e a modelagem do problema como problema de caminho mais curto com múltiplas restrições (*Multi-Constraint Optimal Path Problem* – MCOP). Para ambas modelagens, heurísticas são discutidas.

Peng e Changsong [105] apresentam um algoritmo que descreve o problema de seleção de serviços sensível à QoS em cenários de larga escala como um problema da mochila multidimensional, aplicando uma estratégia de computação evolutiva para obter uma solução aproximada. Na abordagem proposta, as coreografias são modeladas como grafos dirigidos e solicitam serviços fornecendo os requisitos funcionais e requisitos de QoS. Ao aplicar o algoritmo de seleção, os melhores serviços para cada papel de uma coreografia são selecionados de acordo com a QoS oferecida através do uso da estratégia *best-QoS*. Para fornecer QoS adaptativa, a solução usa um banco de dados com o perfil das coreografias, contendo informações sobre desempenho dos serviços utilizados, onde diferentes serviços podem ser requisitados quando ocorrem mudanças no contexto das coreografias.

Mabrouk *et al.* [83] apresentam um algoritmo de seleção de serviços guiado por heurísticas para ambientes dinâmicos. Mais especificamente, o algoritmo heurístico trata do problema de seleção de serviços em duas fases. A primeira fase realiza uma classificação local, que visa ranquear os serviços candidatos para cada papel de

acordo com diferentes níveis de QoS. Para conseguir isso, utiliza-se um algoritmo heurístico baseado em técnicas de agrupamento (algoritmo *K-Means*), onde os níveis de QoS resultantes são utilizados para determinar a utilidade dos serviços candidatos em relação a QoS ofertada através do emprego da estratégia *best-QoS*. Por fim, uma fase de seleção global usa as utilidades obtidas das ofertas de QoS para orientar a seleção de várias composições, onde se seleciona um conjunto de serviços candidatos com a melhor QoS para cada papel que, quando compostos juntos, atendem aos requisitos de QoS global. Ao obter várias coreografias busca-se tornar a adaptação mais eficiente, uma vez que se pode trocar coreografias que utilizam serviços que não estejam mais disponíveis.

Além dessas estratégias, algoritmos híbridos, combinando métodos de seleção local e global, também foram propostos para lidar com as adversidades de desempenho para o problema de seleção de serviços com requisitos de QoS global [81, 85]. Alrifai, Risse e Nejdl [5] propuseram um método de composição de serviços dinâmico baseado na decomposição da QoS global em QoS local. Esse método consiste em duas etapas: em primeiro lugar, os requisitos de QoS global são decompostos em requisitos de QoS local; em segundo lugar, a seleção local é usada para selecionar os melhores serviços para cada papel usando a estratégia *best-QoS*. Segundo os experimentos, a vantagem deste método é que o segundo passo pode ser realizado de forma distribuída usando a abordagem de seleção local, de forma que o tempo para realizar a seleção pode diminuir [5]. No entanto, esse tipo de abordagem não pode garantir uma solução ótima.

Em Liu *et al.* [80], uma abordagem de seleção de serviços híbrida também é proposta. Em um primeiro momento da abordagem, k composições de serviços, baseados no histórico dos valores de QoS dos serviços, são selecionados utilizando um algoritmo genético proposto. Em seguida, para reduzir o número de serviços disponíveis, os serviços nas k melhores composições são filtrados e empregados como os serviços candidatos. Então, os requisitos de QoS global são decompostos em requisitos de QoS local utilizando outro algoritmo genético proposto. Na segunda etapa, os serviços com melhor QoS são selecionados usando a estratégia *best-QoS*.

Palade *et al.* [101] propõem uma abordagem inspirada no algoritmo de colônia de formigas para modelar a seleção de microsserviços. Os autores aplicam essa abordagem para serviços implantados em dispositivos móveis para formar uma comunidade de serviços dentro de uma área geográfica limitada. Cada microsserviço é modelado usando um agente de serviço e a composição é realizada por um conjunto de agentes de serviço que formam uma organização para colaborar. Essa abordagem adota uma estratégia de avaliação *best-QoS* e a adaptação dinâmica é executada quando um novo serviço com QoS potencialmente melhor surge.

Os trabalhos discutidos nesta seção tratam da seleção de serviços para composições de serviços individual, ou seja, independente de outras composições. Essa estratégia é apropriada quando os provedores de coreografias hospedam os próprios serviços ou utilizam serviços dedicados. Sendo assim, essas abordagens não são sensíveis a capacidade dos serviços selecionados, tratando os mesmos com capacidade infinita. Nestas abordagens, caso a carga supere a capacidade de um serviço, ele pode ficar sobrecarregado, não mais garantindo a QoS ofertada. Na próxima seção serão apresentados os trabalhos que realizam a seleção de serviços levando em consideração a capacidade dos serviços candidatos, geralmente, devido ao compartilhamento de serviços entre múltiplas requisições.

3.2 Compartilhamento de serviços

A maioria dos trabalhos em seleção de serviços, geralmente, considera somente requisições isoladas ao realizar a seleção, assumindo que não existe compartilhamento de serviços entre coreografias [153, 5, 80, 123]. Em contraste, a encenação de múltiplas coreografias com serviços compartilhados é usualmente necessária em cenários dinâmicos, como Cidades Inteligentes.

Nesse cenário, as abordagens de seleção de serviços devem ser sensíveis à capacidade máxima de cada serviço, de forma a garantir que o número de requisições não exceda sua capacidade, causando uma possível violação à QoS prometida. Dois aspectos devem ser considerados quando há compartilhamento de serviços: 1) o fluxo de requisições dirigidos aos serviços; e 2) número de coreografias de serviços. Nas Subseções 3.2.1 e 3.2.2 os trabalhos que lidam com esses dois aspectos são discutidos.

3.2.1 Seleção de serviços sensível ao fluxo de requisições

Muitas abordagens de seleção de serviços consideram como referencial de otimização uma única requisição de um usuário enviada para uma coreografia de serviços, visando cumprir os requisitos de QoS desta requisição em particular [8, 149]. Essa consideração pode levar à sobrecarga e à degradação do desempenho de um serviço quando ocorre um aumento da carga devido a requisições de outros usuários. O aumento do fluxo de requisições pode ocorrer devido ao compartilhamento de serviços entre coreografias ou em razão de múltiplas requisições para uma mesma coreografia por diferentes usuários. Nesses casos, as abordagens de seleção de serviços devem ser sensíveis à capacidade dos serviços, ou seja, tais abordagens devem analisar a carga sobre os serviços, garantindo que o número de requisições para um serviço não exceda sua capacidade.

Wang *et al.* [133] apresentam uma abordagem de seleção de serviços sensível à capacidade em ambiente de nuvem para múltiplos usuários. Nesta abordagem, o modelo de utilidade da QoS, que descreve a relação entre a QoS requisitada e a QoS dos serviços candidatos, é construído juntamente com o modelo de capacidade dos serviços utilizando programação linear. Ao aplicar a estratégia *best-QoS*, a abordagem busca garantir que cada provedor de coreografias possa selecionar serviços com a melhor QoS. Entretanto, para manter o balanceamento da carga dos serviços, o foco não é apenas fornecer os serviços com a melhor QoS aos provedores de coreografias, mas também assegurar que a capacidade de cada serviço não seja extrapolada.

De forma semelhante, Wang *et al.* [131] propõem selecionar serviços para atender a demanda de diferentes provedores de coreografias, onde eles calculam o serviço candidato ideal para cada requisito de QoS usando a estratégia *best-QoS* e, em seguida, verificam a capacidade dos serviços candidatos escolhidos. Se qualquer serviço selecionado estiver sobrecarregado, então o problema é transformado em um problema de emparelhamento de peso máximo. Nesse caso, o problema de seleção de serviços é transformado em um problema de emparelhamento entre os requisitos de QoS dos provedores de coreografias e as ofertas de QoS dos serviços candidatos e empregam o método húngaro para realizar uma seleção global, de forma a atender todas as requisições.

Cardellini *et al.* [22] propuseram MOSES, um serviço de *broker* para o gerenciamento adaptativo de orquestrações de serviços lidando com fluxos de requisições. O objetivo deste *broker* é adaptar dinamicamente a configuração de uma orquestração para assegurar os SLAs negociados com diferentes classes de provedores de coreografias. A abordagem considera os SLAs que especificam limites superiores de um requisito de QoS, buscando melhorar a QoS percebida pelo provedor de coreografias. O gerenciamento do processo de adaptação é baseado em um algoritmo de seleção de serviços que usa programação linear, buscando minimizar o custo do *broker* de serviço enquanto garante a QoS negociada com os diferentes papéis nas orquestrações.

Jin et al. [63] propuseram uma abordagem que combina os algoritmos de colônia de formigas, genético e *Kuhn-Munkres* para resolver a seleção de serviços sensível à QoS para múltiplas composições. Esta abordagem considera que vários usuários requisitam uma mesma coreografia e transformam o problema de seleção de serviços em um problema de emparelhamento entre os requisitos de QoS do provedor de coreografias e as ofertas de QoS dos serviços candidatos. Para isso, requisitos de QoS global são decompostos em requisitos de QoS local e, em seguida, utilizam a estratégia *best-QoS* para selecionar serviços com sensibilidade à capacidade. A abordagem considera que os serviços compostos têm os mesmos requisitos funcionais e um único requisito de QoS global é definido. De forma semelhante, He *et al.* [55] propõem um modelo de

seleção de serviços para múltiplos provedores de coreografias com diferentes requisitos de QoS em aplicações multi-inquilinos (*multi-tenant*), onde cada serviço pode ser selecionado por um único provedor de coreografias. Portanto, sem compartilhamento de serviços, o que pode levar à subutilização de serviços dedicados às coreografias individuais. Em alguns casos, coreografias podem não ser implantadas devido à indisponibilidade de serviços.

No trabalho de Zhu *et al.* [155], os autores apresentam uma abordagem para a seleção de serviços sensível à QoS para múltiplos provedores de coreografias com base em seus requisitos de QoS. Para reduzir a sobrecarga computacional no processo de seleção para múltiplos provedores de coreografias, é reduzido o espaço de busca usando o conjunto ótimo de Pareto. Neste conjunto, aplicando a estratégia *best-QoS*, os serviços com os valores de QoS mais altos para alguns atributos de QoS, ao mesmo tempo que possui valores equivalentes para outros, são candidatos. Em seguida, soluções ótimas deste conjunto são selecionadas para os múltiplos provedores de coreografias aplicando o algoritmo de colônia de abelhas, enquanto se garante os requisitos de QoS.

Kang *et al.* [66] discutem que múltiplas requisições para um serviço que possui a melhor QoS dentre os serviços candidatos, pode levar à sua sobrecarga. Para esse problema, os autores propõem uma abordagem de seleção de serviços que considera a estratégia de avaliação de QoS *best-fit*. Além disso, um serviço somente pode ser selecionado se sua capacidade máxima atual permitir.

Baseado no problema tratado por Kang *et al.* [66], outras abordagens foram apresentadas para seleção de serviços para múltiplos usuários. Jongtaveesataporn e Takada [64] discutem um mecanismo que gerencia a ocorrência de requisições com restrições de QoS próximas e distribui entre serviços candidatos com oferta de QoS similares. O processo de encontrar uma lista de serviços baseia-se na estratégia *best-fit*. Entretanto, as abordagens [66] e [64] que utilizam a estratégia *best-fit* são aplicadas em casos de seleção de serviços individuais entre múltiplos usuários com cargas iguais. Entretanto, serviços compostos também podem ser requisitados por múltiplos usuários com cargas heterogêneas, inclusive, coreografias podem compartilhar um conjunto de serviços.

3.2.2 Multicoreografias de serviços

Além do fluxo de requisições, o segundo aspecto que deve ser considerado é o número de coreografias de serviços que devem ser implantadas, podendo impactar diretamente no fluxo de requisições sobre serviços selecionados. Na maioria das abordagens propostas na literatura, apenas uma única coreografia de serviços é explicita-

mente considerada, como em [153]. Entretanto, alguns trabalhos lidam diretamente com múltiplas coreografias.

Por exemplo, Ardagna e Mirandola [7] propuseram uma abordagem que trata o problema de seleção de serviços, onde várias composições são consideradas ao mesmo tempo e múltiplas requisições para uma mesma composição devem ser tratadas de forma simultânea. A abordagem de seleção de serviços adota a estratégia *best-QoS* e é sensível à capacidade. Os requisitos de QoS para cada fluxo de requisições são garantidos pelo valor médio de QoS requisitado por vários provedores de coreografia. Entretanto, a seleção de serviços é realizada para cada composição individual, seguindo algum critério de ordenação, como composições com requisitos de QoS mais severos ou maior carga.

Qian, Xuejun e Yanchun [28] lidam com sistemas baseados em serviços multi-inquilinos, onde serviços podem ser compartilhados por coreografias com diferentes requisitos de QoS. Eles argumentam que tratando igualmente todos os serviços em uma coreografia, os serviços compartilhados pelos inquilinos podem levar a violações de QoS ou desperdício de recursos. No entanto, eles não usam a seleção de serviços para lidar com violações de QoS, mas desenvolveram uma estratégia de monitoramento de serviços orientada a SLA para garantir os requisitos dos serviços compartilhados.

Shen *et al.* [113] discutem uma abordagem para otimizar a seleção de serviços sensível à QoS para processos concorrentes em um ambiente com restrições sobre a capacidade dos serviços. Nesta abordagem, a seleção de serviços é realizada separadamente para cada provedor de coreografias usando a estratégia *best-QoS*. No caso de qualquer conflito, ou seja, um serviço necessário para uma composição é requisitado por outra composição, um mecanismo de tratamento de erros é invocado. Neste caso, provedores de coreografia afetados conduzem uma negociação baseada em leilão, seguido de uma nova seleção. O objetivo da negociação entre provedores de coreografias é lidar com as questões de concorrência indesejável dos serviços e a espera extra, uma vez que algumas composições podem requisitar serviços com alta QoS e subutilizá-lo, enquanto outras composições não podem ser implantadas devido à falta de serviços viáveis.

Wu *et al.* [139] propuseram uma abordagem para seleção de serviços que usa a estratégia *best-fit* com sensibilidade à capacidade para seleção de serviços para coreografias sob demanda. Eles discutem que ao aplicar estratégia de *best-QoS* para várias composições, alguns serviços candidatos ficarão sobrecarregados facilmente e sua QoS prometida não poderá mais ser garantida. Ao mesmo tempo, outros serviços candidatos com bom desempenho de QoS não serão explorados adequadamente. Um *broker* de serviços é utilizado para facilitar a implementação desta estratégia, atuando

como intermediário entre os usuários e provedores de serviços. Primeiro, o *broker* adquire instâncias de serviços especificando a classe de QoS requerida e o número de requisições simultâneas. Em seguida, provisiona várias orquestrações de serviços com diferentes classes de QoS, analisando a capacidade dos serviços candidatos disponíveis, inclusive levando em consideração seu compartilhamento.

3.3 Seleção de serviços sensível ao custo

Um desafio importante sobre coreografias de serviços é o desenvolvimento de abordagens de seleção de serviços eficientes em relação ao gerenciamento da QoS e do custo [68]. Essa seção apresenta trabalhos que discutem abordagens para seleção de serviços levando em consideração o custo.

Na maioria das abordagens, o custo é tido como um atributo de QoS que deve ser minimizado enquanto se aplica a estratégia de avaliação *best-QoS* a todos os atributos presentes no modelo de serviço proposto. Por exemplo, em Zeng *et al.* [153], discutido anteriormente, os autores apresentam uma abordagem para resolver o problema de seleção de serviços sensível à QoS usando a avaliação *best-QoS*, onde o custo, valor pago ao provedor para utilizar um serviço, é tratado como um atributo de QoS que deve ser minimizado.

Xu e Jennings [140] apresentam um algoritmo de seleção de serviços que lida com custo variável. Os autores buscam a minimização de custos na composição de serviços ao prover suporte a descontos de acordo com o tempo de utilização. Para isso, eles utilizam informações de provedores de serviços que oferecem serviços dentro de um intervalo de tempo especificado com certo nível de desconto para grupos de serviços. Sendo assim, o algoritmo de seleção de serviços proposto identifica a combinação de serviços com o custo médio esperado mais baixo para acesso dentro de um intervalo de tempo definido. O algoritmo de seleção analisa todas as combinações possíveis de serviços que poderiam ser usadas para construir uma composição. Para cada combinação são analisadas as regras sensíveis ao tempo nos custos dos serviços, onde os modelos de cobrança são pré-processados para cada serviço. Informações sobre a oferta de QoS e capacidade dos serviços não são levados em consideração, bem como o compartilhamento de serviços.

Ramacher e Mönch [106] propõem uma abordagem de seleção de serviços na presença de requisitos de QoS global que busca a minimização dos custos. Eles argumentam que modelos de cobrança complexos requerem uma generalização do problema de seleção de serviços, em termos de determinar uma seleção para fluxo de requisições durante um período de tempo. Para isso, eles consideram uma seleção de serviços para várias composições de uma só vez, permitindo o compartilhamento

de serviços. Eles usam como base de resolução um modelo de otimização baseado em programação inteira mista para determinar uma seleção de serviços que minimize o custo esperado em um período considerando descontos em quantidade e pacotes, cobrança baseada em assinatura e taxas de admissão. Além da minimização de custos, o modelo proposto aplica a estratégia *best-QoS* para avaliar os requisitos de QoS global.

Ao aplicar a estratégia *best-QoS*, tais abordagens buscam garantir que cada provedor de coreografias possa selecionar serviços com os melhores valores para todos os atributos de QoS, que são, geralmente, conflitantes em relação ao custo. Além disso, a estratégia *best-QoS* permite que um serviço possa ser compartilhado por coreografias com requisitos de QoS assimétricos, levando à subutilização. Por exemplo, um serviço com tempo de resposta de *0.3s* pode ser compartilhado pela coreografia A, que requer um tempo de resposta menor do que *1s*, e pela coreografia B que requer um tempo de resposta menor do que *8s*.

Já Wu *et al.* [139], discutido anteriormente, apresentam uma abordagem para seleção de serviços que usa a estratégia *best-fit*, onde o custo por requisição é caracterizado pelo nível de QoS fornecido. Ao usar a estratégia *best-fit* para avaliar a QoS e custo dos serviços, tais abordagens buscam garantir que os recursos não sejam desperdiçados e que os requisitantes recebam apenas o necessário, diminuindo assim o custo em relação aos recursos utilizados pelos serviços. Além disso, essa estratégia permite que coreografias com requisitos de QoS próximos possam compartilhar serviços de forma eficiente.

Em outros trabalhos, o custo é diretamente relacionado aos recursos onde os serviços são executados. Os trabalhos propostos por Wang *et al.* [130], Khanouche et al. [68], Li *et al.* [77] e Gomes [51] focam na seleção de recursos sensível ao custo para implantação de serviços previamente selecionados, o que foge do escopo desta tese. Mas, como esses trabalhos lidam com alguns aspectos relacionados à abordagem proposta, eles são discutidos a seguir.

Wang *et al.* [130] propõem um modelo de coreografia de serviços para minimizar custos para implementação de processos intensivos, como mineração de dados e processamento de imagem. Durante a execução de uma coreografia, a transferência de dados influencia o desempenho, uma vez que conjuntos de dados podem ser trocados entre diversos serviços que compõem uma coreografia. Esses conjuntos de dados podem ser replicados em diferentes *data centers*, onde o custo de acesso de cada réplica em cada *data center* é diferente. Logo, o custo dos serviços é definido pela localização do recurso onde o serviço será implantando e pela quantidade de dados transferidos. Diante desse cenário, Wang *et al.* [130] apresentam um modelo, baseado em algoritmos bio-inspirados, que realiza a composição de serviços em três fases. Na primeira fase é realizada a seleção dos serviços candidatos de acordo com a estratégia *best-QoS*.

Na segunda fase é realizada a seleção dos *data centers*, levando em considerações os serviços candidatos selecionados. Por fim, a pontuação de cada solução é calculada, onde a composição que possui o menor custo é selecionada.

Khanouche et al. [68] propõem uma abordagem de seleção de serviços sensíveis à QoS e à energia para composição de serviços no ambiente de Internet das Coisas. A ideia principal da abordagem é que sempre é possível economizar energia reduzindo o nível de QoS sem afetar a satisfação do provedor de coreografias. Ao diminuir o consumo de energia ao mesmo tempo que se garante os requisitos de QoS, o custo dos serviços também é minimizado sem afetar os usuários. A abordagem de seleção proposta consiste em pré-selecionar os serviços candidatos que oferecem os níveis de QoS mais altos de acordo com as preferências do provedor de coreografias. Nessa fase, utiliza-se uma estratégia de otimização lexicográfica e técnica de relaxamento de restrições de QoS. Em seguida, a fim de reduzir o consumo de energia de uma coreografia de serviços sem afetar a satisfação dos requisitos de QoS, os serviços mais adequados, dentre os pré-selecionados, são escolhidos usando o conceito de dominância relativa de serviços por Pareto. A dominância relativa de um serviço candidato depende do seu perfil de energia, atributos de QoS e das preferências do provedor de coreografias.

No contexto de redes de sensores sem fio integrado à Internet das Coisas, um dos problemas é a alocação eficiente de recursos em múltiplas redes de sensores sem fio [142, 46]. Em Li *et al.* [77], os autores apresentam uma abordagem orientada a serviços para tratar da alocação eficiente de recursos para múltiplas coreografias em redes de sensores sem fio heterogêneas. Um dos desafios considerados pelos autores remete ao fato de que as aplicações baseadas em coreografias podem competir pelos recursos oferecidos pelos nós sensores. Assim, o objetivo é garantir dinamicamente o compartilhamento dos recursos disponíveis para otimizar a utilização dos recursos enquanto cumpre os requisitos de QoS específicos de cada coreografia. Nesse sentido, a abordagem busca alocar as coreografias, simultâneas ou não, aos nós de sensores com base na utilidade de cada coreografia a partir dos requisitos de QoS (QoS da aplicação e QoS da rede). Para realizar a alocação, eles implementaram um servidor *Web* que atua como um provedor de serviços para responder as requisições dinâmicas das coreografias. Diferentes alocações para os papéis das coreografias sobre os nós do sistema podem consumir diferentes quantidades de energia bem como obter diferentes níveis de QoS, gerando diferentes custos. Logo, as incumbências do servidor *Web* são decompor uma coreografia em tarefas (unidades de execução) e decidir como enviar as tarefas recebidas para as redes de sensores sem fio de forma a garantir a QoS e gerenciar de forma eficiente o consumo de energia.

De forma semelhante, Gomes [51] apresenta uma abordagem para implantação eficiente de um conjunto de coreografias de serviços sujeitas a restrições não-

funcionais. Dado o conjunto de serviços que compõem as coreografias, juntamente com as restrições relacionadas, a abordagem realiza a seleção e a alocação dos recursos em um ambiente de nuvem híbrida com múltiplos provedores, enquanto reduz os custos associados à utilização dos recursos e o atraso de comunicação entre os serviços.

3.4 Comparativo

Nesta seção realizamos um comparativo dos trabalhos apresentados, considerando as características relevantes para seleção de serviços sensível à QoS discutidas nas Seções 3.1, 3.2 e 3.3. Para facilitar a comparação, os trabalhos foram divididos em duas tabelas de acordo com as principais características em comum. Inicialmente, analisamos os trabalhos que focam na seleção de serviços sensível à QoS para aplicações baseadas em composições de serviços. Em seguida, são analisados os trabalhos que lidam com aplicações baseadas em composições de serviços que utilizam compartilhamento de serviços e/ou buscam otimizar algum aspecto durante a seleção dos serviços candidatos.

A Tabela 3.1 mostra os trabalhos relacionados que focam na seleção de serviços sensível à QoS. Esses trabalhos são analisados com base em cinco aspectos:

- *Escopo da QoS*: indica como os requisitos de QoS são considerados nas atividades relacionadas à seleção de serviços, podendo ser tratados por serviço individual (*local*), por coreografia (*global*) ou como *híbrido*, onde os requisitos de QoS são tratados serviço individual e por coreografia.
- *Avaliação de QoS*: indica qual estratégia de avaliação é utilizada para classificar os serviços candidatos em relação aos requisitos de QoS, ou seja, a estratégia *best-QoS* e a estratégia *best-fit*.
- *Base de resolução*: indica as principais técnicas de otimização utilizadas no processo de seleção de serviços. Para isso, os trabalhos foram classificados com base em três critérios: 1) *exato*: utilizam métodos exatos para resolução, inclusive, podendo levar a resolução a uma complexidade intratável; 2) *bio*: utilizam metaheurísticas baseada ou inspirada na natureza; e 3) *heurísticas*: utilizam outros tipos de heurísticas para buscar uma solução satisfatória.
- *Otimização adicional*: especifica estratégias complementares para tornar a seleção de serviços mais eficiente em termos computacionais, como pela redução do espaço de buscas ou relaxamento nos requisitos de QoS.
- *Referencial de otimização*: indica se o número de requisições que um serviço pode receber é levado em consideração no processo de seleção.

Aspectos	Escopo de QoS			Avaliação de QoS		Base de resolução			Otimização adicional		Referencial de otimização	
Critérios / Abordagem	Local	Global	Híbrido	Best-QoS	Best-Fit	Exato	Bio	Heurísticas	Redução	Relaxamento	Por requisição	Por fluxo
Zeng *et al.* [153]	✓	✓		✓		✓				✓	✓	
Canfora *et al.* [20]		✓		✓			✓		✓		✓	
Ardagna e Pernici [8]	✓	✓		✓		✓			*N*/*I*	*N*/*I*	✓	
Yu *et al.* [149]		✓		✓				✓	*N*/*I*	*N*/*I*	✓	
Mabrouk *et al.* [83]		✓		✓				✓	✓		✓	
Alrifai, Skoutas, e Risse [6]		✓		✓		✓			✓		✓	
Kang *et al.* [66]	✓				✓			✓	*N*/*I*	*N*/*I*		✓
Alrifai e Risse [5]			✓	✓		✓				✓	✓	
Wang *et al.* [133]		✓		✓		✓			*N*/*I*	*N*/*I*		✓
Wang *et al.* [131]	✓			✓				✓	*N*/*I*	*N*/*I*		✓
Peng e Chang-song [105]		✓		✓			✓			✓	✓	
Liu [80]			✓	✓			✓		✓			✓
Palade *et al.* [101]		✓		✓			✓		N/I	N/I	✓	

Legenda: ✓ : Contemplado; *N*/*I* : Não implementado.

Tabela 3.1: *Comparativo dos trabalhos para seleção de serviços sensível à QoS.*

A Tabela 3.1 apresenta diversos trabalhos de pesquisa na área de seleção de serviços, juntamente com as técnicas e ferramentas resultantes, que lidam com garantias de QoS utilizando a estratégia de avaliação *best-QoS* aplicando programação linear, como em [153], [8], [6] e [133]. Para alcançar maior escalabilidade requerida em ambientes dinâmicos, outras abordagens baseadas em heurística também foram propostas como em [149] e [83]. Canfora *et al.* [20] e Peng e Changsong [105] exploram o espaço de busca por meio de diversas estratégias utilizando heurísticas bio-inspiradas. Outros trabalhos utilizam uma abordagem híbrida, combinando métodos de seleção local e global, para otimizar o processo de seleção de serviços como em [5] e [80].

Entretanto, a maioria destas soluções consideram uma única requisição enviada para uma coreografia de serviços e visam cumprir os requisitos de QoS para uma requisição particular. Essa consideração pode levar à sobrecarga e à degradação do desempenho de um serviço quando ocorre uma variação da carga. A variação da carga pode ocorrer em razão de múltiplas requisições para uma mesma coreografia por diferentes provedores de coreografias ou devido ao compartilhamento não controlado de serviços entre múltiplas coreografias. Em tais casos, os serviços podem não garantir a QoS ofertada, aumentando o número de violações de QoS e, por consequência, o custo das coreografias. Além disso, os usuários podem ficar insatisfeitos devido ao não cumprimento dos requisitos de QoS acordados.

A seleção de serviços ciente do fluxo de requisições é necessária para ambientes onde ocorrem variações de carga, permitindo eficiência e escalabilidade ao responder aos provedores de coreografias [23]. Liu [80] lida com fluxo de requisições dirigidos à uma mesma coreografia de serviços por múltiplos usuários, enquanto Wang *et al.* [131] lida com fluxo de requisições dirigidos a um mesmo serviço, onde ambos utilizam a estratégia *best-QoS*. Outros trabalhos lidam com fluxo de requisições utilizando a estratégia *best-fit*, como em [66] e [62], buscando aumentar o número de usuários atendidos e tornar o uso dos recursos pelos serviços selecionados mais eficiente.

Dos trabalhos apresentados na Tabela 3.1, todos lidam com apenas uma única composição de serviços. Essa estratégia é adequada, por exemplo, quando os provedores de coreografias utilizam serviços dedicados por composição ou podem hospedar os próprios serviços. Por outro lado, nem sempre é possível ou desejável que as composições de serviços tenham serviços dedicados durante sua execução [82, 113]. Por exemplo, os provedores de coreografias que utilizam um serviço com alto custo, relacionado com a licença de software ou com despesas na alocação de recursos, podem querer compartilhar o serviço com outros provedores de coreografias para diminuir seus gastos. Por fim, alguns serviços de terceiros podem não permitir a replicação, exigindo o compartilhamento para satisfazer múltiplas coreografias.

A Tabela 3.2 apresenta os trabalhos que lidam com múltiplas composições de

serviços e/ou buscam otimizar algum aspecto durante a seleção dos serviços candidatos, podendo controlar o compartilhamento de serviços. Os trabalhos nesta categoria são analisados com base em quatro aspectos:

- *Multicoreografias*: indica se a seleção de serviços é realizada para coreografias *isoladas* ou se as mesmas são agrupadas e processadas em conjunto.
- *Sensibilidade à capacidade*: aponta se a capacidade dos serviços candidatos são levados em consideração no processo de seleção, onde *infinito* significa que serviços tem capacidade infinita e *Sensível à capacidade* significa que serviços são compartilhados de forma controlada, ou seja, a informação da capacidade é utilizada no processo de seleção de serviços.
- *Dimensões adicionais de otimização*: estabelece objetivos complementares na seleção de serviços, como eficiência na utilização dos *recursos* utilizados pelos serviços, minimização dos *custos* e minimização do consumo de *energia*.
- *Aplicações*: indica o tipo de aplicação na qual a solução proposta no trabalho trata. Eles são classificados em três grupos:
  - Aplicações baseadas em *serviço*: onde a seleção de serviços sensível à QoS é utilizada para aplicações SOA tradicional;
  - Aplicações baseadas em serviços de *cloud*: onde a seleção de serviços sensível à QoS busca reduzir o custo de execução de uma aplicação através da seleção de recursos para executar tais serviços;
  - Aplicações para Internet das Coisas (*IoT*): onde a seleção de serviços sensível à QoS busca aumentar a disponibilidade dos serviços, reduzindo o consumo de energia, ao mesmo tempo que tenta minimizar os custos dos serviços.

A maioria dos trabalhos apresentados na Tabela 3.2 realiza a seleção de serviços levando em consideração várias coreografias. Entretanto, alguns trabalhos impõem restrições tal que a seleção de serviços é realizada para cada coreografia individualmente. Por exemplo, Cardellini *et al* [22], Jin et al. [63] e He *et al.* [55] consideram que vários usuários requisitam uma mesma coreografia com um único requisito de QoS global. Esses trabalhos têm como objetivo determinar um grupo de coreografias concretas que consiga atender ao fluxo de requisições de múltiplos usuários. Por outro lado, em Ardagna e Mirandola [7], Wu *et al.* [139], Zhu *et al* [155] e Qian, Xuejun e Yanchun [28], as coreografias são processadas individualmente, de forma independente umas das outras.

Quando múltiplas coreografias devem ser implantadas em um mesmo ambiente de execução, existe uma grande probabilidade de existir um subconjunto de papéis comuns entre elas. Por consequência, elas podem competir por um mesmo sub-

Aspectos / Abordagens	Multicoreografias	Sensibilidade à capacidade	Dimensões adicionais de otimização	Aplicação
Ardagna e Mirandola [7]	Isoladas	Sensível à capacidade	Custo	Serviços
Xu e Jennings [140]	Isoladas	Infinito	Custo	Serviços
Ramacher e Mönch [106]	Interdependentes	Sensível à capacidade	Custo	Serviços
Cardellini *et al.* [22]	Isoladas	Sensível à capacidade	Custo	Serviços
Jin *et al.* [63]	Isoladas	Sensível à capacidade	N/I	Serviços
He *et al.* [55]	Isoladas	Sensível à capacidade	Custo	*Cloud*
Shen *et al.* [113]	Isoladas	Sensível à capacidade	Custo	Serviços
Wang *et al.* [130]	N/A	N/A	Custo	*Cloud*
Wu *et al.* [139]	Isoladas	Sensível à capacidade	Custo	Serviços
Li *et al.* [77]	Interdependentes	Sensível à capacidade	Energia	*IoT*
Khanouche *et al.* [68]	N/A	N/A	Energia	*IoT*
Zhu *et al.* [155]	Isoladas	Sensível à capacidade	N/I	*Cloud*
Gomes [51]	Interdependentes	Sensível à capacidade	Recurso	*Cloud*
Qian, Xuejun e Yanchun [28]	Interdependentes	Sensível à capacidade	Custo	*Cloud*
Palade *et al.* [101]	N/A	Sensível à capacidade	Recurso	*Cloud*

Legenda: ✓ : Contemplado; N/I : Não implementado, N/A : Não se aplica.

Tabela 3.2: *Comparativo dos principais trabalhos relacionados com a seleção de serviços para múltiplas coreografias.*

conjunto de serviços, levando ao compartilhamento de serviços entre as coreografias. Nesses casos, se um serviço se envolver em várias coreografias, haverá uma dependência entre os níveis de QoS que o serviço pode contribuir para cada uma delas [97]. Por este motivo, as abordagens de seleção de serviços para múltiplas coreografias devem ser sensíveis à capacidade máxima de cada serviço, de forma a garantir que o número de requisições não exceda sua capacidade corrente.

Em complemento, as abordagens discutidas anteriormente não lidam diretamente com o compartilhamento de serviços entre coreografias no processo de seleção de serviços, garantindo apenas que a capacidade de cada serviço selecionado não será extrapolada. Sendo assim, o processamento isolado das coreografias, de forma independente umas das outras, favorece as coreografias processadas primeiro em relação às demais coreografias. Além disso, com o aumento do número de coreografias no ambiente, as coreografias processadas no final podem não ser implantadas devido à indisponibilidade de serviços adequados, mesmo com uma grande quantidade de serviços candidatos.

Essas abordagens que lidam com múltiplas coreografias e realizam a seleção de serviços para cada coreografia individualmente não conseguem explorar alguns aspectos que são possíveis ao utilizar uma visão global das coreografias, como entidades de primeira classe na solução. Por exemplo, ao analisar múltiplas coreografias é possível analisar durante o processo de seleção quais papéis são compartilhados por grupos de coreografias, bem como a carga agregada de tais coreografias. A limitação de tais abordagens ocorre porque vários esquemas de seleção de serviços apresentados na li-

teratura sugerem o mapeamento apenas do conjunto de QoS dos serviços para valores de utilidade, sem levar em consideração a utilização dos recursos pelos serviços e o custo em relação aos recursos utilizados pelos serviços selecionados.

Há várias soluções que consideram o custo dos serviços durante a seleção, como em [112], [73], [140] e [106], onde custo é tido como um atributo de QoS que deve ser garantido para satisfazer os requisitos dos provedores de coreografias. Por exemplo, em Parejo *et al.* [104] e Huo *et al.* [58], os autores apresentam abordagens heurísticas para resolver o problema de seleção de serviços sensível à QoS. O custo, valor pago ao provedor para utilizar um serviço, é tratado como um atributo de QoS que deve ser minimizado utilizando a estratégia *best-QoS*. A abordagem busca garantir que cada provedor de coreografias possa selecionar serviços com os melhores valores para todos os atributos de QoS, que são, geralmente, conflitante em relação ao custo.

Quando essas abordagens utilizam a estratégia *best-QoS* e assumem que as cargas sobre diferentes coreografias são iguais, como [63] e [58], elas não conseguem garantir o bom desempenho dos serviços, uma vez que as cargas reais das coreografias podem ser diferentes. Essa estratégia é adequada quando se lida apenas com uma coreografia por vez ou quando os serviços selecionados são exclusivos de cada coreografia, ou seja, uma instância de cada serviço é utilizada apenas por uma coreografia e novas instâncias do serviço podem ser criadas sob demanda.

Por outro lado, quando um serviço não consegue garantir a QoS devido a uma sobrecarga, o usuário deve ser compensado financeiramente por tais violações de QoS. Além disso, há desperdício de recursos quando um serviço com a melhor QoS é oferecido para uma coreografia com requisito de QoS leve. De modo complementar, essa estratégia não é adequada para tratar do balanceamento das requisições entre os serviços candidatos, de forma que ela pode levar ao excessivo compartilhamento de um subconjunto de serviços enquanto que a maioria dos serviços quase não são utilizados, mesmo sendo adequados para a maioria das requisições [113].

Para lidar com tais situações alguns trabalhos propõem uma abordagem de seleção de serviços que considera a distância entre a QoS requisitada e a QoS ofertada por cada serviço candidato, ou seja, eles utilizam a estratégia *best-fit* para medir o grau de correspondência, como em Kang *et al.* [66] e Wu *et al.* [139]. Ao utilizar a estratégia *best-fit*, a seleção de serviços sensível à QoS pretende garantir que os recursos não sejam desperdiçados e que os requisitantes recebam apenas o necessário, fazendo uso eficiente de serviços entre várias requisições [78, 68]. Além do mais, exceder a demanda de QoS normalmente pode não melhorar a satisfação dos requisitos por parte dos usuários [139].

Pelo lado dos provedores de serviços e provedores de computação em nuvem, o custo pode ser visto de uma forma mais ampla. Por exemplo, os SLAs, utilizados

para guiar a QoS esperada pelos provedores de coreografias, podem incrementar as perdas financeiras dos provedores de serviços, caso ocorram violações de SLAs. Em contrapartida, ao fornecer uma QoS mais alta aos provedores de coreografias para minimizar as violações de SLAs, os provedores de serviços também incrementam seus custos ao utilizar mais recursos computacionais. Vários trabalhos voltados para área de computação em nuvem utilizam técnicas de SOA para minimizar os custos dos provedores de serviços enquanto garantem a QoS ofertada. As estratégias propostas buscam minimizar custos com energia, como em [68] e [77], e/ou utilização dos recursos, como em [130] e [51].

Da perspectiva desta tese, as coreografias de serviços são criadas e mantidas por organizações públicas e/ou privadas, onde o custo para execução dos serviços selecionados é um requisito importante. Nessa visão, os provedores de coreografias, ao oferecer várias aplicações baseadas em coreografias, devem explorar ao máximo a possibilidade de minimizar os custos em relação aos recursos utilizados pelos serviços selecionados, seja de forma a minimizar seu custo operacional e diminuir também o custo do seu usuário. O processo de seleção de serviços também deve analisar a carga agregada e a capacidade máxima dos serviços selecionados, uma vez que o custo de um serviço é baseado, além da QoS oferecida, no número de requisições máxima que ele consegue atender em determinado período de tempo [17].

Ao adicionar na seleção de serviços sensível à QoS informações sobre carga agregada e capacidade, pode-se buscar uma melhor utilização dos recursos alocados por cada serviço através do compartilhamento que, por consequência, pode gerar uma economia aos provedores de coreografias, como em [131]. Para realizar esse ajuste, busca-se aumentar a taxa de utilização dos serviços selecionados analisando a carga agregada e a capacidade dos serviços.

Esse problema é bem consolidado na área de implantação de aplicações sensíveis ao custo em ambientes de nuvem, como em Gomes [51], e aplicações para Internet das Coisas, como em Li *et al.* [77] e Khanouche *et al.* [68]. Todavia, essas soluções lidam diretamente na alocação de recursos, onde os serviços que serão implantados já foram selecionados em uma etapa anterior. O principal objetivo desses trabalhos é gerar economia financeira para os provedores de recurso e não aos provedores de coreografias, o que foge do escopo dessa tese.

Conforme destacado, a seleção de serviços sensível à QoS e à capacidade para múltiplas coreografias com objetivo de minimizar o custo ao provedor de coreografias não é tratada em conjunto em nenhum trabalho. Argumentamos que a seleção de serviços ciente do possível compartilhamento de serviços entre várias coreografias pode promover o uso eficiente de recursos e diminuir o custo geral de implantação e utilização de serviços. Como resultado, o sucesso da execução de várias coreografias

depende de uma abordagem de seleção de serviços que considere o impacto da carga agregada resultante na QoS fornecida pelos serviços.

Para o nosso melhor conhecimento, até o momento, o custo foi abordado separadamente, ou seja, o provedor de coreografias realiza a seleção de serviços analisando os requisitos de QoS e o seu orçamento, já o provedor de serviços garante os recursos necessários para executar os serviços selecionados de acordo com sua oferta de QoS e o orçamento informado. No entanto, é de se destacar que as soluções de cada problema estão intimamente relacionadas, uma vez que o custo relacionado ao recurso utilizado para executar um serviço é dependente da QoS requisitada com certa carga estimada.

3.5 Conclusão

Esse capítulo mostrou o estado da arte sobre seleção de serviços sensível à QoS para implantação de coreografias de serviços. Diante do cenário tratado nesta tese, foram discutidos uma série de problemas que devem ser considerados quando lidamos com seleção de serviços para múltiplas coreografias. Inicialmente discutimos as propostas para seleção de serviços sensível à QoS, identificando as estratégias de avaliação de QoS utilizadas para classificação dos serviços e como os requisitos de QoS são considerados na seleção.

Em seguida, os trabalhos que lidam com compartilhamento de serviços entre múltiplas coreografias foram analisados. Nesse sentido, foram identificados os trabalhos que analisam a capacidade dos serviços candidatos na seleção, de forma que o compartilhamento de serviços não cause degradação da QoS requisitada. Podemos perceber que a maioria das abordagens tratam da seleção de serviços para cada coreografia de forma individual, não lidando diretamente com grupos de coreografias. Por fim, analisamos os trabalhos que lidam com a seleção de serviços sensível ao custo. A maioria dos trabalhos que lidam com custo propõem estratégias de seleção eficiente de recursos computacionais utilizados pelos serviços que compõem as coreografias, e não diretamente na seleção de serviços.

Diante dessa análise, podemos perceber que há diversas lacunas nas propostas encontradas. Essas limitações dizem respeito principalmente à ausência de uma visão mais pragmática do problema, na qual é assumida a necessidade da seleção de serviços ser realizada considerando, além da sensibilidade da QoS, a análise da relação entre carga das requisições e capacidade dos serviços. Para preencher essas lacunas, propomos uma abordagem que leva em consideração essas questões e as demais limitações discutidas nesse capítulo.Essa contribuição é discutida a partir do próximo capítulo.

CAPÍTULO 4

Modelo de QoS e custo de serviços

O Capítulo 3 analisou as principais lacunas na literatura para o problema de seleção de serviços sensível à QoS. Essas limitações dizem respeito principalmente à ausência de uma visão mais pragmática ao lidar com múltiplas coreografias, na qual é assumida a necessidade da seleção de serviços ser realizada considerando, além da sensibilidade da QoS, a análise da relação entre carga das requisições e capacidade dos serviços.

Diante dessa análise, propomos uma abordagem que contempla todo o processo de seleção de serviços para múltiplas coreografias, como apresentado no Capítulo 1. Este capítulo inicia o detalhamento da abordagem, formalizando o modelo de serviços, utilizado durante todo o processo de seleção, conforme ilustrado na Figura 1.3.

O capítulo está assim organizado. A Seção 4.1 apresenta o modelo de representação dos serviços. A Seção 4.2 define como os critérios de QoS são definidos junto com o modelo de representação dos serviços, formalizando os requisitos de QoS dos provedores de coreografias e a oferta de QoS pelos serviços candidatos. Em seguida, na Seção 4.3, a função de utilidade para classificação dos serviços com base em sua oferta de QoS é apresentada. Por fim, na Seção 4.4, é proposto o modelo de custo de um serviço baseado no nível de QoS oferecido e na capacidade máxima contratada junto ao provedor.

4.1 Modelo de serviços

Dado um conjunto de papéis $\mathcal{P}$ pode-se definir um conjunto de coreografias abstratas SC, onde para cada coreografia abstrata $c_i \in SC$, especificada pelo provedor, é definido um conjunto de papéis p_i, onde cada papel $p_i \subseteq \mathcal{P}$ define uma atividade específica. Para cada papel o provedor de coreografias determina um conjunto de requisitos de QoS. Além disso, para cada p_i é associado um conjunto de serviços candidatos S, onde cada $s_i \subseteq S$ é capaz de realizar p_i.

No modelo proposto nesta tese é assumido que existe um conjunto de serviços candidatos S, que é definido como uma união de tipos de serviço. Cada tipo de serviço $t_e \in \mathcal{T}$ (por exemplo, serviços de reserva de hotel) é usado para descrever um conjunto de serviços candidatos funcionalmente equivalentes (por exemplo, *Trivago* e *Booking*).

Para diferenciar os membros de um tipo de serviços suas ofertas de QoS devem ser consideradas. Para isso, adotamos um modelo de serviços baseado em QoS fundamentado em um conjunto de propriedades de QoS aplicáveis a todos os serviços, por exemplo, tempo de resposta e vazão. Vale ressaltar que esse modelo é extensível, ou seja, novas propriedades de QoS podem ser adicionadas sem alterar fundamentalmente as técnicas de seleção de serviço construídas sobre o modelo.

Semelhante a outros trabalhos encontrados na literatura [153, 6, 84, 134], assumimos que cada papel p_i é desempenhado por serviços candidatos de um único tipo de serviço t_e. Além disso, apenas um serviço candidato s_i é selecionado para cada papel p_i.

A entrada para o mecanismo de seleção de serviços consiste de um conjunto de coreografias, juntamente com um conjunto de serviços candidatos. O conjunto de coreografias de serviços, denotado por SC, contém z coreografias $c_1, c_2, ..., c_z$. O conjunto de serviços candidatos usados para compor as coreografias, denotado por S, contém m serviços $s_1, s_2, ..., s_m$. No processo de seleção de serviços, para cada coreografia $c_i \in SC$, $i \in [1...z]$, deve-se determinar um conjunto de serviços concretos $\mathcal{SS}$ que podem desempenhar o conjunto de papéis de cada coreografia c_i.

4.2 Critérios de QoS

O processo de seleção de serviços requer modelos que captem tanto os requisitos de QoS das coreografias, quanto as ofertas de QoS dos serviços candidatos [71]. Sendo assim, é necessário um modelo de serviços que seja baseado em QoS, permitindo que os requisitos de QoS sejam especificados para papéis individuais ou para uma coreografia como um todo, enquanto as ofertas de QoS são definidas para os serviços concretos, que estão disponíveis como candidatos para cumprir tais papéis.

Os requisitos e ofertas de QoS são especificados por meio de atributos de QoS, os quais são definidos de acordo com métricas de QoS [108]. Além disso, introduzimos uma classificação dos atributos de QoS em três grupos:

- *Alvo*: o valor da métrica, para um determinado serviço candidato, deve estar o mais próximo possível do valor requisitado. Por exemplo, se uma coreografia requerer um serviço com valor para a métrica disponibilidade maior ou igual

a 99.5%, o serviço candidato s_a, com oferta para a métrica disponibilidade com valor de 99.6%, é preferido em detrimento ao serviço candidato s_b, com oferta para a métrica disponibilidade com valor de 99.8%. Um serviço com valor para a métrica disponibilidade abaixo de 99.5% não é considerado como candidato, uma vez que ele não consegue atender o valor requisitado, ou seja, maior ou igual a 99.5%.

- *Requerido*: o valor da métrica não pode ser quantificada numericamente, mas deve ser satisfeita pelo serviço candidato, exemplos desse tipo de atributo são propriedades transacionais ou a moeda para pagamento em um determinado país. Por exemplo, uma coreografia pode requerer um serviço de pagamento que permita pagamento utilizando a moeda corrente do Brasil (real), logo apenas serviços de pagamento que provê suporte à moeda corrente do Brasil são considerados.
- *Otimizado*: o valor da métrica deve ser maximizado ou minimizado; exemplos desse tipo de atributo são vazão de dados e preço. Quando um atributo de QoS é definido como maximizado, diz-se que o atributo é positivo, pois quanto maior o valor, melhor a qualidade do serviço. Já quando um atributo de QoS é definido como minimizado, diz-se que o atributo é negativo, pois quanto menor o valor, melhor a qualidade do serviço.

Quando um provedor de coreografias especifica os atributos de QoS, ele deve classificá-los de acordo com esses três grupos. Esta tese considera quatro atributos de QoS para os serviços:

- Tempo de resposta: O tempo de resposta do serviço s mede a latência esperada em segundos entre o momento em que uma requisição é enviada ao serviço s e o momento em que os resultados são recebidos pelo requisitante. A duração da execução é calculada pela soma da latência de processamento e a latência de rede. Nesta tese a latência de rede é considerada a mesma para todas as requisições.
- Reputação: A reputação do serviço s é uma medida de sua confiabilidade, sendo definida como a razão entre o número de requisições que atendem à QoS ofertada sobre o número total de requisições ao serviço s.
- Disponibilidade: A disponibilidade do serviço s mede a probabilidade de que o serviço esteja acessível, sendo definido em relação a quantidade total de tempo (em segundos) em que o serviço s está disponível durante um intervalo previamente definido.
- Capacidade: A capacidade do serviço s define a quantidade de requisições por unidade de tempo que um serviço consegue atender.

O provedor de coreografias pode expressar as preferências fornecendo pesos que permitem que determinados atributos de QoS sejam considerados mais importantes do que outros. A soma de todos os pesos relativos dos atributos de QoS deve ser igual a 1. Por *default*, todos os atributos são classificados como *alvo* e com pesos iguais, caso o provedor não o faça explicitamente.

Sendo assim, um atributo de QoS pode ser formalmente definido como:

Definição 4.1 ***(Atributo de QoS)*** *Um atributo de QoS $\mathcal{A}$ é uma tupla $(\mathcal{M}, \square, \tau, \phi)$:*

$\mathcal{M}$ – métrica de QoS;
$\square$ – operador relacional tal que:

- *$\square \in \{<, \leq, =, \neq, >, \geq\}$ se houver relação de ordem total em $\mathbb{D}_{\mathcal{M}}$, ou*
- *$\square \in \{=, \neq\}$ se não houver relação de ordem total em $\mathbb{D}_{\mathcal{M}}$; e*

τ – valor para a métrica ($\tau \in \mathbb{D}_{\mathcal{M}}$);
ϕ – classificação do atributo de QoS ($\phi_i \in \{Alvo, Requirido, Otimizado\}$)

No modelo de representação de serviços baseado em QoS, um requisito de QoS para uma coreografia de serviços pode ser formalmente definido como:

Definição 4.2 ***(Requisito de QoS)*** *Um requisito C de QoS é representado por vetor r de k atributos de QoS $\{\mathcal{A}_1; \ldots; \mathcal{A}_k\}$.*

Para representar a oferta de QoS pelos serviços candidatos, cada serviço possui um conjunto de k propriedades de QoS, cujos valores são representados pelo vetor $q = q_1, ..., q_k$. Cada propriedade de QoS é uma tupla $q_i = (\mathcal{M}_i; \varphi_i)$, onde $\mathcal{M}_i$ é uma métrica e φ_i é um valor médio para esta métrica, com $\varphi_i \in \mathbb{D}_{\mathcal{M}}$. Esse vetor é publicado com cada serviço s_j, juntamente com uma capacidade máxima cap_{s_j}, ou seja, o número máximo de requisições por unidade de tempo para o qual o serviço candidato garante a QoS oferecida.

Portanto, o modelo de representação de serviços baseado em QoS permite que os requisitos de QoS sejam especificados por papéis e por coreografias, enquanto as ofertas de QoS são definidas para os serviços candidatos desempenharem tais papéis. Assim, a oferta de QoS de um serviço representa a qualidade oferecida pelo provedor de serviços, enquanto os atributos de QoS para os papéis ou coreografias representam requisitos de QoS, ou seja, restrições de QoS requisitadas pelos provedores de coreografias.

4.3 Função de utilidade de QoS

Dado um conjunto de serviços candidatos, onde cada serviço possui um conjunto de atributos de QoS, o problema de escolher o melhor serviço candidato é

uma variante do problema de tomada de decisão multicritérios (MCDM) [146]. Em geral, um serviço candidato pode não exceder todos os outros serviços em relação a todos os atributos de QoS. Por exemplo, um serviço pode fornecer o melhor tempo de resposta, mas pode ter alto preço e baixa reputação. Logo, ao realizar uma seleção de serviços que otimize o desempenho geral, deve-se lidar com o *tradeoff* entre todos os atributos de QoS [59].

Uma forma de lidar com esse *tradeoff* é utilizando uma função de utilidade para avaliar a QoS multidimensional de um serviço e classificá-lo em relação aos demais serviços candidatos [5]. Funções de utilidade fornecem uma função objetivo que mapeia cada possível estado de um serviço, representado por um vetor de atributos de QoS, em um valor escalar de ponto flutuante, permitindo comparar e classificar serviços candidatos [128].

Inicialmente, o vetor de atributos de QoS de cada serviço deve ser normalizado para eliminar a diferença de escala entre os vários atributos de QoS dos serviços candidatos. No processo de normalização, cada valor de atributo de QoS de um serviço é convertido em um valor entre 0 e 1, proporcionalmente aos valores mínimos e máximos presentes nos vetores de atributos de QoS dos serviços candidatos [153]. Vale ressaltar que a normalização considera tanto atributos positivos quanto negativos.

A fórmula para normalização dos valores dos atributos de QoS é mostrada nas Equações 4-1 (atributo negativo) e 4-2 (atributo positivo):

$$q'_{h,i,j} = \begin{cases} \frac{qmax_{h,j} - q_{h,i,j}}{qmax_{h,j} - qmin_{h,j}} & , se\ qmax_{h,j} - qmin_{h,j} \neq 0 \\ 1 & , caso\ contrário; \end{cases} \tag{4-1}$$

$$q'_{h,i,j} = \begin{cases} \frac{q_{h,i,j} - qmin_{h,j}}{qmax_{h,j} - qmin_{h,j}} & , se\ qmax_{h,j} - qmin_{h,j} \neq 0 \\ 1 & , caso\ contrário; \end{cases} \tag{4-2}$$

onde $q'_{h,i,j}$ representa o valor normalizado para o j-ésimo atributo de QoS do serviço candidato s_i para o h-ésimo papel. Esse valor é computado utilizando-se o valor corrente do atributo $q_{h,i,j}$, bem como seus valores mínimo e máximo, $qmin_{h,j}$ e $qmax_{h,j}$.

Após a normalização, o vetor de QoS é processado por uma função de utilidade. As funções de utilidade podem basear-se em diferentes estratégias. Caso os atributos de QoS sejam avaliados utilizando-se a estratégia *best-QoS*, os serviços são classificados de acordo com a QoS, onde serviço melhor classificado possui o valor mais alto [5]. Caso os atributos de QoS sejam avaliados utilizando-se a estratégia *best-fit*, é medido a semelhança entre os vetores de QoS dos serviços candidatos e o vetor de QoS requisitado pelo provedor de coreografias de serviços [66].

Como em alguns cenários os provedores de coreografias podem exigir servi-

ços com base na melhor QoS, enquanto em outros casos podem preferir serviços com melhores ajustes entre a oferta e requisição, esta tese propõe uma função de utilidade que engloba ambas as estratégias. A função de utilidade *Util* é definida na Equação 4-3 em termos de funções auxiliares, definidas nas equações 4-4, 4-5, 4-6 e 4-7, que são responsáveis pelo processamento de cada um dos três tipos de atributo de QoS, ou seja, $alvo$, $requerido$ e $otimizado$, observando que a terceira categoria é subdividida em $minimizado$ e $maximizado$.

$$Util(r_h, q_{h,i}) = \begin{array}{l} alvoDistQoS(r'_h, q'_{h,i}) + \sum_{j=p+1}^{s} \rho_j \times requerido(r'_h, q'_{h,i}) + \\ \sum_{j=s+1}^{t} \rho_j \times otimMin(r'_h, q'_{h,i}) + \sum_{j=t+1}^{k} \rho_j \times otimMax(r'_h, q'_{h,i}) \end{array} \quad (4\text{-}3)$$

Aplicando-se a função de utilidade, os serviços candidatos para um papel são ordenados de acordo com os valores dos seus atributos de QoS. A função calcula a utilidade do i-ésimo serviço candidato para o h-ésimo papel, com base em seus respectivos vetores de QoS, $q_{h,i}$ e r_h, onde $q_{h,i}$ representa o vetor de oferta de QoS do i-ésimo serviço candidato para o h-ésimo papel e r_h representa o vetor de atributos do requisito de QoS do h-ésimo papel (r'_h e $q'_{h,i}$ são seus equivalentes normalizados).

Para efeito de simplificação da notação, consideremos que, dos k atributos de um vetor de QoS, os p primeiros atributos são classificados como $alvos$; os próximos $s - p$ atributos são classificados como $requeridos$; os próximos $t - s$ atributos são classificados como $otimizados\ minimizados$; e os $(k - t)$ atributos restantes são classificados como $otimizados\ maximizados$.

A utilidade dos atributos de QoS classificados como $alvo$ é calculada usando a estratégia *best-fit*, empregando-se a função de distância da Equação 4-4 que mede a semelhança entre o requisito de QoS para um papel e os atributos de QoS oferecidos por um serviço candidato:

$$alvoDistQoS(r'_h, q'_{h,i}) =$$

$$\begin{cases} \infty, \ se \ \exists \ j \in \{1, \ldots, p\} \mid & [r'_{h,j} > q'_{h,i,j}] \ (atributo\ positivo) \\ & ou \\ & [r'_{h,j} < q'_{h,i,j}] \ (atributo\ negativo) \\ \sqrt{\sum_{j=1}^{p} \rho_j \times (r'_{h,j} - q'_{h,i,j})^2}, \ caso\ contrário; & \end{cases} \quad (4\text{-}4)$$

onde $q'_{h,i,j}$ é o valor normalizado do j-ésimo atributo de QoS do i-ésimo serviço candidato para o h-ésimo papel, e $r'_{h,j}$ é o valor normalizado do respectivo vetor de atributos do requisito de QoS. Além disso, $q_{min_{h,j}}$ e $q_{max_{h,j}}$ são os valores mínimos e

máximos que podem ser esperados para o atributo entre todos os serviços candidatos, e ρ_j é o peso atribuído a cada atributo.

A utilidade dos atributos de QoS classificados como $requeridos$ é calculada simplesmente atribuindo-se o valor 0, caso o serviço candidato satisfaça o requisito, ou $infinito$, caso contrário, como representado pela Equação 4-5.

$$requerido(r'_h, q'_{h,i,j}) = \begin{cases} 0 & , se \; \forall \; j \in \{p+1, \ldots, s\} \mid r'_{h,j} \equiv q'_{h,i,j} \\ \infty & , \; caso \; contrário. \end{cases} \tag{4-5}$$

onde $q'_{h,i,j}$ é o valor normalizado do j-ésimo atributo de QoS do i-ésimo serviço candidato para o h-ésimo papel, e $r'_{h,j}$ é o valor normalizado do respectivo vetor de atributos do requisito de QoS do papel.

Os atributos de QoS classificados como $otimizado$ utilizam a estratégia *best-QoS* e devem ser avaliados como tendo valores $minimizado$ ou $maximizado$. Para atributos de QoS marcados como minimizados, os valores mais baixos são melhores (Equação 4-6), enquanto que, para atributos de QoS marcados como maximizados, os valores mais altos são melhores (Equação 4-7). Ao usar essas duas equações, como o melhor valor é normalizado para um valor mais alto, o valor final deve ser subtraído de 1, uma vez que adotamos a estratégia de que quanto menor for o valor, melhor será o atributo.

$$otimMin(r'_h, q'_{h,i,j}) = \begin{cases} \infty & , se \; \exists \; j \in \{s+1, \ldots, t\} \mid q'_{h,i,j} > r'_{h,j} \\ 1 - \frac{q_{max_{h,j}} - q'_{h,i,j}}{q_{max_{h,j}} - q_{min_{h,j}}} & , \; caso \; contrário; \end{cases} \tag{4-6}$$

$$otimMax(r'_h, q'_{h,i,j}) = \begin{cases} \infty & , se \; \exists \; j \in \{t+1, \ldots, k\} \mid q'_{h,i,j} < r'_{h,j} \\ 1 - \frac{q'_{h,i,j} - q_{min_{h,j}}}{q_{max_{h,j}} - q_{min_{h,j}}} & , \; caso \; contrário; \end{cases} \tag{4-7}$$

Note que, quanto menor o valor da função de utilidade, melhor será a utilidade de um serviço candidato. Além disso, quando o valor de um atributo de QoS de um serviço não satisfaz os requisitos de QoS, o valor da função de utilidade é definido como infinito.

4.4 Modelo de custo

Na maioria das abordagens encontradas na literatura [153, 5, 155], funções de utilidade são empregadas para selecionar serviços com a melhor QoS possível. Entretanto, selecionar sempre os serviços candidatos com os melhores valores para os atributos de QoS pode levar a ineficiência, uma vez que o custo financeiro dos serviços é tipicamente proporcional à QoS oferecida. Sendo assim, selecionar sempre serviços com a melhor QoS possível pode gerar um desperdício dos recursos associados aos serviços.

Na seleção tradicional de serviços, questões relacionadas à capacidade dos serviços são utilizadas apenas para garantir que a carga sobre o serviço não ultrapasse a capacidade máxima ofertada por ele. Entretanto, a capacidade de requisições que um serviço é capaz de processar, ao mesmo tempo que garante a QoS ofertada, também influencia no custo do serviço, uma vez que quanto maior a capacidade de um serviço mais recursos computacionais são necessários.

Geralmente, o custo de um serviço é tratado como uma métrica adicional de QoS, como ocorre em [153, 6, 84], e se refere ao valor pago pelo provedor de coreografias ao provedor de serviços, seguindo critérios próprios como, por exemplo, licenciamento de uso de software e uso de recursos computacionais. Desta forma, o provedor de coreografias define o orçamento para seleção dos serviços usados em uma coreografia, enquanto que o provedor de serviços define o preço dos serviços. Esse processo não tem como foco minimizar os custos incorridos pelos provedores de coreografias, mas apenas garantir que o custo de cada serviço não ultrapasse o valor requisitado.

Nesta tese o preço de um serviço, definido pelo provedor de serviços, possui um valor fixo dependente da QoS oferecida e da quantidade de requisições por unidade de tempo que o serviço é capaz processar [92]. Essa definição adota aspectos utilizados na seleção de recursos computacionais onde, segundo Eckert *et al.* [42], o preço unitário médio por invocação de um serviço resulta da divisão da taxa fixa pela quantidade invocações do serviço.

O modelo de custo dos serviços, definido para o provedor de coreografias, baseado em [17], é mostrado na Equação 4-8.

$$Custo = \sum_{i=1}^{b} pre_i \times cap_i \tag{4-8}$$

onde b é o número de serviços selecionados, pre_i é o preço do i-ésimo serviço com base nos recursos alocados para atender a QoS oferecida e cap_i é o número de requisições por unidade de tempo contratadas para o serviço.

A função de custo proposta nesta tese é com base no custo das coreografias fornecidas pelos provedores de coreografias. Logo, o custo de uma coreografia é definido analisando o preço dos serviços dado pelos provedores de serviços. O preço de um serviço é baseado no nível de QoS oferecido e na capacidade máxima contratada junto ao provedor de serviços. Em outras palavras, o preço é caracterizado pelo recurso computacional necessário para executar um serviço que atenda aos requisitos de QoS e a carga estimada pelo provedor de coreografias.

Para tratar o problema da economia financeira na seleção dos serviços, esta tese propõe utilizar o modelo de custo em consonância com a função de utilidade. Sendo assim, a seleção de serviços consiste em selecionar os serviços mais adequados para implantar um conjunto de coreografias de forma a satisfazer os requisitos de QoS e minimizar os custos em relação aos recursos utilizados pelos serviços.

Para isso, a função de utilidade é empregada para ranquear os serviços mais adequados para os requisitos de QoS das coreografias. Em seguida, a capacidade desses serviços candidatos são analisadas tendo como base a função de custo. Vale ressaltar que as informações sobre o possível compartilhamento de serviços entre coreografias também influencia a função de custo ao analisar os serviços candidatos. A seleção de serviços proposta nesta tese é formalizada no Capítulo 6.

4.5 Conclusão

Ao lidar com múltiplas coreografias de serviços sensíveis à QoS que devem ser encenadas para satisfazer vários usuários, a seleção eficiente dos serviços se apresenta como o principal fator para satisfação das preferências dos usuários e provedores de coreografias. A seleção de serviços sensível à QoS deve, em primeiro lugar, garantir os requisitos de QoS especificados pelos provedores de coreografias, devendo também otimizar os custos relacionados aos recursos utilizados pelos serviços.

Em virtude disso, selecionar serviços de forma sensível à QoS é um problema de otimização complexo, que se torna ainda mais difícil ao considerarmos diferentes requisitos de QoS sobre um mesmo serviço, bem como seu provável compartilhamento entre múltiplas coreografias.

Inicialmente, o modelo de serviços foi discutido, definindo a representação das coreografias, papéis, tipos de serviços e serviço, e a relação entre eles. Em seguida, apresentamos o modelo de serviços baseado em QoS que permite captar tanto os requisitos de QoS que os provedores de coreografias exigem quanto as propriedades de QoS que os serviços candidatos oferecem. Além disso, foi definida a categorização dos tipos de atributos de QoS considerados na especificação de coreografias. A formalização desses elementos teve como objetivo delimitar o escopo do problema.

Em seguida, foi apresentada uma função de utilidade que provê suporte as diferentes estratégias de avaliação de QoS através da classificação prévia dos atributos de QoS. Tal função é usada para classificar os serviços candidatos referentes a certo papel. Como o objetivo da seleção de serviços é, além de garantir a QoS requisitada por múltiplas coreografias, minimizar adequadamente os custos relacionados a execução dos serviços selecionados, uma função de custo também foi proposta. A função de custo de um serviço contratado é baseada no nível de QoS oferecida e na capacidade máxima contratada junto ao provedor de serviços. Sua definição assume que os valores de QoS e da capacidade dos serviços candidatos existentes são informados junto com a descrição das funcionalidades de cada serviço candidato. A oferta de QoS é informada diretamente pelo provedor de serviços. Já a informação da capacidade de um serviço candidato é obtida a partir de algum mecanismo de *profiling*, que está fora do escopo desta tese.

Entretanto, a função de custo proposta segue um modelo estático. Nesse caso, um provedor disponibiliza um serviço com uma determinada QoS e define o preço para cada requisição do serviço. Em outros casos, provedores podem oferecer serviços com diferentes esquemas de custo, onde muitos fatores podem influenciar o provedor na oferta dos serviços. Esses fatores podem ser a data e a hora do uso do serviço ou propriedades específicas do usuário, como localização. Por outro lado, existem muitos serviços para os quais o custo depende dos detalhes de uso, como a duração do uso do serviço ou a quantidade de dados transferidos. Já alguns provedores de serviços podem dar descontos quando um grupo de serviços é contratado, ou seja, quanto mais serviços de um provedor forem contratados, maior será o desconto. Em tais casos, esses fatores influenciam o custo, tornando dinâmico o esquema de custo.

Para lidar com modelos de custo dinâmicos, Ramacher e Mönch discutem uma abordagem de seleção de serviço que incorpora modelos de custo complexos, incluindo descontos por quantidade e pacotes, bem como cobrança de serviços por assinatura [106]. Já Robson, Rosa e Pires, propuseram uma abordagem para a composição de serviços que provê suporte ao gerenciamento de custo de composições de serviço considerando várias classes de modelos de custo [34]. Um modelo de custo variável, levando considerações diversas políticas de preço, é tratado como trabalho futuro.

No próximo capítulo, apresentamos nossa proposta para representação interna das coreografias de serviços bem como a estratégia para a representação combinada de múltiplas coreografias. Essas representações juntamente com os modelos apresentados neste capítulo, são utilizados para como entrada para a seleção de serviços proposta nesta tese.

CAPÍTULO 5

Representação de múltiplas coreografias de serviços com requisitos de QoS

Este Capítulo introduz a representação interna de coreografias de serviços, que permite a representação combinada de várias coreografias. Essa notação é utilizada para definir a entrada para o mecanismo de seleção de serviços sensível à QoS apresentado no Capítulo 6.

O capítulo está assim organizado. Na Seção 5.1 é proposta uma notação para a representação interna de coreografias de serviços anotadas com requisitos de QoS e carga estimada. Com base na notação proposta, na Seção 5.2, é definida uma estrutura para a representação conjunta de múltiplas coreografias de serviços, juntamente com os requisitos de QoS e carga estimada.

O trabalho deste capítulo gerou a publicação [52].

5.1 Representação interna de coreografias de serviços

Existem na literatura diversas linguagens [110, 38, 9] para modelagem de coreografias de serviços, das quais a maioria constitui variações de BPMN. Isso ocorre em razão de que BPMN2 se tornou o padrão *de facto* para notação alto nível de processos de negócios, fornecendo uma notação de fácil compreensão por todos os usuários envolvidos e uma notação gráfica para modelagem de coreografias de serviços [99].

No entanto, as especificações BPMN2 podem ser complexas e a especificação padrão introduz limitações que um projetista de coreografias deve obedecer, onde o padrão fornece somente uma descrição textual para estas limitações, tornando difícil o seu emprego correto [9]. Além disso, BPMN2 não provê suporte a caracterização de requisitos de QoS e carga impostos sobre os serviços [16].

Em razão disso, esta tese propõe uma notação própria para modelagem de coreografias de serviços com adição de requisitos de QoS e carga sobre os papéis. Esta notação tem como finalidade representar coreografias de serviços de forma não ambígua e permitir que a demanda sobre cada serviço, assim como as dependências

entre os serviços, sejam mais facilmente identificadas e processadas. Outra motivação para adotar essa notação é permitir a conexão dos serviços aos requisitos de QoS especificados sobre as coreografias. Por outro lado, a notação proposta não tem como objetivo servir como linguagem de propósito geral para representação de coreografias, mas sim como representação interna de coreografias para o mecanismo de seleção de serviços. Não obstante, a representação contém os principais elementos comumente adotados em linguagens de definição de coreografia, como BPMN2.

Como a notação proposta é uma representação interna, os provedores de coreografias podem especificar coreografias utilizando BPMN2 ou WS-CDL, por exemplo e, então, tais especificações são convertidas nesta representação interna com a inclusão dos requisitos de QoS e carga associados. Utilizamos essa representação interna para guiar as decisões sobre seleção de serviços, ao passo que a encenação da coreografia, depois que os papéis são substituídos pelos serviços concretos, continua sendo realizada pela linguagem originalmente utilizada.

No contexto desta tese, para permitir a seleção de serviços sensível à QoS, uma representação de coreografia deve respeitar restrições associadas ao descrever uma colaboração entre uma coleção de papéis para atingir um objetivo comum. Cada serviço s_i pode desempenhar um papel p_h em uma coreografia, através da especificação do conjunto de funcionalidades que ele implementa, onde cada funcionalidade é representada por uma operação o. Cada operação tem um conjunto de atributos de QoS, que representam os requisitos de QoS para os serviços candidatos, juntamente com a informação da carga estimada.

Nesta seção, a definição formal da notação proposta é apresentada, juntamente com a transformação de modelos de coreografia descritos em BPMN2 em modelos que usam essa representação. A topologia de coreografias de serviços é representada usando-se grafos de processo, com base na representação proposta por [90], cuja definição proposta nesta tese é apresentada a seguir:

Definição 5.1 ***(Vértices predecessores e sucessores)*** *Seja N o conjunto de vértices e $E \subseteq N \times N$ uma relação binária sobre N que define as arestas. Para cada vértice $n \in N$ definimos o conjunto de vértices predecessores $\bullet n = \{x \in N \mid (x,n) \in E\}$ e o conjunto de vértices sucessores $n\bullet = \{x \in N \mid (n,x) \in E\}$.*

Definição 5.2 ***(Grafo de processo – GP)*** *Um grafo de processo consiste em uma tupla $(b, Z, \mathcal{P}, L, E)$, onde:*

- *b – denota o vértice inicial, de forma que $|b\bullet| = 1$ e $|\bullet b| = 0$, ou seja, o vértice inicial possui apenas um vértice sucessor e nenhum vértice predecessor.*
- *Z – denota o conjunto de vértices finais, de forma que $|Z| \geq 1$ e $\forall z \in Z : |\bullet z| \geq 1$ e $|z\bullet| = 0$, ou seja, cada vértice final tem um ou mais vértices predecessores e*

nenhum vértice sucessor. Múltiplos vértices finais são modelados para representar diferentes estados de finalização (sucesso ou falha).

$\mathcal{P}$ *– denota o conjunto de papéis, de forma que* $\forall p \in \mathcal{P} : |\bullet p| \geq 1 \; e \; |p \bullet| \geq 1$*, ou seja, cada papel tem um ou mais vértices predecessores e um ou mais vértices sucessores.*

L *– denota o conjunto de conectores, onde* L^f *representa o conjunto de conectores fork, usados para separar o fluxo do processo, e* L^j *representa o conjunto de conectores join, usados para agregar partes do fluxo do processo, e para cada conector fork há um conector join equivalente. Os conectores podem ser particionado em conjuntos disjuntos* $L^f = \{AND^f, OR^f, XOR^f\}$ *e* $L^j = \{AND^j, OR^j, XOR^j\}$*, de forma que* $\forall \ell^f \in L^f : (|\bullet \ell^f| = 1 \; e \; |\ell^f \bullet| > 1), \forall \ell^j \in L^j : (|\bullet \ell^j| > 1 \; e \; |\ell^j \bullet| = 1)$*. Para cada* $\ell^f = \{XOR^f, OR^f\} \in L^f$*,* $\ell^f \bullet = \{G_1, ..., G_n\}$*, de forma que uma requisição pode ser encaminhada com uma probabilidade* pr *para um subgrafo* $G_1, ..., G_n$*. Para um* OR^f*, em adição à probabilidade de cada subgrafo, há uma probabilidade* pr *da requisição ser encaminhada simultaneamente para todos os subgrafos. Como exemplo, com dois subgrafos como sucessores para um conector* OR^f*, uma requisição pode ser encaminhada para um dos subgrafos* G_1*,* G_2 *ou simultaneamente para ambos, com probabilidade* pr_1*,* pr_2 *e* pr_{12}*, respectivamente.*

E *– é um conjunto de arestas que definem o fluxo como um grafo direcionado. Cada aresta* $\mathfrak{e} \in E$ *é uma tupla* $(\underrightarrow{\mathfrak{e}}, \overrightarrow{\mathfrak{e}}, \mathfrak{o})$*, onde* $\underrightarrow{\mathfrak{e}} \subseteq (b \cup \mathcal{P} \cup L)$ *é a origem da aresta,* $\overrightarrow{\mathfrak{e}} \subseteq (Z \cup \mathcal{P} \cup L)$ *é o destino da aresta, e* $\mathfrak{o}$ *é a operação sendo requisitada no papel modelado como destino. Se* $\overrightarrow{\mathfrak{e}} \in \{Z \cup L\}$*, então* $\mathfrak{o}$ *é nula.*

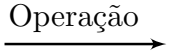
Elemento	**Notação** *GP*	**Elemento**	**Notação** *GP*
Evento inicial		Evento final	
Papel			
Aresta quando $\overrightarrow{e} \subseteq \mathcal{P}$		Aresta quando $\overrightarrow{e} \subseteq (Z \cup L)$	
Conector *fork* de conjunção		Conector *join* de conjunção	
Conector *fork* de disjunção		Conector *join* de disjunção	
Conector *fork* de disjunção mutuamente exclusiva		Conector *join* de disjunção mutuamente exclusiva	

Tabela 5.1: *Representação gráfica da notação do grafo de processo.*

A Tabela 5.1 apresenta uma representação gráfica destes elementos. Na definição de grafo de processo adotada nesta tese, o conjunto de padrões de composição é restrito a um subconjunto dos conectores mais utilizados na modelagem de composições de serviços (sequência, conjunção, disjunção e disjunção mutuamente exclusiva). Essa restrição se justifica pelo fato de que construções mais complexas podem ser obtidas a partir da combinação das construções deste subconjunto, como descrito em [153, 4]. Além disso, foi adicionada informação de probabilidade de encaminhamento de requisições para cada $\ell^f = \{XOR^f, OR^f\} \in L^f$, quantificando o comportamento do processo de negócio.

Como dito anteriormente, diferentes linguagens podem ser utilizadas pelos provedores para especificações de coreografias. Nesta tese é apresentado um modelo de mapeamento entre elementos BPMN2 em componentes de grafo de processo. A Tabela 5.2 apresenta essa tradução, onde $D_1, \ldots, D_n$ são sub-diagramas, $G_1, \ldots, G_n$ são subgrafos, $pr_i, i \in \{1, \ldots, n\}$ é a probabilidade de uma requisição ser encaminhada ao subgrafo G_i, e pr_{12} é a probabilidade da requisição ser encaminhada aos subgrafos G_1 e G_2, simultaneamente. Usando-se as regras de mapeamento propostas é possível implementar um adaptador que gera uma representação na notação PG a partir de modelos em BPMN2. Ao traduzir especificações de coreografias descritas em BPMN2 para a notação de grafo de processo, limitamos a estratégia de mapeamento entre modelos a um subconjunto de elementos de BPMN2, considerando aqueles elementos que são mais comumente utilizados na representação de coreografias.

Um grafo de processo representa uma única coreografia de serviços com um único ponto de inicialização, destacando os papéis executados pelos serviços para alcançarem o objetivo da coreografia. O principal objetivo no uso desta representação é facilitar a identificação de dependências entre os serviços e a análise da QoS e carga para cada serviço. Sendo assim, o uso de grafo de processo é restrito às atividades relacionadas à seleção de serviços, finalidade para o qual os elementos representados são suficientes. A encenação da coreografia ou outras atividades para as quais a notação do grafo de processo não é suficiente, podem ser realizadas usando-se o modelo original em BPMN2 (ou outra linguagem de especificação).

A Figura 5.1 apresenta o GP gerado a partir do modelo em BPMN2 da coreografia de serviços representada na Figura 2.6(b), apresentada no Capítulo 2. A representação em BPMN2 é reproduzida para facilitar a correspondência entre os dois modelos.

Em um GP, cada vértice representa um papel. As arestas representam interações e dependências entre os papéis participantes. Por exemplo, no GP da Figura 5.1(b) o vértice b indica o ponto de início, o vértice z indica o evento final e os demais vértices representam os papéis que compõem a coreografia. A aresta entre *Serviço pedido* e

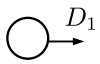
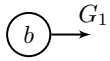
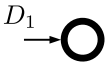
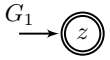
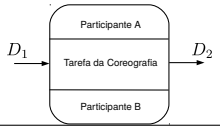
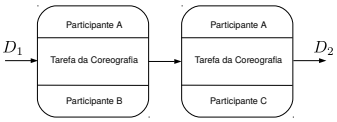
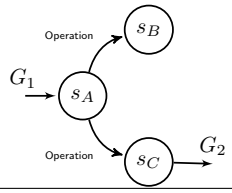
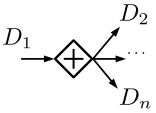
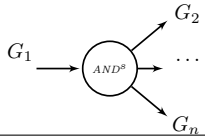
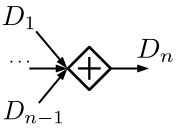
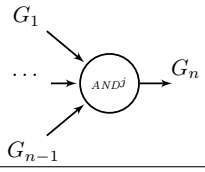
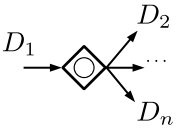
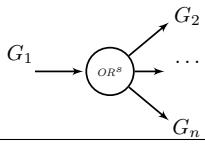
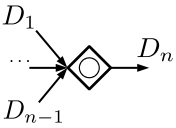
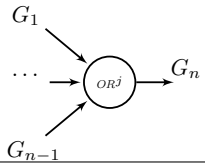
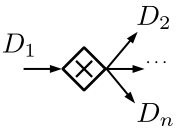
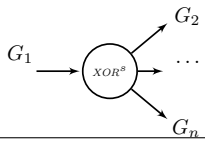
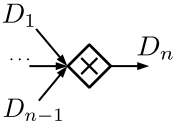
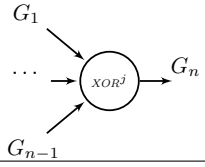
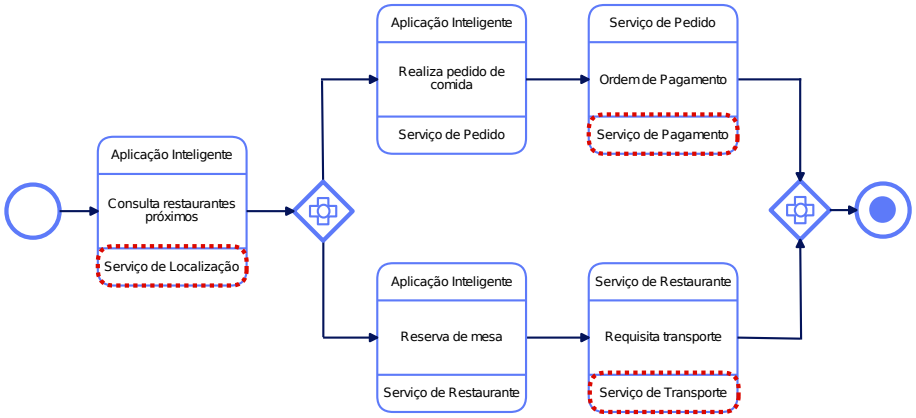
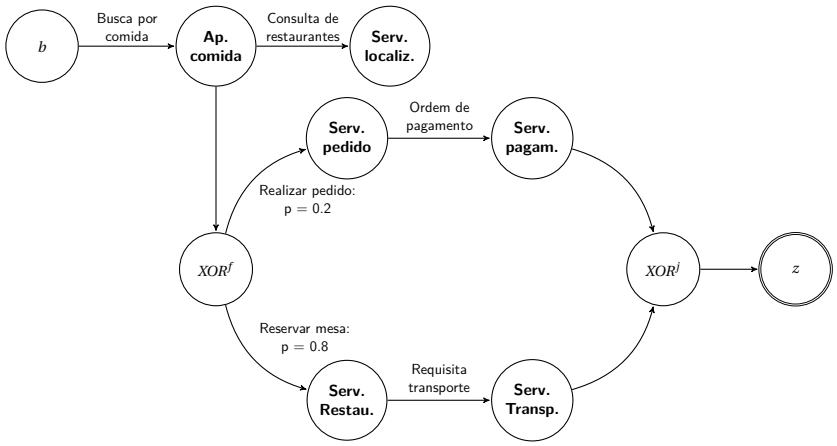
Elemento	Descrição	Notação BPMN	Notação PG
Evento inicial	Evento sem conotação especial que indica o ponto de partida		
Evento final	Evento sem conotação especial que indica o ponto de encerramento		
Tarefa	Representa a troca de mensagens entre dois participantes		
Sequência de tarefas	Representa a sequência da troca de mensagens		
Conector *fork* de conjunção	Divide o fluxo, sendo todos os fluxos de saída acionados simultaneamente		
Conector *join* de conjunção	Combina fluxos paralelos aguardando que todos os fluxos de entrada se completem antes de acionar o fluxo de saída		
Conector *fork* de disjunção	Divide o fluxo, e ativa um ou mais fluxos de saída		
Conector *join* de disjunção	Aguarda que todos os fluxos de entrada ativos se completem antes de acionar o fluxo de saída		
Conector *fork* de disjunção mutuamente exclusiva	Divide o fluxo encaminhando-o para exatamente um fluxo de saída		
Conector *join* de disjunção mutuamente exclusiva	Aguarda que o fluxo de entrada ativo se complete antes de acionar o fluxo de saída		

Tabela 5.2: *Principais elementos em BPMN2 e sua representação para a notação de grafo de processo.*



(a)



(b)

Figura 5.1: *Coreografia para consultar por restaurantes: (a) usando a notação BPMN2. (b) usando a notação GP.*

Serviço pagamento representa a requisição da operação *Ordem de pagamento* do papel *Serviço pagamento* pelo papel *Serviço pedido*.

O provedor ao especificar uma coreografia de serviços, também deve anotar a carga estimada para a coreografia e os requisitos de QoS, que podem ser global ou local. Além disso, o provedor deve especificar para os conectores do tipo XOR^f e OR^f a informação de probabilidade de encaminhamento de requisições. Essa informação é quantificada tendo como base o comportamento do processo de negócio.

Depois de converter cada coreografia em GP, é necessário estimar a carga da coreografia para cada papel e mapear os requisitos de QoS global em requisitos de QoS local (ou seja, por papel), conforme a fase de *análise de coreografias*, representada na Figura 1.3, que discute os passos da abordagem proposta nesta tese.

Na especificação de um grafo de processo, o provedor de coreografias ao definir os vértices representando os conectores de disjunção XOR^f e OR^f anota explicitamente para cada aresta e a probabilidade do fluxo de requisições para cada vértice de destino. No exemplo da Figura 5.1(b), são anotadas as probabilidades para as arestas

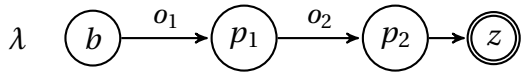
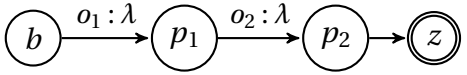
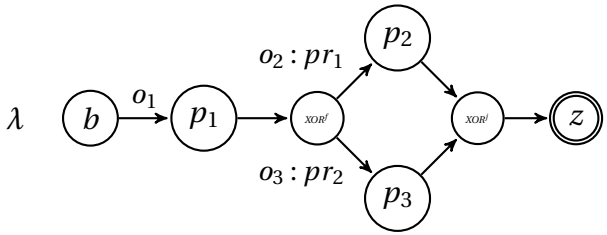
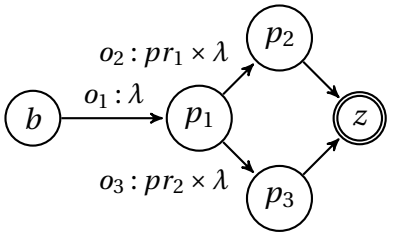
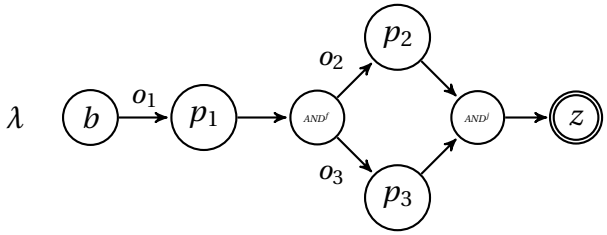
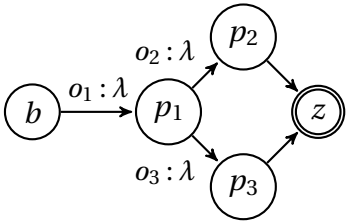
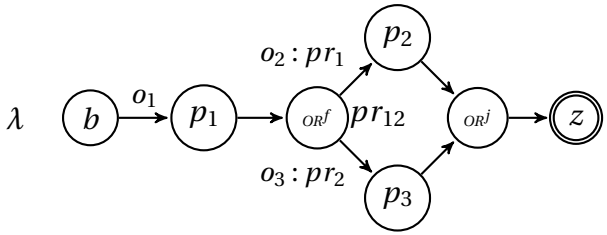
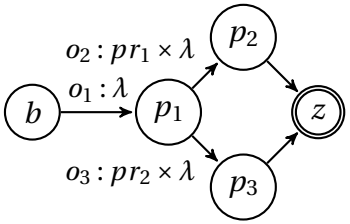
Padrão	**Carga estimada do processo**	**Carga estimada por papel**
Sequência		
Disjunção mutuamente exclusiva		
Conjunção		
Disjunção		

Tabela 5.3: *Quebra da carga estimada para diferentes topologias de um grafo de processo.*

Realizar pedido e *Reservar mesa*, de forma que uma requisição pode ser encaminhada com uma probabilidade *20%* para *Serviço pedido* e com uma probabilidade *80%* para *Serviço restaurante*. Dado a carga estimada sobre uma coreografia e as informações de probabilidade do fluxo de requisições para cada vértice de destino, a carga por papel é estimada. A Tabela 5.3 enumera para cada padrão de composição o método de estimativa de carga para os papéis envolvidos, considerando que λ é a carga de entrada.

Para mapear os requisitos de QoS global em requisitos de QoS local, esta tese segue o modelo proposto por Alrifai et al. [5], onde um requisito de QoS global é decomposto em um conjunto de $\mathcal{N}$ requisitos de QoS local $C_1, C_2, ..., C_\mathcal{N}$, sendo $\mathcal{N}$ o número de papéis em um grafo de processo. Os requisitos de QoS local devem garantir que se os serviços candidatos para cada papel puderem atender tais requisitos, os valores de QoS agregados desses serviços candidatos podem satisfazer também os requisitos de QoS global. Esse processo somente é realizado quando o provedor de coreografias anota, junto à coreografia de serviços, os requisitos de QoS global, uma vez que o provedor pode anotar apenas requisitos de QoS local.

Há duas etapas principais no processo de decomposição da QoS global, determinando os níveis de qualidade e a busca pelo conjunto de requisitos de QoS local para cada conjunto de serviços candidatos, enquanto sua agregação não viola nenhum dos requisitos de QoS global. Na primeira etapa, com o objetivo de encontrar os requisitos de QoS local, decompõem-se o intervalo de cada atributo em um conjunto de valores de qualidade que são empregados como restrições local nas avaliações de serviços candidatos. Esse conjunto de valores de qualidade é chamado de nível de qualidade [4].

Em seguida, cada requisito de QoS global é mapeado em um conjunto de níveis de qualidade, de forma a garantir que requisitos de QoS local sejam relaxados o máximo possível enquanto atende aos requisitos de QoS global. Por fim, os valores de requisito C de QoS e carga λ, por papel, são anotados diretamente no grafo de processo, em uma estrutura chamada de *Grafo de Processo sensível à QoS*.

Definição 5.3 ***(Grafo de Processo sensível à QoS – GPQ)*** *Um Grafo de Processo sensível à QoS consiste em um grafo de processo* $(b, Z, \mathcal{P}, L, E)$ *anotado com a carga esperada* λ *e requisito* C *de QoS para cada operação o definida em uma aresta* $\mathfrak{e} \in E$*, tal que* $\underrightarrow{\mathfrak{e}} \subseteq (b \cup \mathcal{P} \cup L)$ *é a origem da aresta e* $\overrightarrow{\mathfrak{e}} \subseteq (Z \cup \mathcal{P} \cup L)$ *é o destino da aresta, e a nova tupla é reescrita como* $(\underrightarrow{\mathfrak{e}}, \overrightarrow{\mathfrak{e}}, \mathfrak{o}, C, \lambda)$.

A Figura 5.2 apresenta o GPQ gerado a partir do modelo em BPMN2 de uma coreografia de serviços para busca de rota entre dois pontos em uma cidade, e anotado com informações da carga e requisitos de QoS, seguindo a definição 4.2. Assim como em GPs, em um GPQ cada vértice representa um papel e as arestas representam interações e dependências entre papéis. Entretanto, em GPQs as arestas são também

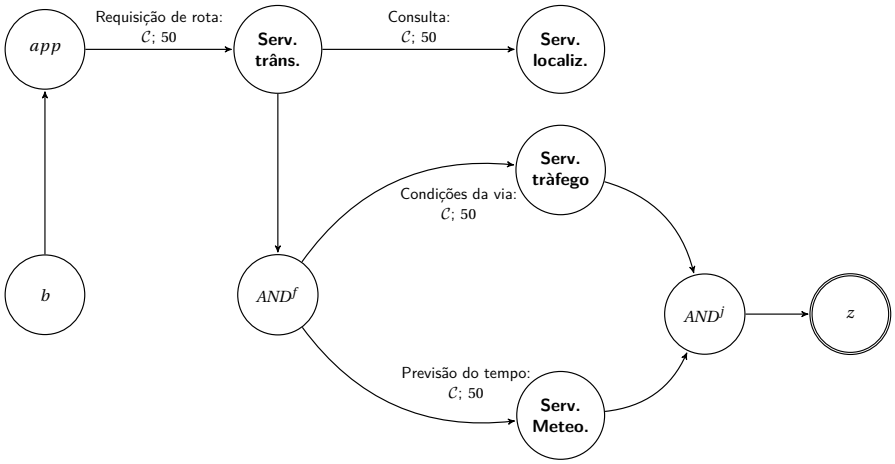


Figura 5.2: *Coreografia para busca de rota usando a notação GPQ.*

anotadas com requisitos de QoS e carga estimada. Por exemplo, no GPQ da Figura 5.2 a aresta entre *Serviço trânsito* e *Serviço localização* representa a requisição da operação *Consulta* do papel *Serviço localização* pelo papel *Serviço trânsito* com um requisito Cde QoS esperado e com uma especificação de carga esperada (representado por 50 requisições por segundo).

5.2 Combinando múltiplas coreografias de serviços

Ao se implementar coreografias de serviços devem-se considerar os requisitos de QoS dos provedores de coreografias como parte do processo de seleção de serviços [54]. Em cenários do mundo real, um serviço pode ser selecionado por várias coreografias para realização de um papel em comum entre elas. O compartilhamento de serviços entre coreografias tem um efeito significativo na QoS geral fornecida, podendo causar a degradação da QoS ofertada, caso o possível compartilhamento de serviços seja negligenciado na seleção de serviços. Nesta tese é conjecturado que não é possível fazer uma seleção de serviços eficiente sem considerar, para cada serviço, todas as coreografias das quais ele participa.

Uma estratégia para seleção de serviços eficiente nesse cenário é levar em consideração que múltiplas coreografias de serviços podem apresentar um conjunto comum de papéis, o que torna necessário lidar com compartilhamento de serviços entre as coreografias na fase de seleção. Essa necessidade torna a satisfação de restrições ainda mais desafiadora, principalmente devido à degradação da QoS de serviços compartilhados e à possível existência de múltiplos requisitos sobre um mesmo serviço. Como exemplo, é possível utilizar uma mesma métrica de QoS para restringir um mesmo serviço que participa de duas ou mais coreografias usando diferentes valores-alvo em cada uma delas.

Por esse motivo, a seleção de serviços deve analisar a carga agregada dos papéis em comum entre várias coreografias que são mapeados para um mesmo serviço candidato. O processo de seleção de serviços deve ser realizado de forma a decidir o serviço mais adequado para cada papel que será usado para implantar cada coreografia, satisfazendo restrições sobre serviços específicos. A possibilidade de expressar, junto com os requisitos de QoS, a carga agregada de papéis em comum favorece um gerenciamento mais preciso em relação à análise da capacidade máxima dos serviços candidatos disponíveis. Como consequência, torna-se possível selecionar de forma mais precisa os serviços necessários à implantação de um grupo de coreografias.

Desta forma, ao invés de realizar a seleção de serviços para cada coreografia individualmente, as coreografias são agregadas em um único modelo antes de realizar o processo de seleção de serviços. Esta estratégia visa permitir uma análise global dos requisitos das múltiplas coreografias, a fim de realizar a seleção de serviços de forma a garantir à QoS requisitada ao mesmo tempo que se busca minimizar os custos relacionados ao uso dos serviços selecionados.

Esta tese apresenta uma abordagem para a seleção de serviços baseada na representação combinada das várias coreografias usando uma estrutura chamada *Grafo de Dependências sensível à QoS*. Esse modelo é formalmente definido a seguir:

Definição 5.4 ***(Grafo de Dependências sensível à QoS - GDQ)*** *Um grafo de dependências sensível à QoS é um grafo direcionado representado por uma tupla* $(b, Z, \mathcal{P}, \mathbb{E})$*, em que:*

- b *– denota o vértice inicial,* $|b\bullet| = 1$ *e* $|\bullet b| = 0$*, ou seja, o vértice inicial possui apenas um vértice sucessor e nenhum vértice predecessor.*
- Z *– denota o conjunto de vértices finais, de forma que* $|Z| \geq 1$ *e* $\forall z \in Z : |\bullet z| \geq 1$ *e* $|z\bullet| = 0$*, ou seja, cada vértice final tem um ou mais vértices predecessores e nenhum vértice sucessor.*
- $\mathcal{P}$ *– denota o conjunto de papéis,* $\forall p \in \mathcal{P} : |\bullet p| \geq 1$ *e* $|p\bullet| \geq 1$*, ou seja, papéis tem um ou mais vértices predecessores e um ou mais vértices sucessores.*
- $\mathbb{E}$ *– é um conjunto de arestas que definem as dependências como um grafo direcionado. Cada aresta* $\mathfrak{e} \in E$ *é uma tupla* $(\underrightarrow{\mathfrak{e}}, \overrightarrow{\mathfrak{e}}, \mathfrak{o}, \mathcal{R})$*, onde* $\underrightarrow{\mathfrak{e}} \in (b \cup \mathcal{P})$ *é a origem da aresta,* $\overrightarrow{\mathfrak{e}} \in (z \cup \mathcal{P})$ *é o destino da aresta,* $\mathfrak{o}$ *é a operação sendo requisitada, e* $\mathcal{R}$ *é um conjunto de requisitos de QoS e carga sobre um mesmo papel p comum entre n coreografias.* $\mathcal{R}$ *possui um conjunto de n requisitos de QoS e carga cujos valores são representados pela tupla* $(C_i; \lambda_i)$ *específicos de cada coreografia, onde* C_i *é um requisito de QoS e* λ_i *é carga estimada para a i-ésima coreografia. Tem-se que* $\overrightarrow{\mathfrak{e}} = z \rightarrow \{\mathfrak{o}$ *é nulo e C é nulo*$\}$.

Nessa estrutura, cada vértice representa um papel que participa de uma ou mais coreografias. Mais especificamente, todos os papéis equivalentes em todos os grafos de processo são combinados em um único vértice do *GDQ*. As arestas representam as dependências entre os papéis. Os requisitos de QoS são anotados nas arestas na forma de requisitos de QoS distintos para cada papel original, a fim de preservar os requisitos de QoS das coreografias individuais. O mesmo se aplica à carga associada aos papéis que foram combinados em um vértice GDQ. Dessa forma, o identificador de cada coreografia é anotado junto com cada requisito de QoS e especificação de carga.

O processo de geração de um grafo de dependências a partir de um conjunto de grafos de processo sensível à QoS é descrito no Algoritmo 7.2 (Capítulo 7). A Figura 5.3 apresenta o grafo de dependências gerado ao combinar as três coreografias apresentadas no Capítulo 2 (Figuras 2.5, 2.6(a) e 2.6(b)). Cada vértice no *GDQ* contém o nome da operação requisitada e o conjunto de requisitos de QoS e carga, preservando os requisitos de QoS e cargas associadas das coreografias originais.

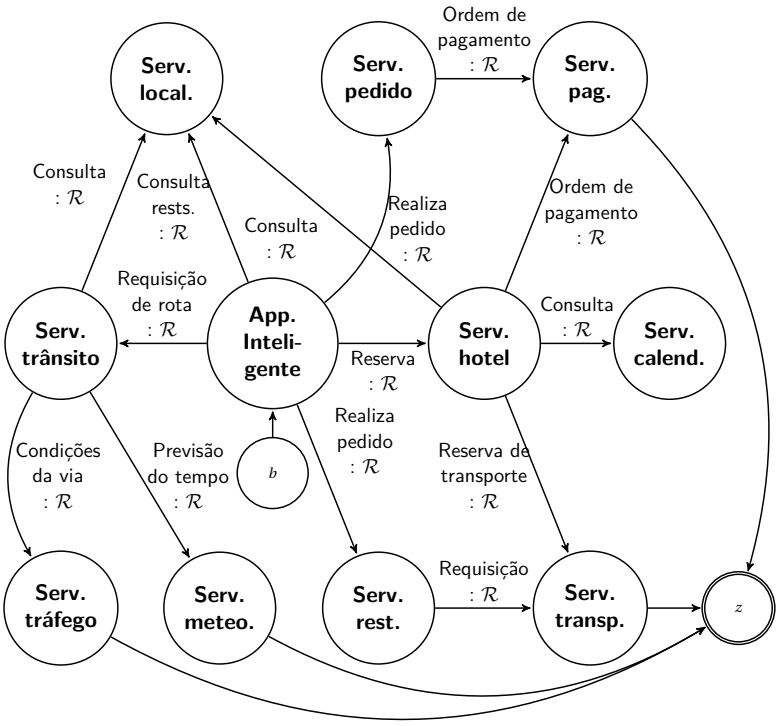


Figura 5.3: *Grafo de dependências sensível à QoS gerado a partir das coreografias (grafos de processo) do cenário da Seção 1.3.*

5.3 Conclusão

As principais dificuldades encontradas na seleção de serviços para múltiplas coreografias estão relacionadas à análise da QoS e da capacidade dos serviços candi-

datos. Isso se dá porque os requisitos de QoS têm que ser analisados face ao compartilhamento de papéis entre diferentes coreografias. Em virtude disso, selecionar serviços de forma sensível à QoS é um problema de otimização complexo, que se torna ainda mais difícil ao considerarmos diferentes requisitos de QoS sobre um mesmo serviço e seu possível compartilhamento entre múltiplas coreografias. Essas dificuldades motivaram a proposta de uma visão combinada de várias coreografias de serviços no momento do processo de seleção de serviços.

Neste capítulo, inicialmente, apresentamos a representação interna para coreografias de serviços. Apesar da existência de várias linguagens para especificação de coreografias de serviços, nenhuma delas permite expressar informações sobre requisitos de QoS e carga. Em virtude disso, propomos uma abordagem para representação de coreografias de serviços anotada com requisitos de QoS e carga, chamada de grafo de processo sensível à QoS. Essa notação pode ser utilizada pelo provedor de coreografias para especificar coreografias de serviços utilizando sua representação gráfica, permitindo anotar diretamente no grafo de processo informações sobre requisitos de QoS e carga.

Ao traduzir modelos de coreografias descritas em BPMN2 para a representação de grafos de processo, limitamos a estratégia de mapeamento entre modelos a um subconjunto de elementos de BPMN2, considerando aqueles elementos que são mais comumente utilizados na representação de coreografias. Além disso, a associação de conectores com valores de probabilidade foram incluídos em nossa representação uma vez que elas são quantificações do comportamento do processo de negócio e clarificam a associação da carga estimada para cada papel.

Dada a carga estimada pelo provedor, para cada grafo de processo é realizada a estimativa da carga para cada papel analisando-se os valores de probabilidade associados aos conectores. Além disso, é realizada a decomposição dos requisitos de QoS global em requisitos de QoS local. O objetivo dessas duas transformações é permitir que a seleção de serviço sensível à QoS e à capacidade possa ser realizada de forma independente para cada papel.

Por fim, tendo como base a notação proposta, apresentamos uma estratégia para a representação conjunta de múltiplas coreografias, em uma representação chamada de grafo de dependências sensível à QoS. A geração de uma representação de um conjunto de coreografias tem como objetivo estabelecer uma visão global dos serviços a serem selecionados e, dessa forma, tornar explícitos os efeitos de seu compartilhamento.

Dessa forma, a representação combinada de coreografias de serviços, proposta neste capítulo, juntamente com o modelo de utilidade e custo, definido no Capítulo 4, formam a base para o processo de seleção de serviços, que consiste em selecio-

nar o conjunto de serviços mais adequados para satisfazer um grupo de coreografias. O próximo capítulo utiliza esses elementos para definir a abordagem proposta nesta tese para seleção de serviços sensível à QoS e à capacidade.

CAPÍTULO 6

Síntese de serviços

A principal contribuição desta tese é uma nova abordagem para seleção de serviços sensível à QoS e à capacidade para implantação eficiente de múltiplas coreografias de serviços. Seguindo a visão geral da nossa abordagem, ilustrada na Figura 1.3, este capítulo descreve a seleção de serviços, que deve escolher, dentre um conjunto de serviços candidatos, os serviços que serão utilizados para desempenhar os papéis em cada coreografia. Este processo de seleção de serviços é chamado de ***síntese de serviços***.

A entrada para a síntese de serviços é um grafo de dependências sensível à QoS, definido no Capítulo 5, assim como um conjunto de serviços candidatos. O objetivo da síntese de serviços é indicar um emparelhamento entre os papéis das coreografias e os serviços candidatos de forma a garantir os requisitos de QoS ao mesmo tempo que busca reduzir a utilização dos recursos que são alocados para os serviços selecionados.

O restante deste capítulo está organizado da seguinte forma. A Seção 6.1 descreve a formalização do problema da síntese de serviços como um problema de emparelhamento, a partir dos conceitos apresentados nos Capítulos 4 e 5. Na Seção 6.2 é discutida uma solução geral para a síntese de serviços com base no emprego do algoritmo húngaro, proposto para encontrar emparelhamento sobre um grafo bipartido. Também são discutidas as limitações desta solução devido aos requisitos adicionais da síntese de serviços. Por fim, na Seção 6.3, são apresentadas as técnicas propostas para realizar a síntese de serviços, bem como a definição do algoritmo húngaro estendido por meio de um conjunto de extensões propostas para cada requisito específico da síntese de serviços.

O trabalho deste capítulo gerou a publicação [78].

6.1 O problema da síntese de serviços

O problema da síntese de serviços sensível à QoS e à capacidade para múltiplas coreografias de serviços (*QoS- and capacity-aware service synthesis for multiple*

service choreography problem – SSMCP) consiste em, dado um conjunto de coreografias de serviços, com um conjunto de papéis em comum, selecionar os serviços necessários para implantar as coreografias, de forma a satisfazer os requisitos de QoS especificados como parte da entrada e minimizar o custo relacionado ao uso dos serviços selecionados.

A definição do problema SSMCP, além do conjunto de serviços candidatos, tem-se um grafo abstrato de dependências com p papéis. Sendo assim, o problema é dividido em p subproblemas, onde cada subproblema trata da seleção de serviços para um papel presente em uma ou mais coreografias, dados os requisitos de QoS e a carga imposta sobre o papel em cada uma das coreografias.

Cada subproblema pode ser descrito da seguinte forma. Considere o conjunto $\{c_1, c_2, \ldots, c_n\}$, representando as coreografias que possuem um dado papel em comum, com seus respectivos requisitos de QoS e cargas λ impostas, e o conjunto de m serviços candidatos $\{s_1, s_2, \ldots, s_m\}$, com suas respectivas ofertas de QoS e capacidades cap. Baseado nesses dois conjuntos, é construída uma matriz de tamanho $n \times m$, onde cada elemento $w(c_i^h, s_j^h)$ representa o custo para a coreografia c_i, dado o respectivo requisito de QoS, selecionar o serviço candidato s_j para o h-ésimo papel.

Sendo assim, o SSMCP consiste em encontrar o emparelhamento de custo mínimo das $c = 1, 2, \ldots, n$ coreografias com respectivos requisitos de QoS, sobre um papel comum entre elas, com os serviços candidatos, de tal modo que os requisitos de QoS das coreografias para o respectivo papel sejam garantidos, o somatório dos custos $w(c_i^h, s_j^h)$ seja minimizado e as restrições de capacidade nos serviços candidatos sejam respeitadas (sendo $\lambda \leq cap$). Vale ressaltar que um requisito de QoS nesta tese, segundo a Definição 4.2, representa uma tupla contendo o identificador da coreografia, o papel sob o qual o requisito de QoS é definido e o vetor de atributos de QoS.

O resultado do SSMCP é representado como $\mathcal{F} = \{f_1, f_2, \ldots, f_p\}$, onde f_h, $1 \leq h \leq p$, é um conjunto de pares $\{(c_1^1, s_1^1), \ldots, (c_n^y, s_m^y)\}$, onde c_i^h representa o i-ésimo requisito de QoS do h-ésimo papel da respectiva coreografia e s_j^h representa o j-ésimo serviço candidato do h-ésimo papel no mapeamento. Cada par desse conjunto representa o mapeamento de uma coreografia, com um dado requisito de QoS para um papel, para um serviço candidato do papel em questão. Vale ressaltar que para cada papel em uma coreografia, um único serviço candidato é selecionado.

O valor de $w(c_i^h, s_j^h)$, que representa o custo da coreografia c_i ao selecionar o serviço candidato s_j para satisfazer o respectivo requisito de QoS do h-ésimo papel, é utilizado para comparar duas alternativas de mapeamento. Esse valor é calculado utilizando a função de utilidade, definida na Equação 4-3, que agrega todas as métricas de QoS. A agregação dos valores de QoS, considerando todas as métricas, deve estar dentro do limite definido pelo valor-alvo para que a função seja satisfeita.

Assim sendo, definimos $\varphi(f_h)$ como o custo composto de todos os serviços selecionados para satisfazer o h-ésimo papel comum entre um conjunto de coreografias. Como consequência, definimos $\varphi(\mathcal{F})$ como o custo composto de todos os serviços selecionados para todas as coreografias que devem ser implantadas. Desta forma, a solução para SSMCP é aquela que satisfaz todas as restrições e apresenta o menor valor de $\varphi(\mathcal{F})$.

Como ilustrado na Figura 6.1, os parâmetros de entrada do SSMCP são as informações sobre os papéis das coreografias combinadas, representadas no grafo de dependências, e o conjunto de serviços candidatos para os respectivos papéis. Logo, o SSMCP consiste em encontrar o melhor emparelhamento dos papéis das coreografias com os serviços candidatos. O problema SSMCP é dividido em p subproblemas, onde cada subproblema trata da seleção de serviços para um papel presente em uma ou mais coreografias. Para isso, para cada papel comum a um conjunto de coreografias, é definido um grafo bipartido representando as respectivas coreografias que possuem o dado papel compartilhado, com seus respectivos requisitos de QoS e carga, e os serviços candidatos para o papel, com suas ofertas de QoS e capacidade. Ao final do processamento, são selecionados os respectivos serviços aptos a desempenharem os papéis em cada coreografia.

Como o problema SSMCP é dividido em p subproblemas, com resoluções independentes, cada subproblema SSMCP pode ser modelado como um problema de programação linear inteira. Considere a variável binária $x_{c_i^h, s_j^h}$ que é igual a 1 se o serviço s_j for selecionado para a coreografia c_i dado o requisito de QoS para o papel h, ou 0, caso contrário. Logo, cada subproblema de SSMCP pode ser mais formalmente expresso como:

$$\text{Minimizar} \quad \varphi(\mathcal{F}) = \sum_{h=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{m} w(c_i^h, s_j^h) \times x_{c_i^h, s_j^h};$$

sujeito a

$$x_{c_i^h, s_j^h} \in \{0, 1\};$$

$$\sum_{i=1}^{n} x_{c_i^h, s_j^h} = 1, \forall h = 1, \ldots, p \; e \; \forall j = 1, \ldots, m;$$

$$\sum_{i=1}^{n} x_{c_i^h, s_j^h} \times \lambda_{h,i} \leq cap_{h,j}, \forall h = 1, \ldots, p \; e \; \forall j = 1, \ldots, m.$$

onde a variável $x_{c_i^h, s_j^h}$ pode ser igual a 0, indicando que o serviço s_j não é selecionado pela coreografia c_i, dado o requisito de QoS da respectiva coreografia para o h-ésimo papel, ou igual a 1, indicando que o serviço s_j é selecionado pela coreografia c_i, dado o requisito de QoS da respectiva coreografia, para desempenhar o h-ésimo papel.

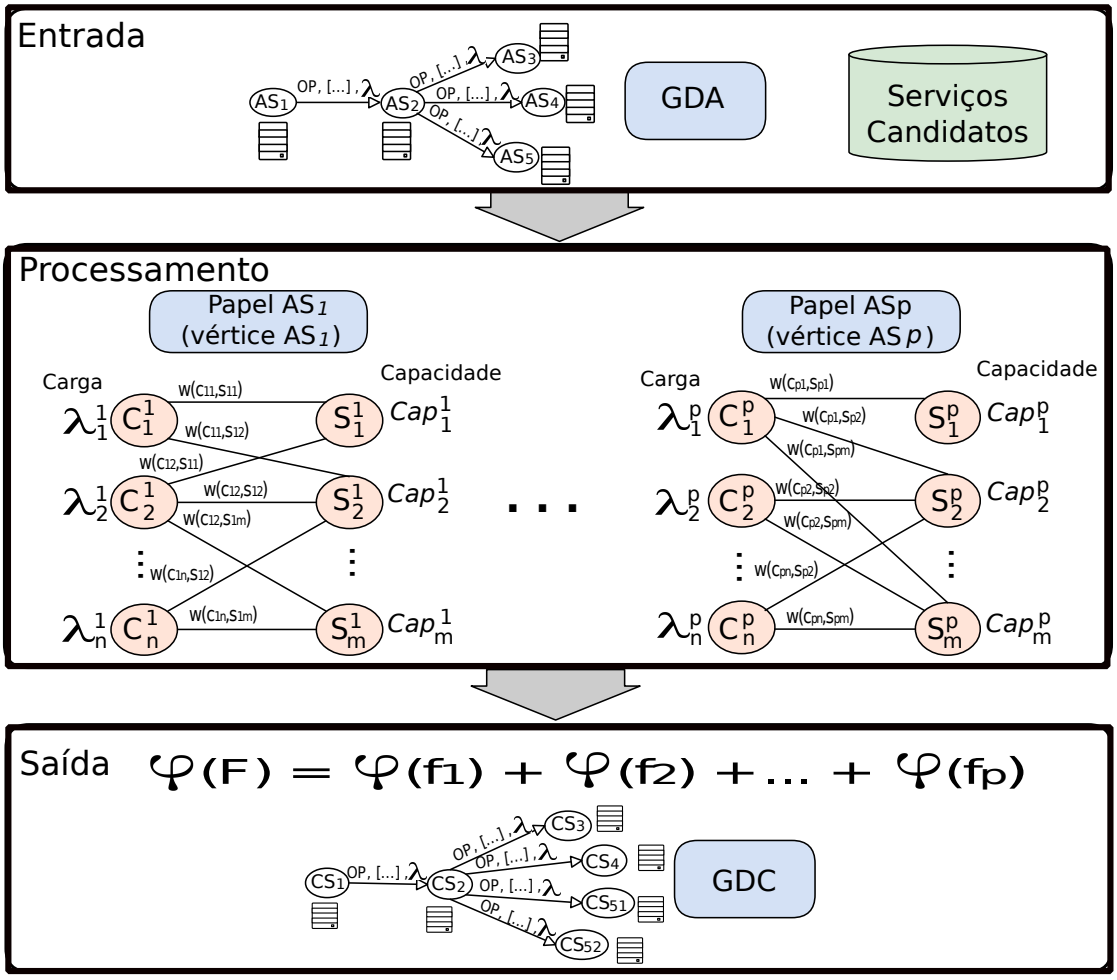


Figura 6.1: *Elementos do problema SSMCP.*

A função objetivo $\varphi(\mathcal{F})$ foi utilizada para minimizar $w(c_i^h, s_j^h)$, tendo como base a função de utilidade definida na Equação 4-3, que busca minimizar a distância entre os requisitos de QoS das coreografias e a oferta de QoS dos serviços selecionados ao aplicar a estratégia *best-fit*. As restrições asseguram a integralidade das variáveis de decisão, onde cada papel com um requisito de QoS de uma coreografia pode ser mapeado para somente um único serviço e que a capacidade máxima de cada serviço candidato selecionado não seja violada.

Para diminuir o custo dos serviços selecionados, além de minimizar o valor da função $\varphi(\mathcal{F})$, também deve-se maximizar a taxa de utilização dos serviços selecionados. A taxa de utilização do serviço s_j, representada como u_j, é calculada com base na capacidade máxima contratada e na real capacidade utilizada, dada a carga estimada da coreografia onde o serviço é utilizado. Sendo assim, busca-se maximizar a taxa de utilização dos serviços aplicando-se a função objetivo U, conforme a seguir:

$$\text{Maximizar} \quad U = \sum_{h=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{h,j} * x_{c_i^h, s_j^h};$$

sujeito a

$$x_{c_i^h, s_j^h} \in \{0,1\};$$
$$\sum_{i=1}^{n} x_{c_i^h, s_j^h} \times \lambda_{h,i} \leq cap_{h,j}, \forall h = 1,\ldots,p \; e \; \forall j = 1,\ldots,m.$$

onde u_j é taxa de utilização do serviço j. Essa função objetivo busca maximizar a taxa de utilização dos serviços selecionados, onde a restrição assegura a integralidade das variáveis de decisão e garantia da capacidade dos serviços selecionados (sendo $\lambda \leq cap$).

Diante da definição do problema SSMCP, existe diferentes possibilidades de soluções. Uma possível solução seria o emprego da uma solução exata baseada no emprego de programação linear, como adotado por Zeng *et al.* [153] e Ardagna e Pernici [8]. Como o uso de programação linear é ineficiente quando o tamanho da entrada do problema cresce, esse tipo de solução não é adequado quando se lida com múltiplas coreografias.

Além disso, é possível descrever o SSMCP como um problema da mochila multidimensional, como Yu *at al.* [148] e Peng e Changsong [105]. Uma outra solução é transformar o SSMCP em um problema de emparelhamento. Ambas soluções requerem o uso de heurísticas para obter uma solução aproximada devido a quantidade de variáveis incluídas na definição do problema. Entretanto, transformar o SSMCP no problema de emparelhamento de custo mínimo é mais natural ao lidar com requisitos sobre a minimização da diferença entre a QoS requisitada e a QoS ofertada, ao mesmo tempo que é buscado a maximização da taxa de utilização. Já ao transformar o SSMCP no problema da mochila multidimensional um conjunto maior de variáveis deveria ser criado, tornando a solução ainda mais complexa.

Sendo assim, o SSMCP é transformado no problema de emparelhamento de custo mínimo. A seguir é apresentada uma solução geral para o problema SSMCP, com base no emprego do algoritmo húngaro [72].

6.2 Solução geral da síntese de serviços

A abordagem proposta nesta tese para a síntese de serviços considera o problema de seleção de serviços como um problema de emparelhamento [39]. Como alguns papéis com diferentes requisitos de QoS podem ser comuns a mais de uma core-

ografia, o processo de seleção de serviços deve assegurar que os diferentes valores alvo de QoS requisitados sejam garantidos pelos serviços selecionados, mesmo havendo compartilhamento. Para isso, a seleção de serviços deve levar em consideração a relação da carga agregada sobre o papel e a capacidade máxima de cada serviço candidato.

Como discutido na Seção 6.1, uma das entradas da síntese de serviços é um Grafo de Dependências Abstrato (GDA). A abordagem de síntese de serviços trata a seleção de serviços para cada vértice do GDA de forma independente. Como cada vértice do GDA representa um único papel, que é comum a um conjunto de coreografias, para cada vértice do GDA existe um conjunto de requisitos de QoS e carga estimada, provenientes de cada coreografia da qual o papel faz parte. Sendo assim, para cada vértice de um GDA, deve-se escolher um ou mais serviços candidatos adequados para o conjunto de coreografias de serviços, com respectivos requisitos de QoS, que possuem um dado papel em comum. Para determinar o melhor serviço candidato para cada papel de uma coreografia, é utilizada uma função de utilidade que mede quão bem um requisito de QoS (sobre um determinado papel) é satisfeito por um serviço candidato.

Nesta tese, o problema da SSMCP pode ser resolvido utilizando uma solução que busca um emparelhamento sobre um grafo bipartido. Neste contexto, um grafo bipartido pode ser definido como se segue: dado um grafo $G = (V, E)$, onde $V = C \cup S (C \cap S = \emptyset)$ e $E \subseteq C \times S$, sendo $w(e)$ o custo da aresta $e(c_i, s_j) \in E$, onde $c_i \in C$ e $s_j \in S$. O elemento C representa as coreografias que possuem um papel compartilhado, com seus respectivos requisitos de QoS e carga. S representa os serviços candidatos, com suas respectivas ofertas de QoS e capacidade. Os custos das arestas representam a aplicação da função de utilidade da Equação 4-3. A Figura 6.2 mostra um exemplo desse tipo de grafo.

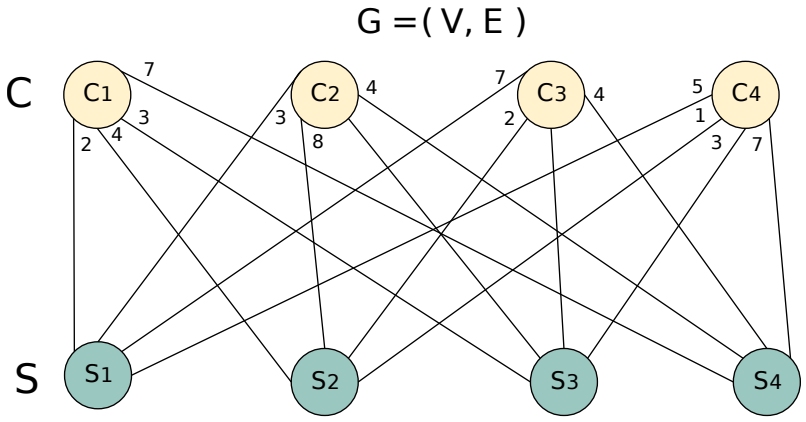


Figura 6.2: *Grafo bipartido $G = (V, E)$ com as bipartições C e S, onde C representa as coreografias, com respectivos requisitos de QoS, que possuem um dado papel em comum e S os serviços candidatos. Para cada aresta é anotado o custo para um serviço s_j desempenhar o papel na coreografia c_i.*

De acordo com a Figura 6.2, a bipartição C representa os requisitos de QoS sobre um mesmo papel compartilhado por quatro coreografias. A bipartição S representa quatro serviços candidatos para desempenhar o respectivo papel. Para cada aresta $e(c_i, s_j)$ é anotado um custo. Por exemplo, o custo para o serviço s_1 desempenhar o papel na coreografia com requisito de QoS c_1 é 2.

Um emparelhamento M é um subconjunto de arestas tal que nenhum par de arestas em M tem um vértice comum [39]. Num grafo $G = (V, E)$ com um emparelhamento $M \subseteq E$, um vértice v que faz parte do emparelhamento é dito saturado se $e \in M$ é incidente em v, caso contrário, o vértice é não saturado ou exposto. O objetivo é encontrar um emparelhamento M de custo mínimo de G.

A Figura 6.3 mostra um emparelhamento M de custo mínimo para o exemplo da Figura 6.2. Nesta figura as arestas de cor laranja representam o conjunto de arestas no emparelhamento M.

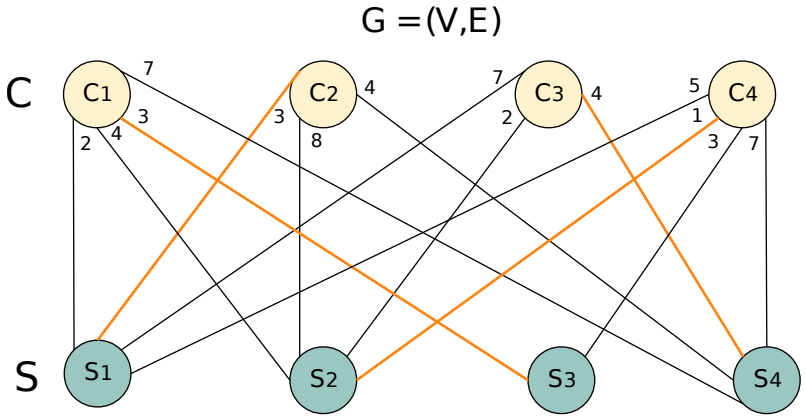


Figura 6.3: *Emparelhamento M de custo mínimo em G, onde as arestas de cor laranja representam o conjunto de arestas no emparelhamento M.*

Para encontrar um emparelhamento de custo mínimo sobre um grafo bipartido é utilizado o *algoritmo húngaro* [72], que encontra um emparelhamento que cobre todos os vértices de C ou então mostra que um emparelhamento não existe [39]. O algoritmo húngaro é um dos algoritmos mais eficientes para encontrar emparelhamentos em grafos bipartidos.

O Algoritmo 6.1 mostra o esboço do algoritmo húngaro que busca um emparelhamento de custo mínimo sobre um grafo G [93]. O algoritmo húngaro procura por caminhos *m-aumentantes* para obter um emparelhamento M que cobre todos os vértices de C. Um caminho *m-aumentante* é um caminho *m-alternante* cujos vértices de origem e término são não saturados. Um caminho *m-alternante* em G é um caminho cujas arestas estão alternadamente em M e fora de M ($E \backslash M$) [39].

Algoritmo 6.1: Algoritmo húngaro.

Entrada: Um grafo bipartido $G = (C, S, E)$ (onde $|C| = n$, $|S| = m$ e $n \leq m$) e uma matriz $n \times m$ com arestas E com custos w

Saída: Um emparelhamento completo M.

1. Executar inicialização:
   (a) Construir um emparelhamento vazio, $M_0 = \emptyset$;
   (b) Atribuir valores viáveis às variáveis α_i e β_j da seguinte forma:

$$\begin{aligned} \forall c_i \in C, \quad \alpha_i &= 0 \\ \forall s_i \in \mathcal{S}, \quad \beta_j &= min_i(w_{i,j}) \end{aligned}$$

2. Definir cada vértice exposto de C como a raiz de um caminho *m-alternante*.
3. Construir caminhos *m-alternantes* enraizados nos vértices expostos no subgrafo da igualdade em busca de caminhos *m-aumentantes*. Atribuir os índices i dos vértices c_i encontrados no caminho *m-alternante* ao conjunto I^*, e os índices j dos vértices s_j encontrados no caminho *m-alternante* ao conjunto J^*. Se um caminho m-aumentante for encontrado, executar o passo (5). Caso contrário, executar o passo (4).
4. Modificar as variáveis α e β (conforme abaixo) para adicionar novas arestas ao subgrafo de igualdade. Em seguida, executar o passo (3).

$$\theta = \frac{1}{2} \times min\{w_{i,j} - \alpha_i - \beta_j : i \in I^*, j \notin J^*\}$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i + \theta & i \in I^* \\ \alpha_i - \theta & i \notin I^* \end{cases}$$

$$\beta_j \leftarrow \begin{cases} \beta_j + \theta & j \in J^* \\ \beta_j - \theta & j \notin J^* \end{cases}$$

5. Aumentar o emparelhamento atual, invertendo as arestas emparelhadas e as arestas não emparelhadas ao longo do caminho *m-aumentante* selecionado, definindo um novo emparelhamento na etapa k, ou seja, M_k.
6. Se M_k contém n arestas, então um emparelhamento M que cobre C foi encontrado e o algoritmo é finalizado. Caso contrário, volte para o passo 2. Caso no enésimo passo $| M | < n$, então, não há um emparelhamento que cobre C e o algoritmo é finalizado sem solução.

Inicialmente, passo 1, o algoritmo húngaro atribui a variável α_i para cada vértice c_i e a variável β_j para cada vértice de s_j. Ele explora o fato de que a minimização

do problema de emparelhamento é viável quando $\alpha_i + \beta_j \leq w(c_i, s_j)$. As arestas em um grafo bipartido são chamadas de admissíveis quando elas fazem parte de uma rotulação de vértices viável, ou seja: $\alpha_i + \beta_j = w(c_i, s_j)$. Defini-se o subgrafo da igualdade (denotado por $G=$) como o subgrafo de G gerado pelas arestas admissíveis.

O algoritmo húngaro trabalha buscando por caminhos *m-aumentantes* (passo 3) para obter um emparelhamento perfeito de C [39]. Se um caminho *m-aumentante* for encontrado, o conjunto atual de M será aumentado ao inverter as arestas emparelhadas e arestas não emparelhadas ao longo desse caminho (passo 5). Como há mais arestas não emparelhadas do que arestas emparelhadas, essa inversão aumenta o conjunto M por um.

Se um caminho *m-aumentante* não for encontrado, as variáveis α_i e β_j serão ajustadas para aumentar o número de arestas no subgrafo da igualdade, tornando-as admissíveis e a busca continua (passo 4).

6.2.1 Considerações da solução geral para a síntese de serviços

O problema SSMCP possui alguns requisitos adicionais que a versão original do algoritmo húngaro não é capaz de tratar. Isso ocorre porque, como discutido no Capítulo 5, cada vértice do *GDA* contém um conjunto de requisitos de QoS, juntamente com as cargas estimadas das diferentes coreografias para o respectivo papel, sendo que cada elemento desse conjunto é associado a uma coreografia específica da qual o papel faz parte. Além disso, cada serviço candidato possui uma capacidade máxima para a qual é capaz de garantir a QoS ofertada.

Como consequência, essas informações sobre carga e capacidade devem ser adicionadas ao grafo, sendo que o processo de busca por emparelhamento deve assegurar que a capacidade máxima de cada serviço candidato selecionado não seja violada ($\lambda_{s_{h,i}} \leq cap_{s_{h,j}}$). A Figura 6.4 mostra a adição de informações de carga e capacidade ao grafo G da Figura 6.3. Por exemplo, c_1 possui uma carga estimada de 4 e o serviço candidato s_3 possui uma capacidade máxima de 20 requisições por unidade de tempo, e o custo de selecionar o s_3 para c_1 é 3. Vale ressaltar que quando um serviço candidato não é capaz de atender um requisito de QoS de uma coreografia, a aresta entre seus vértices possui custo $w(e) = \infty$.

Além disso, como todas as coreografias, com os respectivos requisitos de QoS sobre um papel compartilhado, em um dado vértice do *GDA*, competem pelo mesmo conjunto de serviços candidatos, é possível que um mesmo serviço candidato seja selecionado para mais de uma coreografia. Entretanto, dada a carga das coreografias que compartilham um dado serviço, o limite de capacidade de processamento do serviço candidato selecionado deve ser considerado: $\Lambda_{s_j} \leq cap_{s_j}$, onde Λ_{s_j} é a soma

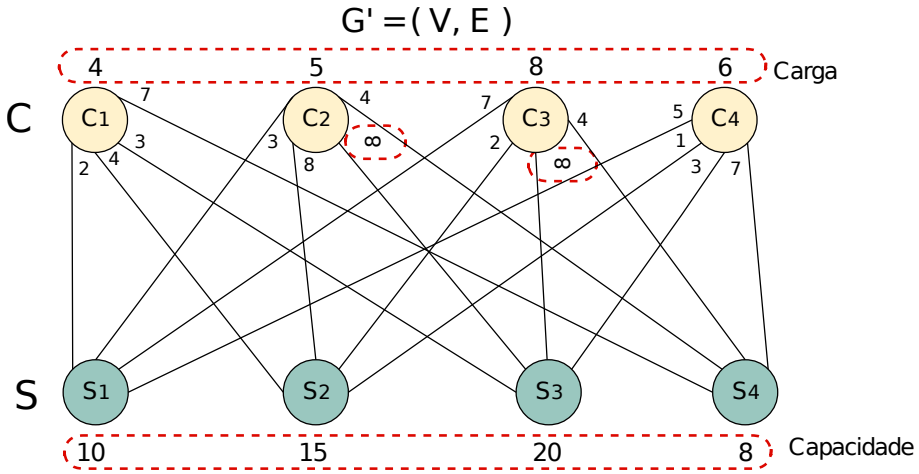


Figura 6.4: *Grafo bipartido $G' = (V, E)$ com as bipartições C e S. Foram adicionadas informações de carga em cada vértice de C e informações de capacidade em cada vértice de S.*

das cargas das coreografias atendidas pelo serviço s_j e cap_{s_j} é a capacidade máxima de processamento do serviço s_j.

Em uma solução para o problema SSMCP, um emparelhamento deve conter todos os vértices de C (requisitos de QoS das coreografias), mas o mesmo não é necessário para os vértices de S. Além disso, mais de um vértice de C pode incidir em um mesmo vértice s_j, desde que a capacidade de s_j seja respeitada. A Figura 6.5 mostra um emparelhamento M' de custo mínimo no grafo G'. Por exemplo, na Figura 6.5 o emparelhamento M' saturou todos os vértices de C.

Entretanto, mesmo que a capacidade máxima de cada serviço em M' (s_1, s_2 *e* s_3) seja respeitada, os vértices c_3 e c_4 incidem em um mesmo vértice s_2, violando a definição de emparelhamento entre grafos bipartidos, ou seja, cada vértice de C pode ser atribuído a apenas um vértice de S e vice-versa.

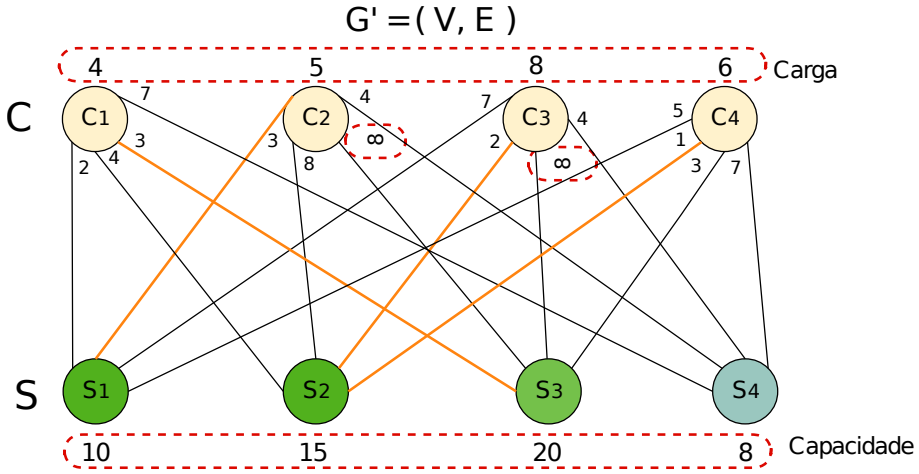


Figura 6.5: *Emparelhamento M' de custo mínimo no grafo G'.*

Portanto, a versão original do algoritmo húngaro não é capaz de lidar com todos os requisitos no espaço do problema de emparelhamento tratado nesta tese. Para isso, o algoritmo deve considerar três requisitos adicionais:

1. Maximizar a taxa de utilização dos serviços selecionados. Nesse caso, a solução deve buscar aproximar a carga agregada das coreografias da capacidade dos serviços selecionados, de forma a diminuir a capacidade residual não utilizada.
2. Permitir que um mesmo vértice de S seja incidido por arestas distintas ligadas a mais de um vértice de C. No problema de emparelhamento original, todo vértice de C incide em no máximo um vértice de S, enquanto no problema SSMCP, há situações em que um mesmo serviço candidato pode ser selecionado por mais de uma coreografia.
3. Evitar que a carga sobre os serviços selecionados ultrapasse a sua capacidade máxima.

6.3 Solução proposta com uso de extensões

Como os requisitos adicionais do problema SSMCP não são tratados na versão original do problema de emparelhamento sobre grafos bipartidos, esta tese adota uma solução para emparelhamento de grafos bipartidos com uma versão estendida do algoritmo húngaro para tratar os requisitos adicionais discutidos na seção anterior, através da proposta de três extensões, sendo uma para cada requisito adicional. Inicialmente, na subseção 6.3.1, é apresentada uma visão geral da solução proposta para o problema SSMCP e em seguida as extensões são apresentadas.

Vale ressaltar que a definição do problema de emparelhamento bem como sua solução segue a notação especificada por Mills-Tettey *et al.* [93].

6.3.1 Visão geral

A Figura 6.6 resume as etapas da estratégia de síntese de serviços proposta. Para cada etapa, são apresentadas as transformações necessárias para aplicação da seleção de serviços. A síntese de serviços recebe como entrada um grafo de dependências e um conjunto de serviços candidatos, realiza a síntese de serviços e gera como saída um grafo de dependências concreto, contendo os serviços selecionados para cada coreografia.

Na primeira etapa, para cada vértice do grafo de dependências, é realizada uma busca pelos serviços candidatos que implementam o papel compartilhado por um conjunto de coreografias. Em seguida, é iniciado o processo de criação do grafo bipartido, com uma bipartição representando os requisitos de QoS de um conjunto

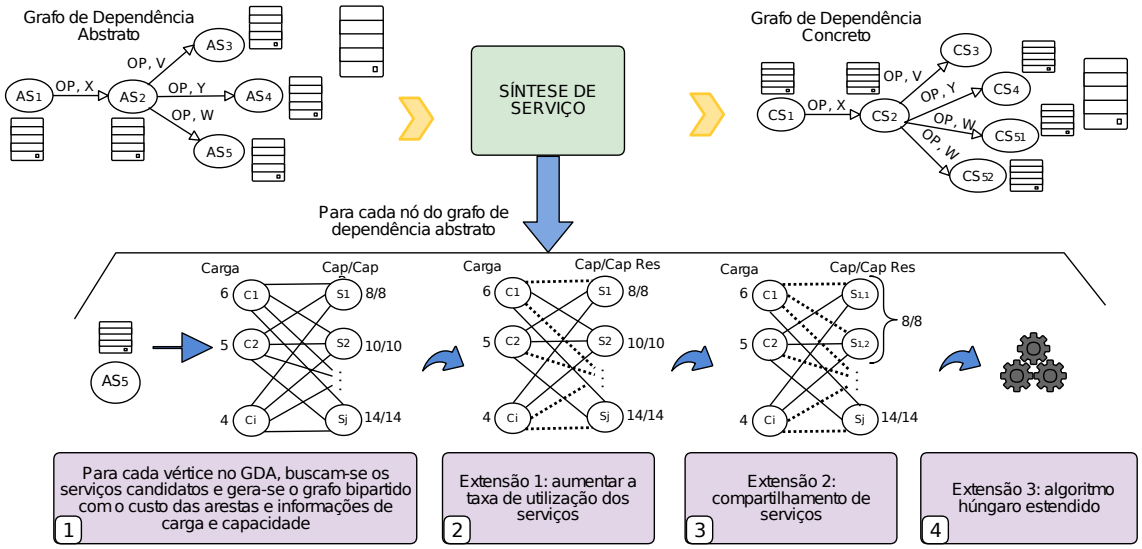


Figura 6.6: *Etapas da síntese de serviços para cada vértice no GDA.*

de coreografias sobre o papel comum entre elas e a outra bipartição representando os serviços candidatos.

Ainda nessa etapa é calculado o custo das arestas, utilizando a função de utilidade, de forma a permitir selecionar serviços que atendam os requisitos de QoS, sem levar ao desperdício de recursos. Além disso, são adicionadas no grafo informações sobre carga para os vértices que representam os requisitos de QoS das coreografias para o papel compartilhado e informações sobre capacidade e capacidade residual para os vértices que representam os serviços candidatos. A saída é um grafo bipartido $G = (C, S, E)$ com informações adicionais sobre carga e capacidade.

Na etapa seguinte, é aplicada a primeira *extensão*, que tem como objetivo incrementar a taxa de utilização dos serviços selecionados. A saída desta etapa é um novo grafo $G' = (C, S', E')$. A próxima etapa da síntese de serviços consiste em permitir o compartilhamento de serviços, que possibilita que um mesmo vértice $s_j \in S$ seja emparelhado com mais de um vértice de C. Esse cálculo é realizado analisando-se a relação entre a carga de cada coreografia que contém o papel compartilhado e capacidade dos serviços candidatos.

Após a geração do novo grafo, a última etapa da síntese de serviços consiste em aplicar o algoritmo de emparelhamento estendido que analisa informações capacidade durante o processo de emparelhamento. A saída dessa etapa e, consequentemente da síntese de serviços, é o grafo de dependências onde os papéis são substituídos por serviços concretos.

Diante da visão geral da solução proposta com uso das extensões para o problema SSMCP, a seguir as três extensões são descritas em detalhes.

6.3.2 Maximização da taxa de utilização dos serviços

Como discutido anteriormente, um dos objetivos da abordagem proposta nesta tese é maximizar a taxa de utilização dos serviços selecionados de forma a reduzir o custo relacionado ao uso dos recursos que são alocados pelos serviços. Ao utilizar a estratégia de avaliação de QoS *best-fit*, em comparação à estratégia *best-QoS*, (discutido no Capítulo 3) conseguimos reduzir o desperdício de recursos ao selecionar serviços de acordo com a distância entre a QoS requisitada e a QoS ofertada, considerando todas as coreografias ao mesmo tempo.

Entretanto, s estratégia *best-fit* tende a diminuir o compartilhamento dos serviços selecionados entre as coreografias, aumentando o número de serviços candidatos selecionados. Um outro efeito é o aumento global da capacidade residual (i.e., não utilizada), portanto reduzindo a taxa de utilização dos serviços. Sendo assim, esta subseção apresenta uma heurística para estender o algoritmo húngaro, que busca maximizar a taxa de utilização dos serviços selecionados, ou seja, minimizar a cap_{res} dos serviços selecionados, ao mesmo tempo em que se aplica a estratégia de avaliação de QoS *best-fit*.

Ao analisar a taxa de utilização de cada serviço candidato, o objetivo deve ser: dado um serviço s_j selecionado, quanto mais próximo Λ_{s_j} for de cap_{s_j}, melhor é a taxa de utilização do serviço s_j. A taxa de utilização u de um serviço candidato s_j é calculada de acordo com a Equação 6-1:

$$u = \left(\frac{cap_{s_j} - cap_{res_{s_j}}}{cap_{s_j}} \right); \tag{6-1}$$

Já a taxa de utilização de um grupo de serviços candidatos selecionados U é calculada utilizando a média geométrica[1], de acordo com a Equação 6-2:

$$U = \sqrt[b]{\prod_{j=1}^{b} \left(\frac{cap_{s_j} - cap_{res_{s_j}}}{cap_{s_j}} \right)}; \tag{6-2}$$

onde b é a quantidade de serviços candidatos selecionados, cap_{s_j} é a capacidade máxima do j-ésimo serviço candidato selecionado, enquanto $cap_{res_{s_j}}$ é sua capacidade residual.

A Figura 6.7 mostra um emparelhamento M sobre o grafo bipartido $G = (V, E)$. A coreografia c_1, com seu respectivo requisito de QoS para o papel compartilhado, incide em três serviços diferentes: s_1, s_2 e s_3. Neste caso, a opção é emparelhar c_1

[1] Média geométrica é apropriada para comparar resultados de métricas computacionais quando se compara diferentes itens e cada um desses itens possui propriedades com diferentes escalas numéricas, como é o caso dos serviços candidatos [45].

com o serviço que possui o menor custo (s_1), independentemente dos demais serviços candidatos. Diante dessa estratégia, de acordo com o emparelhamento M, a taxa de utilização de todos os serviços selecionados, ao aplicar a Equação 6-2, é de *55%*. Esse valor é impactado, principalmente, pela baixa taxa de utilização dos serviços s_1, com taxa de utilização de *44%*, e s_2, com apenas *40%*, tornando-os subutilizados.

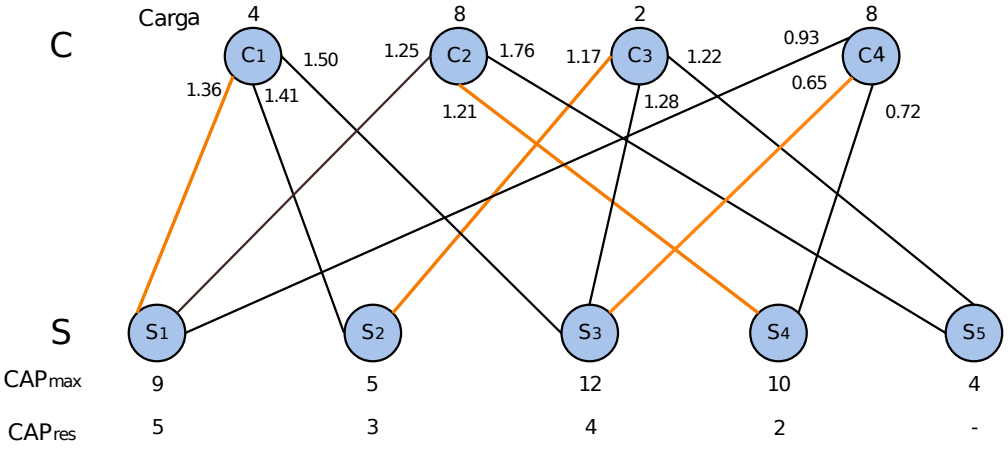


Figura 6.7: *Emparelhamento M sobre $G = (V, E)$. As arestas na cor laranja fazem parte do emparelhamento M.*

Na estratégia de avaliação de QoS *best-fit*, para cada coreografia c_i, com respectivo requisito de QoS para um papel compartilhado, existe uma alta probabilidade de apenas um único serviço candidato s_j possuir o melhor custo. Se o serviço s_j não puder ser selecionado para c_i, a estratégia seleciona o segundo serviço com melhor custo, e assim por diante. Vale lembrar que o serviço candidato com melhor custo possui o menor desperdício de recursos em relação à QoS, uma vez que o custo das arestas representa a menor distância entre QoS requisitada por uma coreografia e a QoS ofertada por um serviço candidato.

Ainda de acordo com a Figura 6.7, o custo da aresta $e(c_1, s_2)$ é muito próximo do custo da aresta $e(c_1, s_1)$. Nesse caso, arestas incidentes em um vértice c_i com custo próximo do menor custo tendem a satisfazer o requisito de QoS com um valor aceitável. Nessa perspectiva, o objetivo da extensão discutida nesta subseção é justamente aumentar o número de serviços candidatos selecionáveis de forma que várias coreografias com requisitos de QoS próximos para um mesmo papel compartilhem um mesmo subconjunto de serviços candidatos, consequentemente aumentando o compartilhamento de serviços entre as coreografias.

Ao elevar o número de serviços compartilháveis, aumenta-se a chance de que um mesmo serviço candidato seja selecionado por várias coreografias e, assim, incrementa-se a taxa de utilização dos serviços selecionados. Para isso, foi definido um *Coeficiente de Aproximação* θ, que representa o custo máximo aceitável de uma aresta $e(c_a, S)$ em relação à aresta de custo mínimo $e_{min}(c_a, s_b)$. Nesse sentido, as arestas de

um mesmo vértice c_i, que incidem em diferentes vértices de S, passam a ter seus custos tratados como iguais, desde que esses custos estejam dentro de um intervalo aceitável definido em função de θ.

O valor do *Coeficiente de Aproximação* deve ser definido de acordo com o intervalo dos valores da função de utilidade, que representam os custos das arestas no problema de emparelhamento. Sendo assim, o valor de θ define o limite aceitável dos custos das arestas em relação ao menor custo. Em outras palavras, dado o menor custo das arestas incidentes em c_i, é definido um intervalo entre esse menor custo e um custo máximo aceitável, de acordo com o valor de θ, isto é, em termos de uma porcentagem acima do menor custo.

Dado o intervalo definido por $e_{min}(c_a, s_b)$ e θ, todas as arestas incidentes no vértice c_a que têm custo dentro do intervalo são tratadas como se tivessem o mesmo custo. Logo, todas essas arestas fazem parte do processo de busca por *caminhos m-aumentantes*. Como dito anteriormente, na versão original do algoritmo de busca por *caminhos m-aumentantes* a partir de um vértice c_a, apenas a aresta com menor custo é selecionada para fazer parte do *caminho m-aumentante* corrente. Caso haja mais de uma aresta admissível[2], é selecionada de forma aleatória uma dentre as admissíveis.

O objetivo do uso de θ é aumentar o número de arestas admissíveis a partir de c_i e, por consequência, aumentar o número de possibilidades de seleção. Entretanto, somente aumentar o número arestas admissíveis não garante o aumento da taxa de utilização dos serviços candidatos selecionados. Logo, é preciso definir uma estratégia que selecione serviços de forma a maximizar a taxa de utilização.

Uma vez que um dado vértice não saturado s_b pode ser alcançável[3] por vários vértices de C, então uma solução é buscar por vários vértices de C que emparelhem com os vértices virtuais de s_b, de forma a minimizar a sua capacidade residual. A estratégia adotada é baseada na heurística *Best-Fit* (BF), adotada na resolução do *problema de bin-packing* [87]. Essa estratégia ordena em ordem crescente os vértices de S alcançáveis por um dado vértice c_i, utilizando o valor da capacidade residual, de forma que quanto menor for a capacidade residual, maior a chance de um vértice s_j ser selecionado.

A heurística *BF*, ao analisar a carga de um vértice c_i, seleciona o vértice s_b com menor capacidade residual, buscando outro vértice s_j caso a capacidade residual não suporte a carga estimada para c_i. A ideia é selecionar um serviço para um papel de forma que o serviço selecionado tenha o mínimo de capacidade residual. Essa estratégia é amplamente adotada na alocação eficiente de recursos para ambientes

[2] Uma aresta é dita ser admissível se ela pode fazer parte de um caminho *m-alternante* a partir de um vértice exposto c_a [93].

[3] Um vértice v é dito alcançável a partir de um vértice u se existe um caminho de u para v [121].

de nuvem, onde cada servidor é tratado como um *bin* e cada máquina virtual é um item a ser empacotado em um *bin* [117, 15, 137]. Na abordagem proposta nesta tese, cada serviço candidato s_j com determinada capacidade cap_{s_j} é tratado como um *bin* e cada coreografia c_i, com respectivo requisito de QoS de um papel com uma carga estimada λ_{c_i}, é um item a ser empacotado.

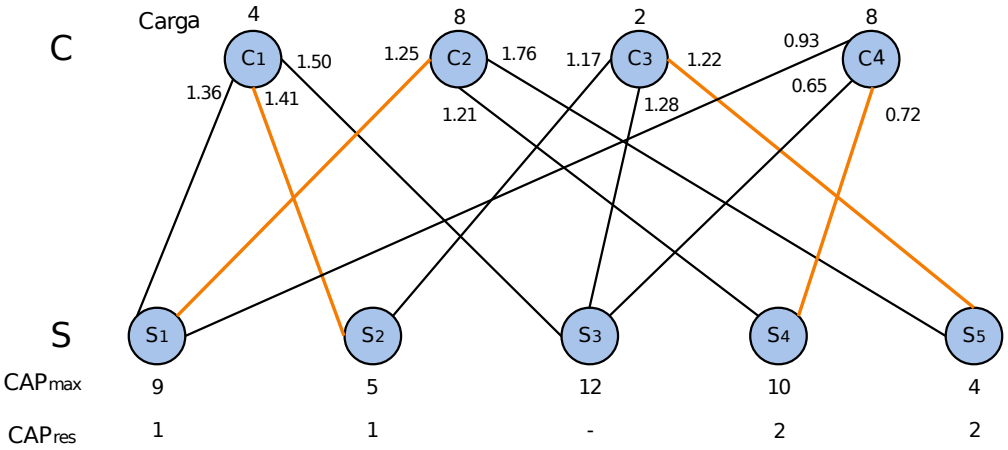


Figura 6.8: *Emparelhamento M' sobre $G' = (C, S, E)$. As arestas na cor laranja fazem parte do emparelhamento M'. As caixas abaixo das informações de capacidade dos serviços candidatos, mostram a taxa de utilização de cada serviço.*

A Figura 6.8 mostra um novo emparelhamento M' usando a heurística proposta. Na figura, o vértice c_1 pode alcançar os vértices virtuais de s_1, s_2 e s_3. Entretanto, ao iniciar o processo de busca por *caminhos m-aumentantes* para o vértice c_1, os serviços alcançáveis são ranqueados pela capacidade residual, sendo que $cap_{res_{s_2}} < cap_{res_{s_1}} < cap_{res_{s_4}}$. Portanto, a aresta $e(c_1, s_2)$ é selecionada em detrimento as demais arestas, mesmo que o custo real da $e(c_1, s_1) < e(c_1, s_2)$. Isso ocorre porque ao calcular os custos das arestas e igualar os custos através do emprego do *coeficiente de aproximação*, a principal variável para tomada de decisão passa a ser a capacidade residual, ou seja, o objetivo é minimizar a capacidade residual para maximizar a taxa de utilização dos serviços selecionados.

Ao comparar o emparelhamento M da Figura 6.7 com o emparelhamento M' da Figura 6.8, nota-se que ao se empregar a heurística proposta, a taxa de utilização dos serviços selecionados é aumentada. De acordo com a Figura 6.8, dado o emparelhamento M', a taxa de utilização de todos os serviços candidatos selecionados, ao aplicar a Equação 6-2, é de *72%*. Além disso, o serviço s_3 que possui uma capacidade alta não foi selecionado, ficando à disposição para futuras requisições.

6.3.2.1 Considerações sobre o coeficiente de aproximação

Intuitivamente, podemos inferir que, quanto maior o *coeficiente de aproximação* θ, maior o número de arestas admissíveis em G e, em decorrência, há um aumento na probabilidade de incrementar o subconjunto de vértices alcançáveis a partir de um vértice c_a. Por consequência, há um aumento no compartilhamento de serviços entre as coreografias.

Para tentar identificar um valor adequado de θ foi realizado um experimento para seleção de serviços para um conjunto de coreografias variando o valor de θ. Nesse experimento, foram selecionados serviços para implantar um grupo de 100 coreografias geradas sinteticamente, considerando que elas possuem 10 papéis cada e que em média 25% dos papéis são compartilhados pelas coreografias. Para cada papel existe 250 serviços candidatos disponíveis. Em relação à oferta de QoS dos serviços candidatos, foi utilizado um gerador de oferta de QoS sintético. No experimento foi analisada a diferença entre a QoS requisitada pelas coreografias para cada papel e a QoS ofertada pelos serviços selecionados. Além disso, também foi analisada a taxa de utilização dos serviços selecionados, objetivo do uso da heurística proposta.

A Tabela 6.1 mostra valores para a diferença entre a QoS requisitada e a QoS ofertada, e para a taxa de utilização dos serviços selecionados com variação do θ. Vale ressaltar que esses dois parâmetros impactam diferentemente no custo de utilização dos recursos associados aos serviços selecionados. Quanto menor o valor da diferença entre a QoS requisitada e a QoS ofertada, menor também será o desperdício, pois significa que o serviço selecionado oferece uma QoS bem próxima da QoS requisitada. Caso contrário, quanto maior a diferença, maior o desperdício, uma vez que o serviço selecionado oferece uma QoS bem superior da QoS requisitada, levando ao superdimensionamento. De forma semelhante, quanto maior a taxa de utilização de um serviço selecionado, menor é o desperdício, uma vez que o recurso associado ao serviço não é subutilizado.

De acordo com a Tabela 6.1, ao incrementar o valor de θ, ocorre o aumento do valor da diferença de QoS, ou seja, a diferença entre a QoS requisitada e a QoS ofertada. Esse aumento ocorre devido à seleção de serviços com ofertas de QoS mais distantes da QoS requisitada, o que eleva o desperdício de recursos em relação à QoS, uma vez que serviços com oferta de QoS alta são selecionados para coreografias com requisitos de QoS que não demandem tanto.

Já quando se analisa a taxa de utilização, ao incrementar o valor de θ, aumenta-se a taxa de utilização dos serviços, diminuindo o desperdício de recursos alocados pelos serviços selecionados. Analisando-se os dados, ao se comparar os valores de θ igual a 0% e igual a 140%, a taxa de utilização aumentou em 28%. Os resultados mostram que quando θ atinge um certo valor, 160% com os dados da Tabela 6.1, não

Tabela 6.1: *Valores para a diferença entre a QoS requisitada e a QoS ofertada e para a taxa de utilização dos serviços selecionados com variação do coeficiente de aproximação.*

Coeficiente de aproximação	Diferença entre a QoS requisitada e a QoS ofertada	Taxa de utilização dos serviços
0%	31,2%	50,2%
20%	31,2%	52,1%
40%	31,7%	54,5%
60%	32,5%	56,9%
80%	33,2%	59,9%
100%	34,3%	60,3%
120%	36,3%	63,3%
140%	37,4%	64,6%
160%	39,8%	67,8%
180%	42,2%	68,2%
200%	44,3%	68,5%
220%	45,9%	68,5%

há mais ganho em relação à taxa de utilização, e há uma piora significativa na diferença entre a QoS requisitada e a QoS ofertada.

Dados os valores gerados como requisitos de QoS e carga para as coreografias e valores de oferta de QoS e capacidade para os serviços candidatos, essa análise permite selecionar para θ o valor 120%. Esse valor de θ mostra que a taxa de utilização dos serviços selecionados para as 100 coreografias aumentou 26%, se comparado à não utilização do θ, embora a diferença entre a QoS requisitada e a QoS ofertada tenha piorado em 15%. No entanto, esses valores de θ analisados são dependentes dos valores dos requisitos de QoS e carga das coreografias, bem como das ofertas de QoS e capacidades dos serviços candidatos. Podemos inferir que não existe um valor de θ ideal para todos os cenários, onde o valor mais adequado para θ depende do conjunto de ofertas de QoS, bem como dos valores dos requisitos de QoS.

6.3.3 Compartilhamento de serviços

O primeiro requisito imposto pelo problema SSMCP envolve a possibilidade que um mesmo vértice de S seja incidido por mais de um vértice de C, uma vez que, na versão original do problema de emparelhamento, um vértice c só pode ser atribuído

a um único vértice s e vice-versa. Isto significa que um serviço pode ser selecionado para mais de uma coreografia ao mesmo tempo, desde que sua capacidade não seja excedida. Sendo assim, ajustes são necessários no algoritmo de emparelhamento para resolver o problema SSMCP, conforme descrito a seguir.

Seja um grafo bipartido $G = (V, E)$, onde $V = C \cup S$ e $C \cap S = \emptyset$. C representa as coreografias que possuem um papel compartilhado, com seus respectivos requisitos de QoS e carga. S representa os serviços candidatos, com suas respectivas ofertas de QoS e capacidade. E representa as arestas anotadas com custos, onde o custo de cada aresta representa o valor da função de utilidade se o serviço candidato s_j é selecionado para uma coreografia com requisito de QoS c_i.

Ao aplicar essa extensão que permite o compartilhamento de serviços, o problema pode ser visto como um problema de emparelhamento de custo mínimo sobre o novo grafo bipartido $G' = (V', E')$. Ao se criar G', um serviço candidato s_j pode ser selecionado por mais de uma coreografia utilizando a solução original do problema de emparelhamento. Para garantir o requisito de compartilhamento de serviços, um vértice s_j, representando um serviço candidato com uma capacidade máxima cap e sobre o qual incidem ϵ_j arestas, cada qual com uma carga estimada λ, deve ser mapeado para um subconjunto de vértices, chamado de vértices virtuais de s_j [63], $s_{j,1}, s_{j,2}, ..., s_{j,\nu}$, onde ν é número de vértices virtuais do serviço candidato s_j. Cada vértice virtual $s_{j,\alpha}$ terá o mesmo valor de utilidade de s_j. Além disso, cada vértice virtual $s_{j,\alpha}$ tem o mesmo valor de capacidade do vértice real s_j, ou seja, esse valor é compartilhado por todos vértices virtuais de s_j.

Para uma aresta $e(c_i, s_j)$, que liga o vértice de um serviço candidato s_j ao vértice c_i no grafo G, existirão ν arestas ligando c_i a cada vértice virtual de s_j ($s_{j,1}$, $s_{j,2}$, ..., $s_{j,\nu}$). A Figura 6.9 mostra o mapeamento entre vértices de serviços candidatos para vértices virtuais em um novo grafo G'. Neste exemplo, para o vértice s_1 são definidos dois vértices virtuais $s_{1,1}$ e $s_{1,2}$ e para o vértice s_2 são definidos três vértices virtuais $s_{2,1}$, $s_{2,2}$ e $s_{2,3}$. Vale ressaltar que as arestas incidentes em $s_{1,1}$ e $s_{1,2}$ provenientes de um mesmo vértice de C, por exemplo, c_1, possuem o mesmo custo. Além disso, esses vértices virtuais também compartilham o mesmo valor de capacidade.

A quantidade de vértices virtuais para um serviço candidato s_j deve ser o mínimo entre o número de arestas incidentes em s_j, e a razão entre a capacidade do serviço candidato e a menor carga dentre as coreografias em C cujas arestas incidem em s_j. Esse valor pode ser calculado de acordo com a Equação 6-3.

$$\nu_j = \left\lfloor \min(\epsilon_j, cap_{s_j} \ / \min_{\forall c_i \in C}(\lambda_{c_i})) \right\rfloor ; \tag{6-3}$$

onde ν_j representa o número de vértices virtuais do serviço s_j, ϵ_j é o número de arestas incidentes no serviço s_j, cap_{s_j} é a capacidade do serviço s_j e λ_{c_i} é a carga que a i-ésima

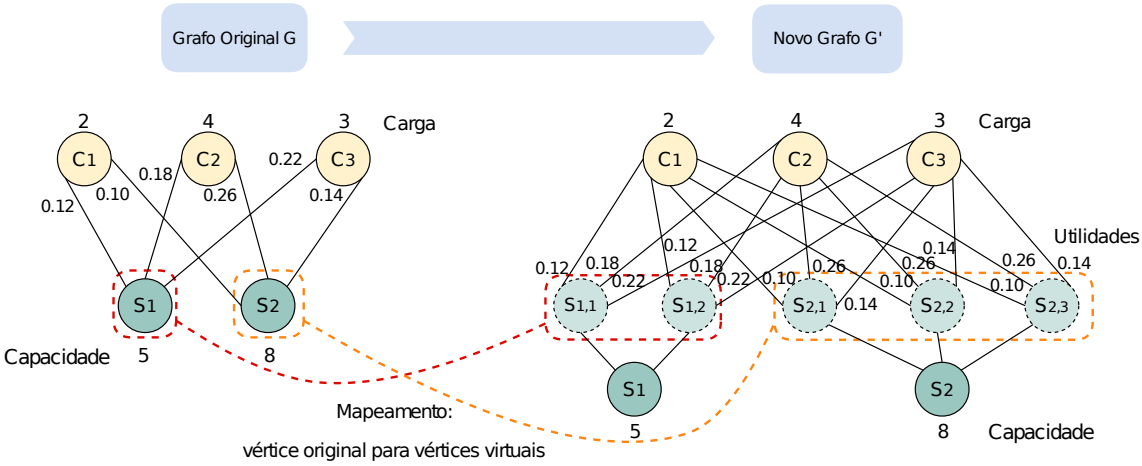


Figura 6.9: *Mapeamento entre vértices de serviços candidatos para vértices virtuais.*

coreografia impõe sobre o serviço s_j.

A Figura 6.10 mostra um emparelhamento M de custo mínimo em G (esquerda) e um emparelhamento M' de custo mínimo no novo grafo G' (direita), onde as arestas em laranja fazem parte do emparelhamento. No grafo original G não é possível emparelhar todos os vértices de C. Por exemplo, o vértice c_2 não foi emparelhado.

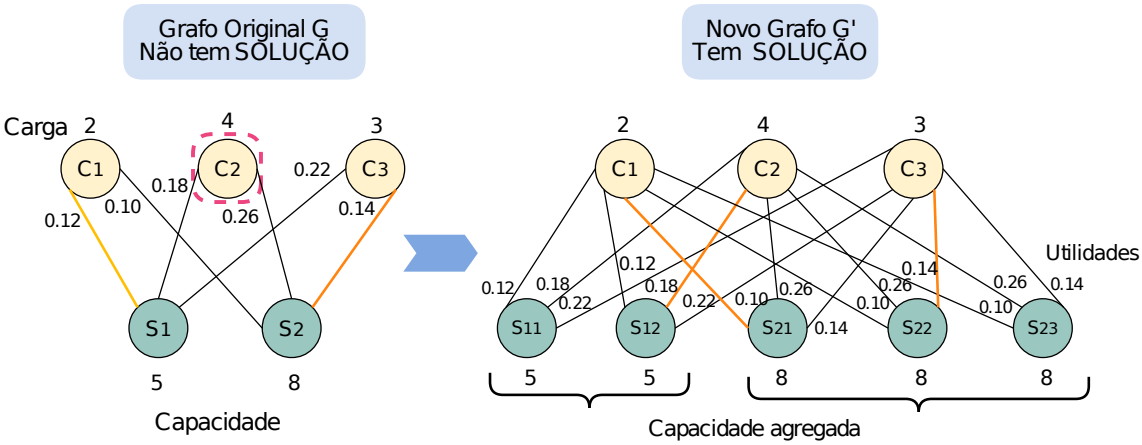


Figura 6.10: *Emparelhamento M' de custo mínimo em G' capaz de saturar todos os vértices de C.*

Ao aplicar a extensão proposta nesta subseção e executar o algoritmo de emparelhamento em G', todos os vértices de C são emparelhados por algum vértice de S, utilizando o conceito de vértices virtuais. Neste exemplo, o vértice c_2 foi emparelhado com o vértice $s_{1,2}$, enquanto o vértice c_1 é emparelhado com o vértice virtual $s_{2,1}$ e o vértice c_3 é emparelhado com o vértice virtual $s_{2,2}$. Como os vértices c_1 e c_3 são emparelhados com o mesmo vértice s_2, por meio dos vértices virtuais $s_{2,1}$ e $s_{2,2}$, o vértice s_2 é incidido por mais de um vértice de C, ou seja, o serviço s_2 é compartilhado por duas coreografias.

Um serviço s_j somente pode ser selecionado para uma ou mais coreografias se sua capacidade cap_j for suficiente para atender a carga agregada dos vértices de C que incidem em s_j. Por exemplo, na Figura 6.10, o serviço s_2, com capacidade 8, é compartilhado por duas coreografias com uma carga agregada com valor 5. Nesse caso, a carga agregada é menor do que a capacidade ofertada pelo serviço. Entretanto, nem sempre um serviço pode ser compartilhado. Por exemplo, no grafo da Figura 6.10 o serviço s_1, com capacidade 5, não pode ser compartilhado pelas coreografias c_2 e c_3 que possuem uma carga agregada com valor 7.

6.3.4 Prevenção de sobrecarga

O segundo requisito do problema da síntese de serviços que não é tratado pelo algoritmo húngaro referi-se à não permissão de sobrecarga sobre serviços selecionados. Inicialmente essa seção descreve um exemplo onde a prevenção de sobrecarga deve ser tratada, discutindo-se as limitações do algoritmo de emparelhamento original em lidar com essa limitação. Em seguida, a essa extensão é propriamente apresentada.

A capacidade máxima de um serviço é dada pelo número máximo de requisições por unidade de tempo que o serviço pode atender, ao mesmo tempo em que é capaz de garantir a QoS ofertada. A carga, por sua vez, refere-se ao número de requisições por unidade de tempo estimado para um papel de uma coreografia. Entretanto, como um serviço candidato pode ser selecionado para mais de uma coreografia, deve-se considerar a carga agregada Λ das requisições de QoS das coreografias para cada serviço candidato.

Um serviço s_j somente pode ser selecionado para uma ou mais coreografias se sua capacidade cap_{s_j} for suficiente para atender a carga agregada (conforme discutido na Seção 6.2), ou seja, atender a Equação 6-4:

$$cap_{s_j} \geq \Lambda_{s_j} \tag{6-4}$$

Para tratar essa restrição é necessário analisar as informações de carga para cada vértice de C e informações de capacidade para cada vértice de S em um grafo G. Essas informações são utilizadas no processo de busca por emparelhamento no grafo da igualdade $G =$, ou seja, na procura por um *caminho m-aumentante*. Entretanto, antes de discutir como essas informações são utilizadas, é necessário definir como ocorre a atualização das informações de capacidade dos serviços selecionados.

Como no grafo G existe um conjunto de vértices virtuais, cada vértice virtual de s_j $(s_{j,1}, s_{j,2}, ..., s_{j,v})$ tem o mesmo valor de capacidade que s_j. Em um emparelhamento M de G, dado um vértice s_a, pode-se obter um vértice virtual $s_{a,1}$ emparelhado com um vértice c_b, enquanto outro vértice virtual de s_a, por exemplo, $s_{a,2}$ é não sa-

turado. No caso do vértice virtual $s_{a,1}$ ser saturado, a carga proveniente do vértice c_b deve ser subtraída da capacidade remanescente do vértice virtual $s_{a,1}$. Como ambos os vértices virtuais compartilham a mesma capacidade do vértice original s_a, o valor corrente da capacidade de cada vértice virtual deve ser atualizado, independentemente de ser saturado ou não.

Para isso, incluímos um campo de informação adicional para guardar, além da capacidade máxima, a capacidade residual de cada serviço candidato s_j. Esse campo de capacidade residual é incluído junto à cada serviço de S. A capacidade residual, simbolizada por cap_{res}, representa a porção da capacidade total que ainda pode ser utilizada por outras coreografias sem afetar a oferta de QoS contratada.

A capacidade residual cap_{res_s} de cada serviço candidato, que também se aplica aos vértices virtuais, é calculada de acordo com a Equação 6-5, onde Λ_{s_j}, valor da carga agregada sobre um serviço candidato s_j, é calculado por meio do somatório das cargas de cada vértice c_i que incide no vértice s_j.

$$cap_{res_{s_j}} = cap_{s_j} - \Lambda_{s_j}, \ \ onde \ \ \Lambda_{s_j} = \sum_{i=1}^{\epsilon_j} \lambda_{c_i}, \ \ \forall e(c_i, s_j) \in M \tag{6-5}$$

A Figura 6.11 mostra um emparelhamento M (arestas de linha contínua) com informações da capacidade máxima e da capacidade residual de cada vértice de S. Os vértices de cor cinza, vértices expostos, representam vértices que ainda não fazem parte do emparelhamento M. As arestas $e(c_1, s_{1,1})$, $e(c_2, s_{2,1})$, $e(c_3, s_{1,2})$ e $e(c_5, s_{3,1)}$ fazem parte de M. As informações de capacidade residual dos vértices de S devem ser atualizadas de acordo com as informações de carga dos vértices de C. Por exemplo, na figura o vértice virtual $s_{2,1}$ é emparelhado com o vértice c_2. Nesse caso, o valor da $cap_{res_{s_{2,1}}}$ deve ser atualizado de acordo com a aplicação da Equação 6-5: $cap_{res_{s_2}} = 12 - 6$, onde $\Lambda_{s_2} = 6$.

Neste exemplo, a capacidade do serviço s_2 (cap_{s_2}) é igual a 12, assim como a capacidade de seus vértices virtuais $s_{2,1}$, $s_{2,2}$ e $s_{2,3}$. Como a aresta $e(c_2, s_{2,1})$ faz parte de M, $\Lambda_{s_2} = 6$, uma vez que $\lambda_{c_2} = 6$. Como todos os vértices virtuais compartilham a mesma capacidade, o valor da capacidade residual de cada um deles deve ser também atualizado, independentemente de ser saturado ou não. Por exemplo, na Figura 6.11, cap e cap_{res} dos vértices virtuais $s_{2,1}$, $s_{2,2}$ e $s_{2,3}$ possuem os mesmos valores, mesmo que apenas o vértice $s_{2,1}$ seja saturado.

Já para o serviço s_1, que possui dois vértices virtuais, $s_{1,1}$ e $s_{1,2}$, como representado na Figura 6.11, deve-se agregar as cargas individuais dos vértices c_1 e c_3 que incidem nos vértices virtuais $s_{1,1}$ e $s_{1,2}$, de acordo com a Equação 6-5: $cap_{res_{s_1}} = 10 - 9$, onde $\Lambda_{s_1} = 4 + 5$.

Neste exemplo, a capacidade do serviço candidato s_1 (cap_{s_1}) é igual a 10,

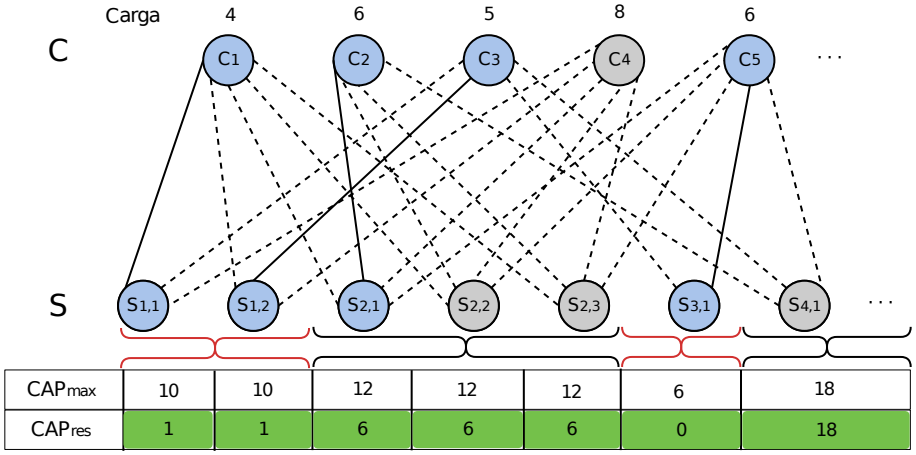


Figura 6.11: *Emparelhamento sobre o grafo G. Informações da capacidade máxima cap e capacidade residual cap_{res} são representadas ao abaixo de cada vértice em S.*

assim como a capacidade de seus vértices virtuais $s_{1,1}$ e $s_{1,2}$. Como as arestas $e(c_1, s_{1,1})$ e $e(c_3, s_{1,2})$ fazem parte de M, $\Lambda_{s_1} = 9$, uma vez que $\lambda_{c_1} = 4$ e $\lambda_{c_3} = 5$. Como todos os vértices virtuais de s_1 compartilham a mesma capacidade, o valor da capacidade residual de cada um deles deve ser também atualizado.

Dadas as informações dos vértices virtuais e as informações da capacidade máxima dos serviços, inclusive capacidade residual, o algoritmo de emparelhamento original necessita de algumas extensões para garantir a restrição de prevenção de sobrecarga. As informações de capacidade máxima e capacidade residual de cada vértice de S, bem como as informações de carga de cada vértice de C, são utilizadas na solução do emparelhamento ao buscar por um novo *caminho m-aumentante* a partir de um vértice exposto de C, como é o caso do vértice c_4 na Figura 6.11.

No algoritmo estendido, como no algoritmo original, para cada vértice s_j exposto de G, é buscado um *caminho m-aumentante* a partir de s_j. Caso um *caminho m-aumentante* seja encontrado em G, antes de adicionar o conjunto das novas arestas ao novo emparelhamento M', é necessário verificar o valor da capacidade residual de cada vértice s_j presente no *caminho m-aumentante* descoberto. Em outras palavras, no algoritmo estendido, para cada aresta $e(c_i, s_j)$ do *caminho m-aumentante* ser adicionada ao emparelhamento M', deve-se verificar a condição $cap_{s_j} \geq \Lambda_{s_j}$. Caso a condição não seja satisfeita, o algoritmo estendido segue seu fluxo normal, ou seja, descarta-se o *caminho m-aumentante* corrente e um novo *caminho m-aumentante* deve ser buscado a partir do mesmo vértice exposto c_i.

A Figura 6.12 mostra um emparelhamento M em G, onde as arestas em linha contínua, $e(c_1, s_{1,1})$, $e(c_2, s_{2,1})$, $e(c_3, s_{1,2})$ e $e(c_5, s_{3,1})$ pertencem a M. Além disso, a figura também mostra um exemplo de *caminho m-aumentante* (arestas com linhas na cor laranja) encontrado em G a partir do vértice exposto c_4. Os vértices de cor cinza

representam vértices que ainda não fazem parte do emparelhamento M, enquanto os vértices na cor azul já fazem parte de M. Os vértices expostos c_4 e $s_{2,2}$, circundados, representam as extremidades de um *caminho m-aumentante* descoberto.

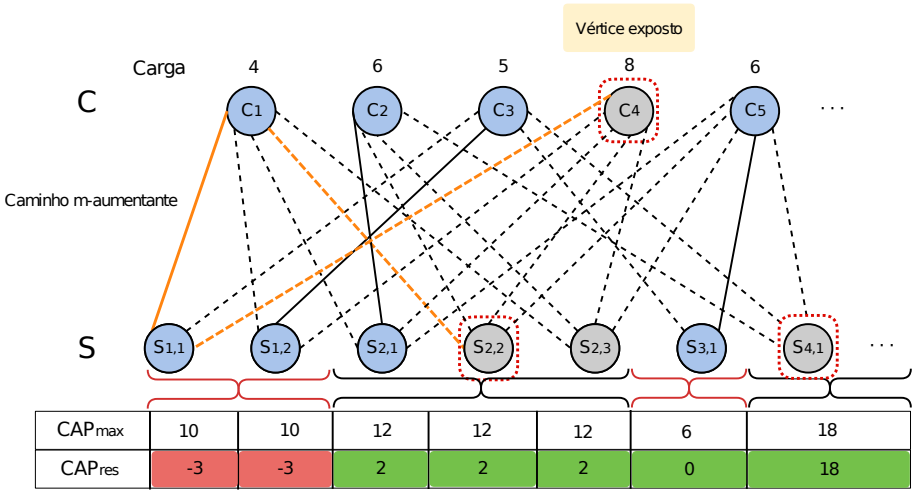


Figura 6.12: *Busca por um caminho* m-aumentante *em G, linhas alaranjadas, a partir do vértice exposto c_4. Esse caminho* m-aumentante *não satisfaz a condição 6-4, uma vez que a capacidade residual dos serviços virtuais $s_{1,1}$ e $s_{1,2}$ estão com valores negativos (marcados em vermelho). As arestas em linha contínua fazem parte do emparelhamento M.*

No exemplo da Figura 6.12, o *caminho m-aumentante* contém os seguintes vértices: c_4, $s_{1,1}$, c_1 e $s_{2,2}$. Neste caso, deve-se verificar se o valor da capacidade de cada vértice de S pertencente ao *caminho m-aumentante* é positivo, ou seja, $cap_{res_{s_{1,1}}} \geq 0$ e $cap_{res_{s_{2,2}}} \geq 0$. De acordo com a Figura 6.12, no algoritmo estendido os valores da capacidade residual dos vértices $s_{1,1}$ e $s_{1,2}$ são negativos (marcados com vermelho) e o valor da capacidade residual dos demais vértices são positivos (marcados com verde). Portanto, esse *caminho m-aumentante* deve ser descartado e o algoritmo de emparelhamento estendido procede a busca por outro *caminho m-aumentante*.

Já a Figura 6.13 mostra um outro *caminho m-aumentante* (arestas com linhas na cor laranja) encontrado em G a partir do vértice exposto c_4. O *caminho m-aumentante* contém os seguintes vértices: c_4, $s_{2,1}$, c_2 e $s_{4,1}$. Neste caso, deve-se verificar se o valor da capacidade residual de cada vértice de S pertencente ao *caminho m-aumentante* é positivo, ou seja, $cap_{res_{s_{2,1}}} \geq 0$ e $cap_{res_{s_{4,1}}} \geq 0$. Neste caso, os valores de capacidade residual dos vértices $s_{2,1}$ e $s_{4,1}$ são positivos (marcados com verde). Logo, o *caminho m-aumentante* é aceito e o emparelhamento M é incrementado, criando um novo emparelhamento M'.

A Figura 6.14 mostra o novo emparelhamento M' que cobre todos os vértices de S (arestas na cor laranja). Como todos os vértices de C estão saturados em M',

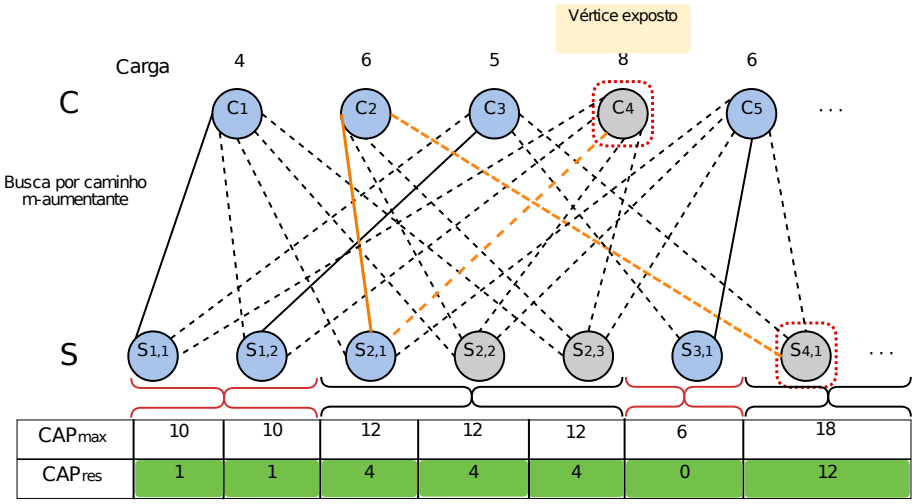


Figura 6.13: *Busca por um novo caminho* m-aumentante *em G, linhas alaranjadas, a partir do vértice exposto c_4. Esse caminho* m-aumentante *satisfaz a condição 6-4, uma vez que a capacidade residual dos serviços virtuais $s_{2,1}$ e $s_{4,1}$ estão com valores positivos (marcados em verde)*

.

o algoritmo é finalizado com as seguintes arestas emparelhadas: $e(c_1, s_{1,1})$, $e(c_2, s_{4,1})$, $e(c_3, s_{1,2})$, $e(c_4, s_{2,1})$ e $e(c_5, s_{3,1})$. Cada aresta $e(c_i, s_j)$ do emparelhamento significa que o serviço candidato s_j foi selecionado para desempenhar o respectivo papel na coreografia c_i.

Em resumo, ao aplicar esta extensão, o algoritmo húngaro é capaz encontrar *caminhos m-aumentantes* sem causar a sobrecarga dos serviços candidatos que participam de um emparelhamento. Para tanto, basta adicionar ao algoritmo de emparelhamento original uma verificação da capacidade residual dos vértices de S no processo de busca por *caminhos m-aumentantes*.

Vale ressaltar que, ao aplicar essa extensão, alguns *caminhos m-aumentantes* válidos, do ponto de vista do algoritmo original, são descartados e, por isso, o tempo de busca tende a aumentar, devido ao incremento no número de buscas por *caminhos m-aumentantes*.

6.4 Conclusão

O problema da síntese de serviços consiste em selecionar os serviços mais adequados para implantar um conjunto de coreografias de forma a satisfazer os requisitos de QoS e minimizar os custos em relação aos recursos utilizados pelos serviços. Este capítulo, descreve a estratégia proposta na tese para solucionar este problema por meio de extensões ao algoritmo húngaro.

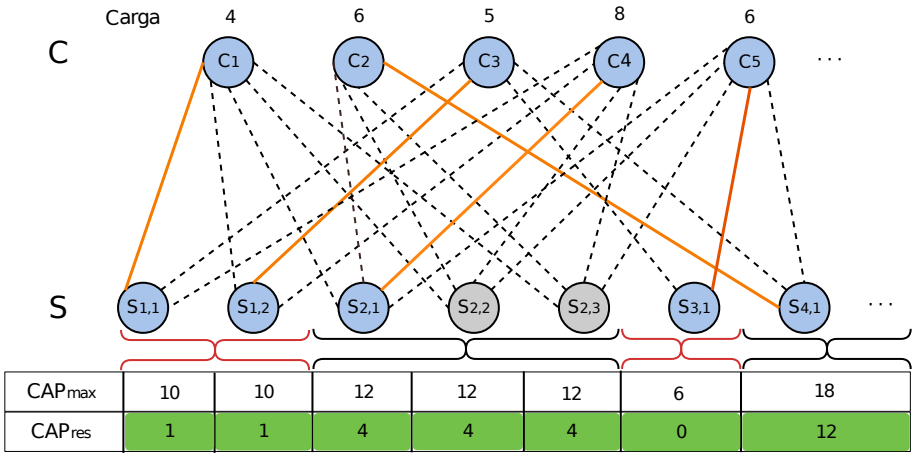


Figura 6.14: *Emparelhamento em G que cobre todos os vértices de C e que satisfaz a condição 6-4, uma vez que a capacidade residual de todos os serviços de S estão com valores positivos (marcados em verde)*

.

Os parâmetros de entrada para a síntese de serviços são um grafo de dependências e um conjunto de serviços candidatos, onde o problema é transformado no problema de emparelhamento sobre grafos bipartidos. Dados esses dois parâmetros de entrada, é iniciado o processo de construção dos grafos bipartidos, com uma bipartição representando as coreografias, com respectivos requisitos de QoS e carga, que compartilham um papel, e a outra bipartição representando os serviços candidatos, juntamente com suas ofertas de QoS e capacidade. Além disso, é calculado o custo das arestas, utilizando a função de utilidade proposta nesta tese.

Para encontrar um emparelhamento de custo mínimo sobre o grafo bipartido é utilizado o algoritmo húngaro. Entretanto, como o problema da síntese de serviços possui três requisitos adicionais, foi proposta extensões ao algoritmo húngaro. Sendo assim, inicialmente é aplicada a extensão para permitir o compartilhamento de serviços, e, em seguida, a extensão que busca aumentar a taxa de utilização dos serviços selecionados, através do uso de uma heurística. Depois de aplicar as duas extensões, é definido um novo grafo com adição dos vértices virtuais e com aumento das arestas admissíveis, através do uso do coeficiente de aproximação. Por fim, o algoritmo húngaro estendido é executado para resolver o problema da síntese de serviços. A saída é o grafo de dependências, onde os papéis são substituídos por serviços concretos.

A síntese de serviços consiste na principal contribuição desta tese, desde a definição do problema da síntese de serviços como um problema de emparelhamento e sua formalização como um problema de programação linear inteira até o algoritmo de emparelhamento estendido. Sendo assim, a síntese de serviços permite que seja selecionado um conjunto de serviços para implantação de múltiplas coreografias, satisfazendo um conjunto de requisitos de QoS e levando em consideração o custo

dos recursos utilizados pelos serviços selecionados.

Ao realizar a síntese de serviços para um conjunto de coreografias, deseja-se que a seleção de serviços, além de garantir os requisitos de QoS, minimize o custo em relação ao uso de recursos computacionais. Ao aplicar a estratégia de avaliação de QoS *best-fit* na síntese de serviços cria-se objetivos conflitantes. Esses conflitos ocorrem porque, ao selecionar serviços de acordo com a distância entre a QoS requisitada e a QoS ofertada para todas as coreografias ao mesmo tempo, tende-se a diminuir a taxa de utilização dos serviços, aumentando o desperdício de recursos. Para resolver esse problema, introduzimos uma heurística que utiliza um parâmetro denominado θ, para aumentar a taxa de utilização dos serviços selecionados mesmo aplicando a estratégia de avaliação de QoS *best-fit*.

De uma perspectiva prática, se a única preocupação com a execução da síntese de serviços for a distância de QoS, selecionar um valor para θ é trivial ($\theta = 0$). Analogamente, se a única preocupação é a taxa de utilização, podemos prontamente selecionar um valor alto para θ, de modo a ignorar o custo ao não analisar a distância de QoS. No entanto, os provedores de coreografias impõem restrições nos custos das coreografias, o que significa que adoção de um valor para θ de modo a alcançar um equilíbrio entre esses dois objetivos resulta em uma economia de recursos computacionais, conforme os resultados discutidos no Capítulo 8. Cada cenário de aplicação para o problema da síntese de serviços requer um valor apropriado para θ. Determinar esse valor θ para cada cenário é uma tarefa não trivial, pois depende dos valores dos requisitos de QoS das coreografias e da oferta de QoS pelos serviços candidatos.

A busca por um modelo para descrever o valor mais adequado para θ em qualquer cenário é considerado como um trabalho futuro. Neste sentido, uma possível solução seria a utilização da teoria de pós-otimização, como a solução adotada por [98]. A teoria de pós-otimização, desenvolvida para programação linear, busca determinar intervalos válidos para os parâmetros de um modelo, para os quais uma solução ótima permanece inalterada [50]. Sendo assim, pode-se investigar um método, baseado na teoria de pós-otimização, que possa identificar um intervalo de valores para θ que mantenha a qualidade da solução para diferentes conjuntos de entrada para a síntese de serviços, ou seja, para diferentes requisitos de QoS das coreografias e diferentes ofertas de QoS pelos serviços candidatos.

A seguir, no próximo capítulo, é apresentada uma arquitetura que agrega todas as técnicas apresentadas neste capítulo (e nos capítulos anteriores) para a realização da seleção de serviços eficiente para múltiplas coreografias.

CAPÍTULO 7

Arquitetura e implementação

Este capítulo apresenta uma arquitetura que integra, de forma coordenada, as estratégias propostas nesta tese para seleção de serviços sensível à QoS e à capacidade para múltiplas coreografias. A arquitetura considera os passos da abordagem, conforme ilustrado na Figura 1.3, para permitir uma seleção eficiente de serviços para múltiplas coreografias. Este capítulo também discute a implementação de um protótipo usado para avaliar a abordagem proposta.

O restante deste capítulo está organizado da seguinte forma. A Seção 7.1 apresenta a arquitetura em camadas para definição do processo de seleção de serviços. A arquitetura considera a análise e o gerenciamento de informações de um conjunto de coreografias. Estas informações são utilizadas para selecionar os serviços que serão usados na implantação das coreografias de forma a garantir os requisitos de QoS e diminuir os custos.

A Seção 7.2 apresenta em detalhes a camada de análise de coreografias, onde são discutidas as funcionalidades propostas e suas respectivas implementações. A Seção 7.3 discute a camada de síntese de coreografias, responsável por combinar múltiplas coreografias em uma única representação. Por fim, a camada de seleção de serviços é discutida na Seção 7.4, que apresenta os principais aspectos da implementação da síntese de serviços.

O trabalho deste capítulo gerou a publicação [78].

7.1 Arquitetura geral para seleção de serviços para múltiplas coreografias

Esta seção tem por objetivo apresentar uma visão em camadas da abordagem proposta nesta tese, propondo uma arquitetura para tratar a seleção de serviços, de acordo com o ciclo de vida de uma coreografia, como ilustrado na Figura 2.2. Nesta tese é proposta uma arquitetura que fornece uma abstração dos detalhes da especificação de coreografias e do processo seleção de serviços para múltiplas coreografias.

A arquitetura é baseada em uma máquina de execução de modelos semelhante à *Communication Virtual Machine – (CVM)*, um ambiente de tempo de execução que provê suporte para a coordenação automática de serviços de comunicação centrados no usuário com base no uso de modelos [37]. Na abordagem proposta nesta tese, modelos criados por especialistas de domínio, representando as coreografias de serviços, são analisados e transformados em novas representações para permitir a seleção de serviços de forma a atender aos requisitos do provedor de cada coreografia.

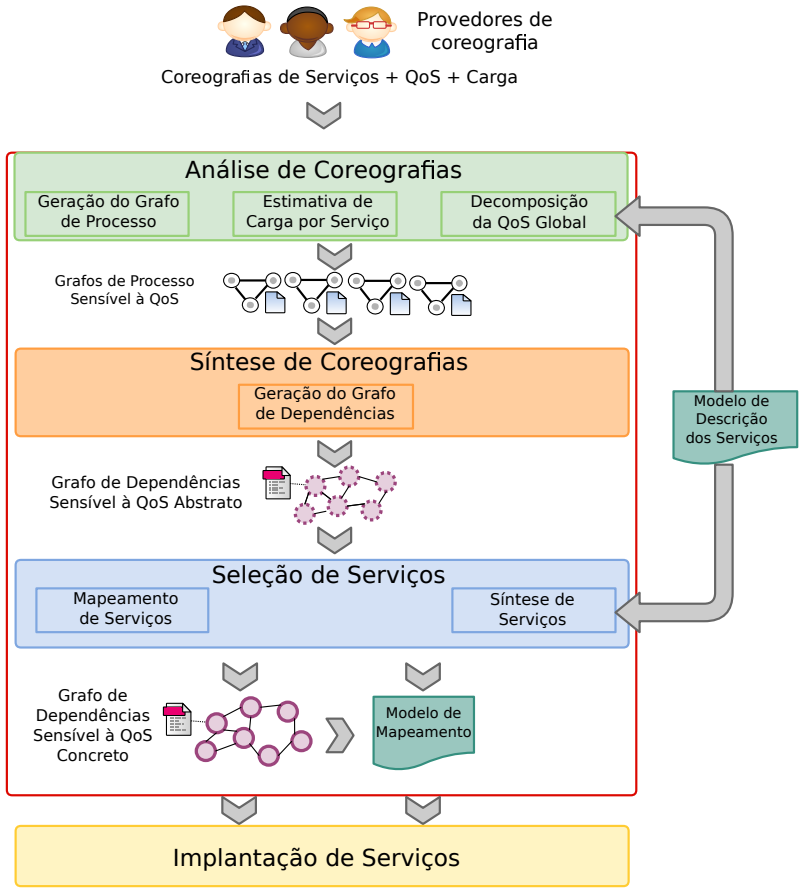


Figura 7.1: *Arquitetura geral proposta.*

Conforme ilustrado na Figura 7.1, a arquitetura está estruturada em três camadas: análise de coreografias, síntese de coreografias e seleção de serviços, seguindo os passos da abordagem proposta na tese, conforme ilustrado na Figura 1.3.

A primeira camada da arquitetura, ***análise de coreografias***, é responsável pela análise das coreografias de serviços submetidas pelos provedores. Nesta camada, cada especificação de coreografia, juntamente com as informações de requisitos de QoS e carga, é transformada em um grafo de processo sensível à QoS, onde requisitos de QoS e carga são anotados diretamente em cada papel presente na coreografia. Essa representação interna serve como entrada para a segunda camada.

A segunda camada da arquitetura, ***síntese de coreografias***, é responsável pela

geração do grafo de dependências, a partir de um conjunto de grafos de processo. Mais especificamente, todos os vértices (papéis) equivalentes dos grafos de processo são combinados em um único vértice do grafo de dependências. Além disso, os requisitos de QoS e carga dos vértices originais são anotados separadamente nos vértices combinados, a fim de preservar os requisitos dos grafos de processo originais. O grafo de dependências gerado nesta camada é a entrada para a próxima camada, seleção de serviços, juntamente com o modelo de descrição dos serviços.

A camada de ***seleção de serviços***, ao receber um grafo de dependências, realiza a seleção de serviços para o grupo de coreografias nele representado, empregando o algoritmo de síntese de serviços, discutido no Capítulo 6. O resultado produzido por esta camada é o grafo de dependências concreto, onde os papéis são substituídos pelos serviços selecionados. O grafo de dependências concreto é enviado à um serviço de ***implantação de serviços***, responsável por implantar os serviços selecionados no ambiente de implantação de coreografias escolhido. A implantação dos serviços está fora do escopo desta tese, podendo ser realizada por uma *engine* de implantação de coreografias, como aquela proposta por Gomes [51] ou a *CHOReOS Enactment Engine*, proposta por Leite *et al.* [74].

A arquitetura proposta encapsula funcionalidades específicas para seleção dos serviços utilizados para a implantação combinada de múltiplas coreografias de serviços. Sua definição emprega o princípio arquitetônico de separação de interesses que é a base para sua flexibilidade, uma vez que a abordagem de seleção de serviços é independente das linguagens de especificação das coreografias e do ambiente de implantação de coreografias. Para demonstrar e validar a abordagem proposta, implementamos um protótipo da arquitetura apresentada.

A seguir, nas Seções 7.2, 7.3 e 7.4, cada camada é descrita em detalhes, inclusive seus aspectos de implementação.

7.2 Análise de coreografias

A camada de *análise de coreografias* constitui o ponto de interação com os provedores de coreografias, através da definição de uma interface. Um provedor de coreografias pode especificar uma coreografia de serviços diretamente na notação de grafo de processo ou por meio de alguma linguagem específica para modelagem de coreografias (por exemplo, diagrama de coreografia da BPMN2 anotado com QoS e carga). Neste último caso, a coreografia de serviços deve ser convertida para a representação de grafo de processo, utilizando o modelo descrito na Seção 5.1.

O objetivo da camada de análise de coreografias é gerar uma representação interna das coreografias submetidas pelos provedores. Para cada coreografia de servi-

ços, é realizada uma análise das informações de requisitos de QoS e carga estimada, de forma a gerar uma representação onde tais informações são anotadas diretamente em cada papel de uma coreografia. Para estes fins, as principais funcionalidades da camada de análise de coreografias são: geração do grafo de processo, estimativa de carga por serviço e decomposição da QoS global. A Figura 7.2 apresenta a arquitetura desta camada.

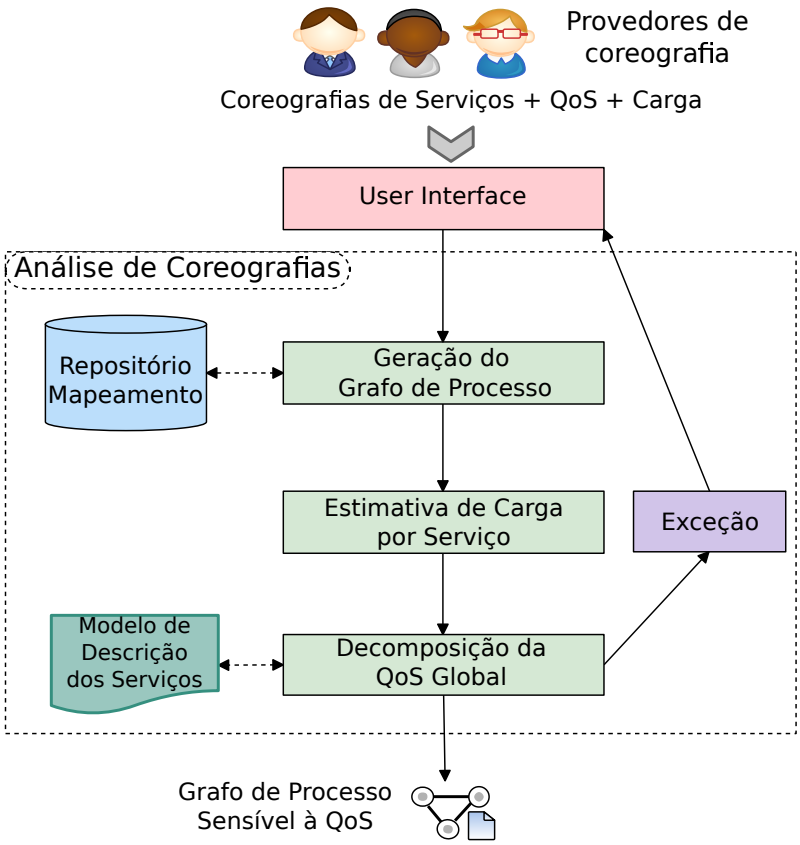


Figura 7.2: *Camada Análise de Coreografias.*

O componente *geração do grafo de processo* tem por objetivo converter uma coreografia especificada por meio de BPMN2 para modelagem de coreografias em um grafo de processo. Ao receber uma coreografia este componente acessa o repositório para acessar um modelo de mapeamento adequado e, em seguida, o grafo de processo correspondente é gerado. O repositório armazena as regras de mapeamento dos elementos em determinada linguagem de especificação (como BPMN2) em componentes do grafo de processo. Usando as regras de mapeamento do repositório é possível gerar modelos na notação grafo de processo a partir de modelos em diferentes linguagens de especificação de coreografias.

O componente de *estimativa de carga por serviço* realiza a decomposição da carga estimada da coreografia para os papéis que a compõem. Este componente recebe um grafo de processo e informações de requisitos de QoS e carga e, então, inicia

a construção do grafo de processo sensível à QoS. Ao decompor a carga da coreografia este componente anota para cada papel a informação da carga. Em seguida, este grafo de processo modificado é enviado para o próximo componente, que irá adicionar também informações de requisitos de QoS para cada papel.

O componente de *decomposição da QoS global*, fazendo uso do modelo de descrição de serviços, decompõe os requisitos de QoS globais em requisitos de QoS locais para cada grafo de processo, objetivando garantir os requisitos globais ao mesmo tempo que torna o processo de seleção eficiente. Caso não seja possível decompor a QoS global, uma exceção é gerada e o provedor de coreografias é notificado. Este por sua vez pode relaxar os requisitos de QoS e submeter novamente a coreografia. A saída deste componente é um grafo de processo sensível à QoS.

A seguir estas funcionalidades são descritas, juntamente com principais aspectos de implementação.

7.2.1 Implementação da análise de coreografias

Como o objetivo da implementação do protótipo discutido neste capítulo é demonstrar a abordagem proposta, é assumido que a especificação de cada coreografia de serviços é dada diretamente na notação de grafo de processo. Logo, o primeiro elemento na camada de análise de coreografias implementado foi um mecanismo para a representação dos grafos de processo. Em nossa implementação, a especificação de cada grafo de processo é representada em um arquivo XML, seguindo os modelos propostos em [74, 43].

Ao receber uma especificação de coreografia utilizando a representação de grafo de processo, a camada de análise de coreografias deve realizar, inicialmente, a estimativa de carga para cada papel que faz parte da coreografia, uma vez que o provedor de coreografias estima a carga de entrada por coreografia.

Na especificação de um grafo de processo, o provedor de coreografias ao definir os vértices representando os conectores de disjunção XOR^f e OR^f deve anotar explicitamente para cada aresta ϵ a probabilidade do fluxo de requisições seguir para cada vértice de destino. No exemplo da Figura 5.1, são anotadas as probabilidades para as arestas *Realizar pedido* e *Reservar mesa*, de forma que uma requisição pode ser encaminhada com uma probabilidade de *20%* para *Serviço pedido* e com uma probabilidade de *80%* para *Serviço restaurante*.

Para estimar a carga para cada papel, é analisada a aresta que incide em cada vértice e o tipo do respectivo vértice. O Algoritmo 7.1 descreve o processo para estimativa de carga para cada papel em um grafo de processo dada uma carga para a coreografia, com base no modelo da Tabela 5.3.

Algoritmo 7.1: Algoritmo de estimativa de carga para cada papel em um grafo de processo.

Entrada: Grafo de processo $GP = (V_{GP}, E_{GP})$ e carga estimada l.
Saída: Grafo de processo com informação de carga GPQ.

```
1  GPQ = GP
2  para cada e ∈ E_GP faça
3      se e_→ ⊆ (b ∪ P ∪ AND^j) & e⃗ ⊆ P então
4          λ = l
5          addLoad(GDQ.getEdge(e_→, e⃗, o), λ)
6      fim
7      se e_→ ⊆ (OR^f ∪ XOR^f) & e⃗ ⊆ P então
8          λ = pr * l
9          addLoad(GDQ.getEdge(e_→, e⃗, o), λ)
10     fim
11 fim
12 retorna GDQ
```

O algoritmo de estimativa de carga analisa o padrão de composição do vértice para cada aresta do grafo de processo e gera as informações sobre a carga estimada para cada papel. Na Linha 3 são verificadas as arestas cujos vértices de origem enviam a mesma carga que recebem para o vértice de destino, ou seja, vértices que representam conectores do tipo AND^j. Na Linha 7 são verificadas as arestas cujos vértices de origem enviam uma carga diferente do que recebem para o vértice de destino, ou seja, vértices que representam conectores do tipo OR^f e tipo XOR^f. Esse algoritmo gera um grafo de processo com informações de carga para cada aresta cujo vértice de destino é um papel de uma coreografia.

O grafo de processo com informações de carga é enviado para o componente decomposição da QoS global para finalizar o processo de geração do grafo de processo sensível à QoS. Para implementar o componente de *Decomposição da QoS Global* esta tese utiliza uma solução baseada no método proposto por Alrifai *et al.* [5], onde um requisito C de QoS global é decomposto em um conjunto de $\mathcal{N}$ requisitos $C_1, C_2, ..., C_\mathcal{N}$ de QoS locais sendo $\mathcal{N}$ o número de papéis no grafo de processo. Esse processo somente é realizado quando o provedor de coreografias anota, junto à coreografia de serviços, os requisitos de QoS globais, uma vez que o provedor de coreografias pode anotar apenas requisitos de QoS locais.

Há duas etapas principais neste processo: (1) determinação dos níveis de QoS e a (2) definição do conjunto de requisitos de QoS locais para cada papel. Na primeira etapa é utilizado o conceito de níveis de QoS, que são definidos como um conjunto

de valores representativos. Na segunda etapa, dado o conjunto de níveis de QoS de cada papel, cada requisito de QoS global é mapeado em um desses níveis de QoS, onde os níveis de QoS selecionados são usados como limites locais para os atributos de QoS correspondentes. No final do processo, a agregação dos requisitos de QoS locais não pode violar o requisito de QoS global. Para uma dada coreografia, pode existir um requisito de QoS global para cada um dos k atributos de QoS. Logo, é necessário encontrar o mesmo número de requisitos de QoS locais para cada papel.

Para definir um conjunto de valores representativos, a primeira etapa utiliza as informações de ofertas de QoS dos serviços candidatos presentes no modelo de descrição de serviços como base para definição dos valores dos atributos dos requisitos de QoS locais. Sendo assim, cada atributo de QoS dos serviços candidatos para um determinado papel é dividido em um conjunto de valores discretos de QoS, sendo chamados de níveis de QoS [4]. Como os valores de níveis de QoS são baseados nos valores dos serviços candidatos, deve-se mapear os papéis presentes no grafo de processo em tipos de serviços disponíveis. Como discutido no Capítulo 5, assumimos que cada papel é mapeado para um único tipo de serviço. Logo os termos papel e tipo de serviço no grafo de processo são tratados de forma intercambiável.

Os níveis de QoS são inicializados para cada papel, dividindo o intervalo de valores de cada atributo de QoS em um conjunto de d valores discretos de QoS, como mostrado na Figura 7.3, onde $\mathcal{A}_k^m$ indica o valor do *k-ésimo* atributo de QoS do *m-ésimo* serviço candidato para o papel p, $L_{\mathcal{A}_k}^d$ representa o *d-ésimo* nível de QoS do *k-ésimo* atributo de QoS do respectivo papel.

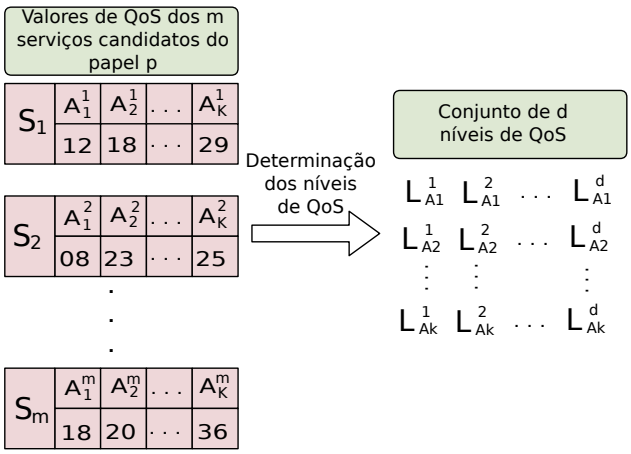


Figura 7.3: *Determinação dos d níveis de QoS para os k atributos de QoS para o papel p.*

A definição dos níveis de QoS é executada separadamente para cada atributo de QoS $\mathcal{A}_k$ que faz parte de um requisito de QoS global, onde os d níveis de QoS para cada atributo são definidos como: $qmin_{\mathcal{A}_k} \leq L_{\mathcal{A}_k}^1 \leq L_{\mathcal{A}_k}^2 \leq \ldots \leq L_{\mathcal{A}_k}^d \leq qmax_{\mathcal{A}_k}$. Sendo assim, o algoritmo para cálculo dos níveis de QoS é dividido em três passos:

1. Ordenação: ordenar os serviços candidatos do papel p com base em seus respectivos valores de QoS para cada atributo.
2. Particionamento: particionar os serviços classificados em d subconjuntos, onde os valores mínimo e máximo são adicionados diretamente ao conjunto de níveis de QoS.
3. Seleção: para cada subconjunto separadamente, um serviço é selecionado aleatoriamente, onde o valor do seu atributo de QoS é definido como um nível de QoS.

Depois de executar a primeira etapa, onde os d níveis de QoS foram definidos, a segunda etapa consiste em decompor os requisitos de QoS globais com base nos d níveis de QoS. Dado um requisito C de QoS global, para uma coreografia de serviços com $\mathcal{N}$ papéis e um conjunto com d níveis de QoS para os respectivos atributos de QoS para cada papel, o objetivo da segunda etapa é decidir qual nível de QoS deve ser usado como requisito de QoS local. Alrifai *et al.* [5] formularam esta etapa de problema como uma programação linear inteira mista. Nesta tese, um algoritmo guloso foi utilizado para mapear cada requisito de QoS global em um dos níveis de QoS. Os requisitos de QoS locais devem garantir que se os serviços candidatos para cada papel puderem atender tais requisitos, os valores de QoS agregados desses serviços candidatos também podem satisfazer os requisitos de QoS globais. Por fim, os valores de requisito C de QoS e carga λ, por papel, são anotados diretamente no grafo de processo, gerando o grafo de processo sensível à QoS, como aquele apresentado na Figura 5.2.

Em nossa implementação, a especificação de cada GPQ é representada em um arquivo XML consistindo em três partes. A primeira parte contém a definição dos diferentes vértices (papéis), bem como dos conectores. A parte intermediária contém a definição das arestas, representando o vértice de origem, o vértice de destino e a operação. A última parte contém o requisito de QoS e a carga para cada papel.

Cada vértice é definido por um tipo que estabelece os atributos que obrigatoriamente devem ser especificados. O diagrama da Figura 7.4 descreve os quatro tipos de vértices permitidos em um GPQ. Para um vértice definido como *START*, que indica o vértice inicial do grafo de processo sensível à QoS, deve-se descrever as arestas incidentes nos vértices sucessores. Um vértice definido como *END* indica o vértice final do grafo de processo sensível à QoS e não possui nenhum vértice sucessor. Para cada vértice definido como *ROLE* há a descrição das operações que ele oferece assim como as arestas incidentes nos vértices sucessores.

Para os demais vértices deve-se definir o tipo de conector, de acordo com os tipos de conectores disponíveis: AND^f, OR^f, XOR^f, AND^j, OR^j e XOR^j. Para cada um desses tipos de vértice, deve-se descrever as arestas incidentes nos vértices sucessores.

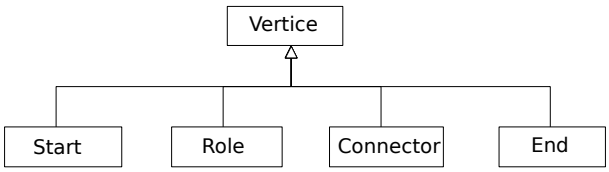


Figura 7.4: *Tipos de vértices em um grafo de processo.*

Depois de descrever os vértices do GPQ, deve-se especificar o conjunto de arestas que definem o fluxo do grafo de processo. Para cada aresta, além do identificador, deve-se especificar o vértice de origem da aresta e o vértice de destino da aresta. Para as arestas que possuem vértice de destino, definidos como *ROLE*, também deve-se especificar a operação sendo requisitada no vértice de destino. Quando o vértice de origem for um conector de disjunção do tipo *fork* (OR^f e XOR^f) também é especificada a probabilidade de execução para o vértice de destino.

Por fim, as informações do requisito de QoS e a carga estimada devem ser anotados para cada papel. Nesse caso, para cada vértice de destino definido como *ROLE*, descrevem-se os requisitos de QoS e a carga para a operação do papel. Os requisitos dos provedores de coreografias são descritos como um conjunto de atributos de QoS juntamente com a carga estimada. Os requisitos de QoS são especificados usando-se as definições apresentadas no Capítulo 5. As métricas implementadas são tempo de resposta, disponibilidade e reputação. As informações para cada atributo de QoS, seguindo a Definição 4.1, são:

- nome da métrica;
- operador relacional;
- valor alvo requisitado da métrica;
- peso do atributo; e
- classificação da métrica: `Alvo`, `Requirido` e `Otimizado`.

No Apêndice A é apresentada a representação XML do grafo de processo sensível à QoS para a coreografia de *consulta de rota* apresentada na Figura 5.2.

7.3 Síntese de coreografias

A camada de *síntese de coreografias* tem por objetivo a geração do grafo de dependências sensível à QoS abstrato (GDA). Todos os grafos de processo recebidos da camada de análise de coreografias são sintetizados em uma única representação contendo todas as informações necessárias para a seleção de serviços de forma a atender aos requisitos das coreografias individuais ao mesmo tempo que se otimiza o custo dos serviços selecionados.

No processo de geração do grafo de dependências abstrato, todos os papéis equivalentes dos grafos de processo são combinados em um único vértice do GDA. Além disso, os requisitos de QoS dos papéis combinados são anotados separadamente, a fim de manter a especificação dos requisitos de QoS das coreografias originais. O mesmo é verdadeiro para a carga associada aos papéis que foram combinados.

A saída da camada de síntese de coreografias é um grafo de dependências que permite ter uma visão global dos papéis que são comuns a mais de uma coreografia, juntamente com os diferentes requisitos de QoS e carga de cada coreografia.

7.3.1 Implementação da síntese de coreografias

A camada de síntese de coreografias foi implementada usando o procedimento descrito no Algoritmo 7.2, de forma a sintetizar um grafo de dependências sensível à QoS abstrato a partir de um conjunto de grafos de processo sensíveis à QoS. A geração do grafo de dependências sensível à QoS abstrato é realizada usando-se a descrição dos grafos de processo como entrada, especificados isoladamente em cada arquivo XML, e construindo como saída um novo arquivo XML.

O Algoritmo 7.2 descreve o processo de geração de um grafo de dependências a partir de um conjunto de grafos de processo sensíveis à QoS. O algoritmo inicia com a eliminação dos conectores através da operação `reduce` (Linha 4) para cada GPQ. Em seguida, cada aresta no GPQ é então avaliada para verificar se ela contém algum vértice ainda não incluído no grafo de dependências resultante, de forma a haver apenas um único nó inicial e um único nó final. Caso a aresta contenha um desses vértices para o GPQ, ele é então rotulado como vértice equivalente no grafo de dependências (Linhas 6 e 9).

Em seguida, caso não exista a aresta no grafo de dependências equivalente, a aresta que está sendo avaliada é adicionada. Por fim, o requisito de QoS e a carga da aresta sendo avaliada é adicionada à aresta equivalente no grafo de dependências (Linha 23). No grafo de dependências gerado, o requisito de QoS e carga isolada de cada coreografia é mantida. A Figura 5.3 apresenta o grafo de dependências gerado ao combinar as três coreografias apresentadas no Capítulo 2 (Figuras 2.5, 2.6(a) e 2.6(b)).

Nesta tese é proposta uma representação XML para o grafo de dependências. O arquivo XML do grafo de dependências segue a mesma estrutura da representação XML do GPQ, com alterações na definição dos vértices (papéis) e na definição de requisitos de QoS. Em relação aos vértices, apenas vértices do tipo *START*, *END* e *ROLE* são permitidos, uma vez que os conectores são removidos. Além disso, de acordo com a implementação do algoritmo 7.2, para cada vértice de destino definido como *ROLE*, são anexadas informações de requisitos de QoS e carga das diferentes coreografias que possuem o respectivo papel em comum. Diante disso, na especificação dos requisitos

de QoS para cada papel, deve-se representar também o identificador da respectiva coreografia. Assim, no grafo de dependências, as informações de requisito de QoS e carga de cada coreografia são mantidas separadamente, embora o papel seja comum a mais de uma coreografia.

Algoritmo 7.2: Geração de um grafo de dependências a partir de um conjunto de grafos de processo.

Entrada: Conjunto de grafos de processo $GP[]$.
Saída: Grafo de dependências GD.

```
1  GD.addNode(b)
2  GD.addNode(z)
3  para cada gp_i em GP[] faça
4      reduce(pg_i)
5      para cada e ∈ E_{gp_i} faça
6          se e = b_{gp_i} então
7              e ← b
8          fim
9          se e ∈ Z_{gp_i} então
10             e ← z
11         fim
12         se GD não contém vértice e então
13             GD.addNode(e)
14         fim
15         se GD não contém vértice e então
16             GD.addNode(e)
17         fim
18         se GD não contém aresta (e, e, o) então
19             GD.addEdge(e, e, o)
20         fim
21         qos ← getQoSOfEdge(e, e, o)
22         λ ← getLoadOfEdge(e, e, o)
23         addQoSProp(GD.getEdge(e, e, o), qos, λ)
24     fim
25 fim
26 retorna gd
```

Esta representação de grafo de dependências também é mantida internamente como objetos Java representados usando uma lista de adjacências. A estrutura

de um grafo de dependências é realizada através da implementação das classes apresentadas no trecho do Código 7.1. Para um grafo de dependências são definidas as seguintes informações:

- identificador único do grafo de dependências;
- lista de vértices do grafo de dependências: cada vértice é uma instância da classe *Vertice* que define o identificador do vértice e o tipo (início, fim, papel ou conector);
- lista de arestas do grafo de dependências: cada aresta é uma instância da classe *Edge* que define o vértice de origem, o vértice de destino, operação sendo requisitada ao vértice de destino pelo vértice de origem e o requisito de QoS junto com a carga estimada;
- lista de adjacência: implementada usando um *Map T<K,V>*, onde a chave é definida como sendo os vértices do grafo de dependências e o tipo do valor é uma lista de arestas que incidem no respectivo vértice.

Código 7.1 Trecho do código que define a estrutura de um grafo de dependências (representado como um lista de adjacência).

```
1  class DependencyGraph {
2      Long id;
3      List<Vertice> verticeList;
4      List<Edge> edgeList;
5      Map<Vertice, List<Edge>> adjacencyList;
6      ...
7  }
8  class Edge {
9      Vertice source;
10     Vertice target;
11     String operation;
12     List<DesiredQoS> requirements;
13     Integer load;
14     ...
15 }
16 class Vertice {
17    String id;
18    String verticeType;
19    ...
20 }
21 c
```

Como discutido no Capítulo 6, o objetivo da síntese de serviços é encontrar o melhor emparelhamento de um conjunto coreografias com respectivos requisitos de QoS, sobre um papel comum entre elas, com os serviços candidatos. Entretanto, na implementação de um grafo de dependências sensível à QoS, as informações sobre o requisito de QoS e a carga para cada papel são anotados junto às arestas, seguindo a Definição 5.4, e não diretamente em cada papel. Logo, ao buscar as informações sobre

requisitos de QoS e carga, estamos interessados nas arestas cujo vértice de destino é do tipo *ROLE*, sendo que esse vértice representa o papel de uma coreografia sobre o qual os requisitos de QoS e carga são informados.

7.4 Seleção de serviços

A camada de *seleção de serviços* é responsável por selecionar um conjunto de serviços, garantindo os requisitos de QoS dos provedores ao mesmo tempo que busca minimizar os custos. Para isso, esta camada recebe como entrada um GDA, juntamente com as informações sobre os serviços candidatos. Um GDA, como definido na Seção 5.2, contém todas as informações necessárias para a seleção de serviços de forma a atender aos requisitos das coreografias nele combinadas. As principais funcionalidades da camada de seleção de serviços são: síntese de serviços e mapeamento de serviços. A Figura 7.5 apresenta o fluxo desta camada.

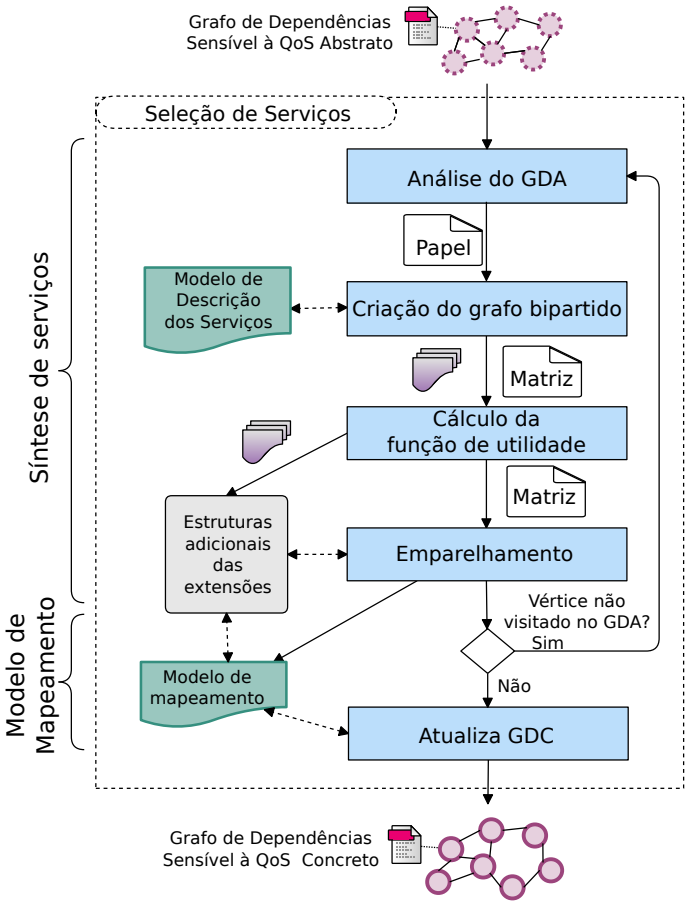


Figura 7.5: *Diagrama mostrando os detalhes da camada Seleção de Serviços.*

A funcionalidade de *síntese de serviços* é responsável pela execução da abordagem para seleção de serviços sensível à QoS e à capacidade para implantação eficiente

de múltiplas coreografias de serviços, discutida no Capítulo 6. Para isso, é analisado separadamente cada papel no GDA, onde são selecionados os serviços para cada coreografia que compartilha o respectivo papel. De acordo com a Figura 7.2, o componente *análise de GDA* é responsável por manter as informações sobre os papéis que estão sendo analisados e enviá-los para o componente responsável pela criação do *grafo bipartido*. Para criar o grafo bipartido, além das informações das coreografias que compartilham o papel em questão, é necessário buscar as informações dos serviços candidatos para o respectivo papel, acessando o modelo de descrição de serviços. O modelo de descrição dos serviços contém as informações atualizadas sobre os serviços candidatos, organizados em tipos de serviço.

O grafo bipartido é representado como uma matriz e enviado, juntamente com informações adicionais das coreografias e serviços candidatos, para o componente responsável pelo *cálculo da função de utilidade*, que deve adicionar na matriz informações do custo para uma coreografia selecionar um serviço candidato para o respectivo papel. Para isso, este componente utiliza a função de utilidade definida pela Equação 4-3. A matriz é enviada para o componente *emparelhamento* executar o algoritmo de emparelhamento. O componente *cálculo da função de utilidade* também envia as informações adicionais das coreografias e serviços candidatos para o componente *estruturas adicionais das extensões* que é responsável por definir e manter as estruturas de dados para implementação das extensões propostas sobre o algoritmo húngaro, propostas no Capítulo 6.

O componente *emparelhamento*, usando a matriz e as informações do componente *estruturas adicionais das extensões* realiza a seleção dos serviços candidatos mais adequados para um conjunto de coreografias, de forma a satisfazer os requisitos de QoS e minimizar os custos em relação aos recursos utilizados pelos serviços. Esse processo é executado para cada papel do GDA.

O modelo *mapeamento de serviços* recebe as informações sobre quais serviços candidatos foram selecionados para cada papel de cada coreografia específica, pelo componente *emparelhamento*. O modelo de mapeamento contém as informações dos serviços selecionados por coreografia de serviços. Esse modelo é utilizado pelo componente *atualiza GDC* responsável por atualizar as informações dos serviços selecionados no grafo de dependências concreto (GDC). No GDC, os papéis presentes no GDA são substituídos pelos serviços concretos selecionados. Os serviços presentes no GDC representam os serviços selecionados pelos provedores de coreografias.

7.4.1 Implementação da seleção de serviços

Para realizar sua função, a camada de seleção de serviços recebe como entrada um grafo de dependências sensível à QoS abstrato, juntamente com o modelo de

descrição de serviços. A informação atualizada sobre os serviços candidatos, organizados como tipos de serviço, é mantida no modelo de descrição dos serviços e inclui: descrição do serviço, propriedades de QoS e capacidade. Na implementação, os dados que descrevem os serviços candidatos são mantidos em arquivos *CSV* específicos para cada tipo de serviço. Com isso, dado um papel é possível obter os dados disponíveis de todos os seus serviços candidatos, uma vez que cada papel é mapeado para um único tipo de serviço.

A ideia geral da síntese de serviços é calcular o valor da função de utilidade para cada coreografia, com o respectivo requisito de QoS, associada a cada papel do GDA com relação aos respectivos serviços candidatos, armazenando o resultado em uma matriz representando um grafo bipartido, onde as linhas representam as coreografias, as colunas representam os serviços candidatos e os elementos representam os valores de utilidade. Em seguida, é realizado um pré-processamento sobre a matriz, aplicando-se as extensões discutidas no Capítulo 6. Por fim, o algoritmo de emparelhamento estendido é executado. Este processo é descrito no Algoritmo 7.3.

Para cada vértice do grafo de dependências abstrato (linha 2), o algoritmo obtém o conjunto de serviços candidatos (linha 3) presentes no modelo de descrição de serviços e que satisfazem o papel do vértice corrente. Vale ressaltar que para cada papel em uma coreografia existe um grupo de serviços candidatos, que são representados por um tipo de serviço. Sendo assim, o método $getCandidateServices(papel, mds)$ retorna um conjunto de serviços candidatos para o respectivo papel, onde cada serviço candidato é representado com um elemento do vetor de ofertas de QoS. Na Linha 4 os atributos de QoS dos serviços devem ser normalizados para eliminar a diferença de escala entre os vários atributos de QoS dos serviços candidatos. O resultado do método $normalize(vetorPropQoS[])$ é um novo vetor, chamado de $vetorOferQoS[]$, com os valores dos atributos de QoS normalizados, enquanto o vetor $vetorPropQoS[]$ mantém os valores originais.

Para cada coreografia da qual o papel faz parte, o laço que inicia na Linha 5 obtém o respectivo valor para a variável $desiredQoS$, que armazena informações da coreografia, juntamente com o requisito de QoS e a carga estimada para o papel em questão. Na Linha 6, os atributos de QoS de cada requisito da coreografia devem ser normalizados para eliminar a diferença de escala entre os vários atributos de QoS, como também realizado para as ofertas de QoS. O resultado do método $normalize(desiredQoS)$ é uma nova variável $reqQoS$ que guarda os valores dos atributos normalizados, enquanto o $desiredQoS$ mantém os valores originais.

Na Linha 7, obtêm-se as informações de ofertas de QoS de cada um dos serviços candidatos ao papel em questão, armazenado na variável $oferQoS$. Na Linha 8, é realizado o cálculo da função de utilidade do serviço candidato para a coreografia

representada pelo respectivo requisito de QoS. Ao aplicar o cálculo da função de utilidade é retornado um objeto, do tipo de dado $UtilPerService$. Esse tipo de dado guarda o valor da função de utilidade do serviço candidato analisado, juntamente com as informações do mesmo, armazenando em uma lista que mantém as utilidades de todos os serviços candidatos para a coreografia que contém o papel da iteração corrente.

Algoritmo 7.3: Pseudocódigo do algoritmo de síntese de serviços.

Entrada: Grafo de dependências abstrato GDA
Modelo de descrição de serviços mds
Saída: Grafo de dependências concreto GDC

```
1  GDC = GDA
2  para cada papel ∈ GDA faça
3      vetorPropQoS[] = getCandidateServices(papel, mds)
4      vetorOferQoS[] = normalize(vetorPropQoS[])
5      para cada desiredQoS ∈ papel faça
6          reqQoS = normalize(desiredQoS)
7          para cada oferQoS ∈ vetorOferQoS[] faça
8              utilPerService = util(oferQoS, reqQoS)
9              utilPerServiceList.add(utilPerService)
10         fim
11         utilChorServicesList.add(new(oferQoS, utilPerServiceList))
12     fim
13     matrizUtilidades[][] = criaMatrizUtil(utilChorServicesList)
14     inicializaEstruturas(utilChorServicesList)
15     matrizUtilAlterada[][] = maximizaTaxa(matrizUtilidades[][], θ)
16     newMatrizUtility[][] = compServicos(matrizUtilAlterada[][])
17     matchingChorService = emparelhamento(newMatrizUtility[][])
18     modeloMapeamento.add(matchingChorService)
19 fim
20 atualizaGDC(modeloMapeamento, GDC)
21 retorna GDC
```

Em seguida, na Linha 11, as informações da coreografia, juntamente com as utilidades para todos os serviços, são armazenadas em uma lista do tipo $UtilChorService$. Logo, no final do laço (na Linha 12), o próximo passo é a construção do grafo bipartido e a definição das estruturas de dados necessárias para aplicação do algoritmo de emparelhamento.

Na Linha 13 é construído o grafo bipartido utilizando uma matriz, aqui chamada de $matrizUtilidades[][]$. Na chamada do método $criaMatrizUtil()$, são atribuídos valores para a matriz de utilidade, onde as linhas representam a bipartição que contém as coreografias, com respectivos requisitos de QoS, que compartilham o papel corrente, e as colunas representam a bipartição que contém os serviços candidatos.

A Figura 7.6 exemplifica uma matriz de utilidades. No exemplo, cinco coreografias de serviços (*C1, C2, ..., C5*) possuem um papel em comum, sendo que há cinco serviços candidatos (*S1, S2, ..., S5*) para o papel em questão. Os elementos da matriz representam a utilidade calculada na Linha 8. Por exemplo, a utilidade do serviço candidato *S4* para a coreografia *C3* é *0,55*. A figura mostra ainda os vetores (em azul) que armazenam as informações da carga sobre o papel em cada coreografia da qual ele faz parte, bem como a capacidade de cada serviço candidato. Por exemplo, na coreografia *C1*, o papel possui uma carga estimada de *4* requisições por segundo e o serviço candidato *S2* possui uma capacidade máxima de *8*.

		Capacidade				
		6	8	4	2	6
		S1	S2	S3	S4	S5
4	C1	∞	0,08	0,20	0,18	0,32
2	C2	0,44	0,53	0,66	0,64	0,78
2	C3	0,36	0,45	0,57	0,55	0,69
2	C4	0,66	0,67	0,68	0,68	0,70
4	C5	0,16	0,25	0,38	0,36	0,50

Carga

Figura 7.6: *Matriz com a utilidade com todos os serviços candidatos para um papel comum entre um conjunto de coreografias. Além disso, são representados os vetores que armazenam informações sobre cada coreografia e sua respectiva carga, bem como os vetores armazenam informações sobre cada serviço candidato e sua respectiva capacidade.*

Em seguida, nas Linhas 14, 15 e 16 são realizadas operações de pré-processamento. O pré-processamento define as estruturas de dados para armazenar informações sobre as coreografias, cargas, serviços e capacidades, bem como suas relações. Além disso, também são aplicadas as extensões sobre o algoritmo húngaro original, onde tais extensões realizam modificações na matriz de utilidade original criando uma nova matriz de utilidades. Na implementação foi criada uma classe denominada $PreProcessamento$ que realiza esses três passos, ou seja, define as estruturas de dados bem como suas atualizações.

Na linha 14 são inicializadas as estruturas de dados para armazenar informações sobre cada coreografia e suas respectivas cargas, e informações sobre os servi-

ços candidatos, com respectivas informações sobre capacidade e capacidade residual. A definição dessas estruturas é representada na classe chamada $PreProcessamento$ que define as seguintes estruturas de dados:

- *vetLoad[]*: define um vetor com as cargas das coreografias que compartilham o papel corrente.
- *vetChor[]*: define um vetor que guarda a relação entre a coreografia e a posição na matriz de utilidades.
- *vetService[]*: define um vetor que guarda a relação entre o serviço candidato e a posição na matriz
- *vetCapacity[]*: define um vetor com a capacidade dos serviços candidatos.
- *vetCapacityResidual[]*: define um vetor com a capacidade residual dos serviços candidatos.

Na Linha 15, do Algoritmo 7.3, é aplicada a extensão que tem por objetivo alterar a matriz de utilidades, de forma a aumentar a taxa de utilização dos serviços selecionados quando o algoritmo de emparelhamento for aplicado. O método $maximizaTaxa()$ recebe como parâmetro a $matrizUtilidades[][]$, que representa o grafo bipartido, e o *coeficiente de aproximação*. Para cada linha da $matrizUtilidades[][]$ é buscado o elemento que possui o menor valor da função de utilidade calculada entre o requisito de QoS do papel da coreografia em questão (linhas da matriz) e seus serviços candidatos (colunas da matriz). Ainda na mesma linha, são buscados os elementos que possuem valores no intervalo entre o menor valor da função de utilidade e o *coeficiente de aproximação* (θ), onde tais elementos são alterados para um valor igual ao menor valor da função de utilidade da respectiva linha. A saída deste método é uma nova matriz ($matrizUtilAlterada[][]$), como mostra a Linha 15.

A Figura 7.7 mostra um exemplo de matriz de utilidade, onde os elementos marcados com verde representam os valores de utilidade equivalentes para cada linha, de acordo com o valor definido pelo *coeficiente de aproximação*. Por exemplo, para a coreografia *C4*, o serviço candidato *S1* possui a menor utilidade. Ao aplicar o coeficiente de aproximação, os serviços candidatos *S2, S3, S4 e S5*, que possuem valores próximos do serviço candidato *S1*, passam a ser tratados como tendo o valor de utilidade igual ao serviço *S1*, sendo marcados com verde. Já para a coreografia *C1*, onde o serviço candidato *S2* possui a menor utilidade, ao aplicar o coeficiente de aproximação os serviços candidatos *S1, S3, S4 e S5* continuam com valores acima do coeficiente de aproximação. Logo, nenhum serviço candidato terá seu valor de utilidade igualado ao valor do serviço candidato *S2*.

Na Linha 16 do Algoritmo 7.3 é aplicada a extensão que possibilita um serviço candidato ser selecionado por mais de uma coreografia ao mesmo tempo. Nesse

			Capacidade				
			6	8	4	2	6
			S1	S2	S3	S4	S5
Carga	4	C1	∞	0,08	0,20	0,18	0,32
	2	C2	0,44	0,53	0,66	0,64	0,78
	2	C3	0,36	0,45	0,57	0,55	0,69
	2	C4	0,66	0,67	0,68	0,68	0,70
	4	C5	0,16	0,25	0,38	0,36	0,50

Figura 7.7: *Matriz de utilidade onde as menores utilidades por linha são marcadas com verde.*

caso, para cada serviço candidato é calculado o número de vértices virtuais, que representa a quantidade de coreografias que podem compartilhar tal serviço candidato, desde que a capacidade máxima do serviço seja respeitada. Sendo assim, ao chamar o método $comServiços()$ ocorre o incremento do número de colunas na matriz de entrada, representando os *vértices virtuais* para cada serviço candidato representado nas colunas. A quantidade de vértices virtuais para cada serviço candidato representado na $matrizUtilAlterada[][]$ é calculada pela implementação da Equação 6-3. Para isso, são utilizadas as informações contidas nos vetores definidos no Código **??**, bem como é definido o vetor $copiesServicesArray[]$, que guarda a relação entre um serviço e seus respectivos vértices virtuais. O retorno do método $comServiços()$ é uma nova matriz denominada $newMatrizUtility[][]$. A Figura 7.8 mostra que o número de colunas aumentou de *5*, da matriz original da Figura 7.7, para *13*. Por exemplo, o serviço candidato *S1* possui *3* vértices virtuais, significando que ele pode ser compartilhado por até três coreografias de serviços, desde que a capacidade máxima seja respeitada.

			Capacidade											
			6	6	6	8	8	8	8	4	4	2	6	6	6
			S1	S1	S1	S2	S2	S2	S2	S3	S3	S4	S5	S5	S5
Carga	4	C1	∞	∞	∞	0,08	0,08	0,08	0,08	0,20	0,20	0,18	0,32	0,32	0,32
	2	C2	0,44	0,44	0,44	0,53	0,53	0,53	0,53	0,66	0,66	0,64	0,78	0,78	0,78
	2	C3	0,36	0,36	0,36	0,45	0,45	0,45	0,45	0,57	0,57	0,55	0,69	0,69	0,69
	2	C4	0,66	0,66	0,66	0,67	0,67	0,67	0,67	0,68	0,68	0,68	0,70	0,70	0,70
	4	C5	0,16	0,16	0,16	0,25	0,25	0,25	0,25	0,38	0,38	0,36	0,50	0,50	0,50

Figura 7.8: *Nova matriz de utilidade ($newMatrizUtility[][]$) depois de aplicar ao executar o método comServiços().*

Por fim, na Linha 17 do Algoritmo 7.3, o algoritmo de emparelhamento estendido é aplicado sobre matriz $newMatrizUtility[][]$. Entretanto, devido aos requisitos adicionais da síntese de serviços, algumas modificações foram realizadas sobre o algoritmo de emparelhamento de acordo com a extensão que permite prevenir sobrecarga sobre os serviços selecionados. O Código 7.2 mostra o esqueleto do algoritmo

de emparelhamento estendido. Na Linha 4 é definido o construtor da classe *Matching*, que recebe, além da matriz que representa o grafo bipartido com os valores da função de utilidade, informações sobre carga das coreografias e capacidade dos serviços. Com essas informações as estruturas de dados auxiliares no processo de busca por emparelhamento são inicializadas.

Código 7.2 Trecho da classe *Matching* que executa o algoritmo de emparelhamento estendido.

```
1  class Matching {
2      // definição das estruturas de dados
3
4      public Matching(Double[][] newMatrizUtility,
5                      Integer[] vetLoad,
6                      int quantOfService,
7                      Integer[] vetCapacity){
8          // inicializa as estruturas
9      }
10     public Integer[] execute() // busca pelo emparelhamento
11     {
12         reduce();
13         computeInitialSolution();
14         int c = fetchUnmatchedChor();
15         while (c < rows)
16         {
17             initializePhase(c);
18             if ( buscaCaminhoAumentante() ) {
19                 c = fetchUnmatchedChor();
20             }
21             else{
22                 // caminho m-aumentante inválido
23                 // marcar como já percorrido
24             }
25         }
26         Integer[] result = matchServiceByChor;
27         return result;
28     }
29     // demais métodos de apoio
30     ..
31 }
```

No algoritmo de emparelhamento estendido foram adicionadas estruturas de dados para armazenar informações de carga para cada coreografia que usa o papel em análise, representadas pelas linhas da matriz de utilidade. Também foram adicionadas informações de capacidade máxima e capacidade residual de cada serviço candidato, representadas pelas colunas da matriz de utilidade. Note-se que essas informações incluem os vértices virtuais. Como a capacidade de cada serviço candidato é compartilhada por seus vértices virtuais, também foi adicionada uma estrutura de dados para armazenar informações sobre o mapeamento entre os serviços e os seus vértices virtuais.

O método *execute*(), na Linha 10 do Código 7.2, é responsável por coordenar o algoritmo de emparelhamento estendido. Na Linha 12, o método *reduce*() busca decrementar o número de iterações do algoritmo ao definir arestas de custo zero no grafo da igualdade [93]. Na Linha 13, o método *computeInitialSolution*() calcula um emparelhamento guloso entre coreografias e serviços da matriz cujos custos são zero. No algoritmo estendido são adicionadas verificações sobre a capacidade dos serviços candidatos em relação à carga das coreografias. Já na Linha 14 é verificado se foi encontrado um emparelhamento que cobre todas coreografias. Caso seja verdade, o problema de emparelhamento é resolvido e as Linhas 26 e 27 serão executadas. Caso contrário, o método *fetchUnmatchedChor*() retorna uma coreografia que ainda não foi emparelhada, ou seja, um vértice exposto.

A Linha 15 inicia um laço para a busca por *caminhos m-aumentantes* a partir do vértice exposto *c*. Enquanto existir um vértice exposto na bipartição que representa as coreografias, as instruções entre as Linhas 16 à 24 são executadas. Na Linha 17 são calculados os pesos das arestas cuja origem é o vértice exposto *c* para permitir o início da busca por *caminhos m-aumentantes*.

Na Linha 18, o método *buscaCaminhoAumentante*() busca por *caminhos m-aumentantes* a partir de *c*. No algoritmo estendido foram adicionadas verificações sobre a capacidade dos serviços candidatos que compõem um *caminho m-aumentante*, em relação à carga das coreografias. Além disso, na busca por *caminhos m-aumentantes*, caso vários serviços possam ser alcançáveis a partir de um vértice representando uma coreografia, o algoritmo estendido prioriza os serviços com a menor capacidade residual em relação aos demais serviços, utilizando a heurística *Best-Fit*, adotada na resolução do *problema de bin-packing* [87], como discutido no Capítulo 6. Caso um *caminho m-aumentante* válido (em relação a capacidade dos serviços) for encontrado, a linha 19 é executada, ou seja, verifica-se se ainda existe alguma coreografia que não foi emparelhada.

No algoritmo estendido, caso um *caminho m-aumentante* não seja validado devido à sobrecarga sobre os serviços que compõem o caminho, esse caminho deve ser descartado. Para implementar essa característica, foram criadas estruturas de dados adicionais para armazenar cópias temporárias dos *caminhos m-aumentantes* descobertos. Nesse caso, apenas quando um *caminho m-aumentante* descoberto é validado, suas informações são persistidas nas estruturas de dados originais. Além disso, um *caminho m-aumentante* não validado é marcado como já percorrido. Na linha 26 o emparelhamento é armazenado no vetor *result*, que é retornado na linha 27. No algoritmo estendido as verificações adicionais lidam apenas com as estruturas de dados que armazenam informações sobre carga e capacidade, não sendo necessário fazer alteração na lógica do algoritmo de emparelhamento original.

A saída do método $emparelhamento(newMatrizUtility[][])$ (Linha 17 do Algoritmo 7.3) é um mapeamento que contém as informações sobre quais serviços foram selecionados para cada papel de cada coreografia. Na Linha 18 são adicionados os mapeamentos de cada papel, formando o modelo de mapeamento. Por fim, depois de sair do laço da Linha 19, o método $atualizaGDC(modeloMapeamento, gdc)$ é chamado para gerar o grafo de dependências sensível à QoS concreto, na Linha 20. Esse modelo mantém informações sobre os serviços selecionados para atender todas as coreografias submetidas para implantação.

7.5 Conclusão

Neste capítulo apresentamos uma arquitetura para a execução coordenada da abordagem proposta para seleção de serviços sensível à QoS e à capacidade para múltiplas coreografias. A arquitetura proposta seguiu a abordagem geral, apresentada no Capítulo 1, definindo as funcionalidades em cada camada, ou seja, desde a análise das especificações de coreografias de serviços submetidas até a seleção dos serviços mais adequados para cada papel. Para cada camada, também foi apresentada a discussão da implementação das principais funcionalidades.

Inicialmente, a camada de análise de coreografias recebe um grafo de processo com informações sobre requisitos de QoS e a carga estimada para uma coreografia. O primeiro passo foi decompor a carga estimada da coreografia para os papéis que a compõem. Em seguida, a fim de melhorar a flexibilidade no processo de seleção de serviços sensível à QoS, optamos por utilizar um método de decomposição da QoS global em QoS local para o caso onde o provedor descreve requisitos de QoS global para as coreografias. O objetivo é decompor os requisitos de QoS globais em um conjunto de requisitos de QoS locais que servirão como limites superiores e inferiores, de modo que a satisfação dos requisitos de QoS locais assegure a satisfação dos requisitos de QoS globais. Ao realizar esses dois passos, é anotada para cada papel a informação da carga e o requisito de QoS, gerando o grafo de processo sensível à QoS

Em seguida, os grafos de processo sensíveis à QoS são sintetizados em um grafo de dependências sensível à QoS abstrato, na camada de síntese de coreografias. Por sua vez, o grafo de dependências sensível à QoS abstrato junto com o modelo de descrição dos serviços são utilizados pela camada de seleção de serviços para realizar a síntese de serviços, cuja saída é o grafo de dependências sensível à QoS concreto. Este por sua vez, é gerado a partir do modelo de mapeamento de serviços, que mantém as informações sobre os serviços selecionados para cada coreografia, além de manter informações sobre o compartilhamento de serviços pelas coreografias.

A funcionalidade de cada camada foi projetada levando-se em consideração as características das coreografias de serviços. Diante desta visão, as camadas de análise de coreografias e síntese de coreografias lidam com papéis individuais. Em outras palavras, estas duas camadas buscam papéis comuns entre coreografias para unificar tais papéis em um único vértice no grafo de dependências, que serve de base para o processo de seleção de serviços.

Por fim, a abordagem proposta apresenta vantagens quando se lida com grupos de coreografias. Sendo assim, a seleção de serviços tem o melhor desempenho quando é possível agrupar um conjunto de coreografias de serviços, permitindo tirar proveito das informações das coreografias combinadas para realizar uma seleção de serviços eficiente. Conforme apresentado no próximo capítulo, o protótipo implementado foi usado para demonstrar a abordagem proposta e avaliá-la por meio de experimentos, através dos quais foram analisadas a qualidade da solução em relação aos objetivos e o tempo de processamento da seleção de serviços.

CAPÍTULO 8

Avaliação

No Capítulo 6 foi apresentada a abordagem proposta para realizar a seleção de serviços sensível à QoS e à capacidade para um conjunto de coreografias de serviços. Essa abordagem foi incluída em uma arquitetura, considerando as atividades da síntese de serviços, para realizar a seleção de serviços para múltiplas coreografias respeitando os requisitos de QoS e carga. O funcionamento da síntese de serviços se baseia nos tratamentos sugeridos para a modelagem conjunta de coreografias e constitui a principal contribuição desta tese.

Diante disso, com o objetivo de avaliar os resultados obtidos pela abordagem desenvolvida, realizamos um conjunto de experimentos que são descritos neste capítulo. Em particular, esta tese está interessada em dois tipos de avaliação: avaliação da qualidade da síntese de serviços e avaliação do desempenho da síntese de serviços.

A Seção 8.1 avalia a qualidade da síntese de serviços com relação às atividades realizadas durante a seleção de serviços, ou seja, garantir que todos os requisitos de QoS especificados sejam atendidos, ao mesmo tempo que a seleção de serviços é realizada de forma eficiente. Já na Seção 8.2, o desempenho da síntese de serviços é avaliado, uma vez que é necessário garantir que os resultados da síntese de serviços sejam obtidos rapidamente, não causando sobrecarga excessiva sobre o processo de implantação das coreografias. A Seção 8.3 discute as principais limitações dos experimentos. Por fim, a Seção 8.4 apresenta as conclusões sobre os experimentos.

Os experimentos foram projetados e descritos seguindo a estrutura proposta por Wohlin *et al.* [135] para a apresentação de experimentos: definição do experimento, métricas de avaliação e análise e interpretação dos resultados.

8.1 Avaliação da síntese de serviços

A qualidade da síntese de serviços é em função da capacidade da abordagem em selecionar um conjunto de serviços que garanta os requisitos dos provedores de coreografias, ao mesmo tempo que minimiza os custos relacionados aos recursos utilizados pelos serviços selecionados.

Ao realizar a síntese de serviços, a seleção de serviços utiliza a estratégia *best-fit* para avaliar a QoS dos requisitos das coreografias e a oferta de QoS dos serviços candidatos. Esta estratégia tem como objetivo a redução do desperdício de recursos ao selecionar serviços de acordo com a distância entre a QoS requisitada e a QoS ofertada para todas as coreografias de serviços.

Além disso, a síntese de serviços busca maximizar a taxa de utilização dos serviços selecionados ao aplicar extensões no algoritmo de emparelhamento. Ao aplicar essas duas estratégias, durante a síntese de serviços, buscamos minimizar os custos dos recursos associados pelos serviços selecionados.

8.1.1 Definição dos experimentos

Para a proposta da nossa avaliação é necessário considerar um conjunto de n coreografias compostas por m papéis. As coreografias possuem um subconjunto de papéis em comum e a cada coreografia de serviços está associada uma QoS requisitada e carga estimada. Os serviços candidatos devem ser agrupados em tipos de serviços, ou seja, um tipo de serviço possui um conjunto de serviços candidatos que oferecem uma funcionalidade. Sem perda de generalidade, cada papel é mapeado para um único tipo de serviço como adotado em [153] e [139]. Para cada serviço candidato devem ser definidas a oferta de QoS e a capacidade dos serviços.

Os *datasets* de serviços, apresentados em [3] e [154], possuem uma quantidade limitada de informações e os serviços não são organizados por tipos de serviços, o que inviabiliza uma comparação entre serviços candidatos para um mesmo papel. Por este motivo, esta tese optou por gerar sinteticamente as informações sobre oferta de QoS de serviços, como em [8, 83, 5, 96].

A Figura 8.1 ilustra os parâmetros do experimento que exploram seis dimensões, onde os números entre parênteses especificam o valor ou intervalo de valores para cada dimensão. Cada parâmetro é definido a seguir:

1. Número de coreografias de serviços para serem implantadas;
2. Tamanho das coreografias de serviços: número de papéis que compõem uma coreografia;
3. Número de definições de papéis existentes: quantidade de papéis disponíveis que podem ser utilizados para especificar uma coreografia;
4. Número de serviços candidatos por papel: quantidade de serviços candidatos para cada papel que são utilizados no processo de seleção de serviços;
5. quantidade de papéis em comum entre coreografias; e
6. atributos de QoS utilizados na oferta e requisito de QoS.

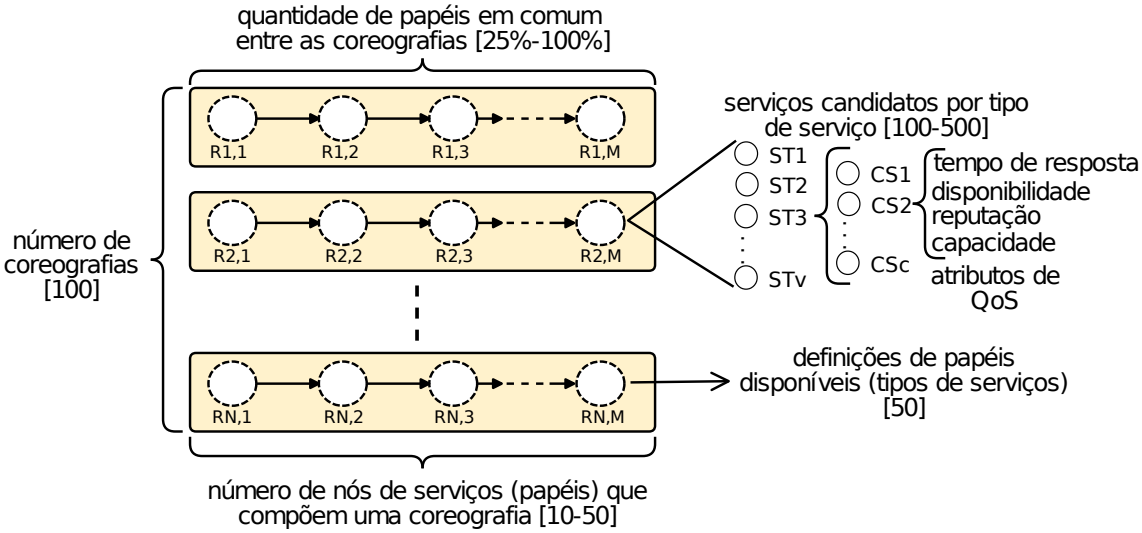


Figura 8.1: *Desenho do experimento.*

Em relação à oferta de QoS dos serviços candidatos, foi utilizado um gerador de oferta de QoS sintético, onde os atributos de QoS utilizados são tempo de resposta, disponibilidade e reputação, além da capacidade. Os valores destes atributos de QoS são gerados de acordo com a distribuição usada em outros trabalhos na literatura:

- *Disponibilidade*: os valores de disponibilidade são gerados assumindo-se uma distribuição uniforme no intervalo entre 0.95 a 0.9999, como em [8] e [7].
- *Reputação*: os valores da reputação são gerados assumindo-se uma distribuição uniforme no intervalo entre 0.8 a 0.99, como em [8].
- *Tempo de resposta*: como adotado em [8] e [96], é assumido que o tempo de resposta tem uma distribuição Gaussiana cujo centro é η_{tr} e o desvio padrão é δ_{tr}. Os valores de η_{tr} e δ_{tr} são gerados de acordo com os valores mais representativos fornecidos pelo *dataset QWS* [3], como adotado em [56] e [129]: $\eta_{tr} = 290$ milissegundos e $\delta_{tr} = 246$ milissegundos.
- *Capacidade*: a capacidade também segue a distribuição Gaussiana cujo centro é η_{cap} e o desvio padrão é δ_{cap}. Os valores de η_{cap} e δ_{cap} são gerados de acordo com os valores mais representativos fornecidos pelo *dataset QWS* [3]: $\eta_{cap} = 18$ req/seg e $\delta_{cap} = 8$ req/seg.

Para cada tipo de serviço foram gerados entre 100 e 500 serviços candidatos, tomando como base os experimentos em [7], [6] e [83]. Para especificar as coreografias de serviços, foi desenvolvido um gerador de coreografias de serviços sintéticos que gera coreografias de serviços abstratos, considerando um cenário com 100 coreografias que possuem entre 10 a 50 papéis, como usado nos experimentos de [7] e [6]. Neste experimento, foi utilizado o padrão sequencial para composição dos papéis, como em [4]. Outros modelos podem ser reduzidos ou transformados para o mo-

delo sequencial, aplicando-se técnicas para lidar com múltiplos caminhos de execução e laços como em [25].

Os experimentos utilizam papéis em comum entre coreografias, com o conjunto de papéis em comum variando de 25%, 50%, 75% ou 100%, seguindo parâmetro usado em [52]. Por exemplo, quando a quantidade de papéis em comum é 50%, significa que cada papel utilizado é comum em metade das coreografias de serviços. Note-se que este parâmetro indica apenas a probabilidade de ter um número específico de serviços em comum entre um conjunto de coreografias de serviços. Os valores dos requisitos de QoS das coreografias de serviços foram gerados aleatoriamente, levando-se em consideração uma distribuição Gaussiana em relação aos valores médios de cada atributo de QoS dos serviços candidatos para determinado papel, como em [84].

8.1.2 Métricas de avaliação

A principal métrica utilizada para medir a qualidade dos resultados obtidos pela síntese de serviços é o custo dos serviços selecionados para satisfazerem os requisitos dos provedores de coreografias. O custo de utilização de um serviço é baseado na QoS oferecida e na quantidade de requisições que um serviço consegue atender em um dado período de tempo, conforme definido na Seção 4.4. Sendo assim, o custo é definido pela relação entre as seguintes métricas:

1. **Utilidade da QoS agregada**: essa métrica é calculada sobre os atributos de QoS, tanto para a QoS solicitada por uma coreografia de serviços quanto para a QoS oferecida pelos serviços candidatos selecionados. Quanto maior a utilidade da QoS agregada oferecida, maior o seu impacto no custo do serviço selecionado em relação ao recurso utilizado para executá-lo. Além disso, quanto maior a distância entre a utilidade da QoS agregada requisitada e a utilidade da QoS agregada oferecida, maior o desperdício dos recursos.
2. **Capacidade contratada**: a capacidade contratada é a capacidade máxima oferecida por um serviço candidato selecionado capaz de garantir a QoS requisitada e carga esperada por um papel de uma coreografia de serviços. A capacidade contratada de cada serviço selecionado tem impacto direto no custo para utilização de um serviço, uma vez que o custo é baseado na QoS oferecida e na quantidade de requisições que um serviço consegue atender em um dado período de tempo. Além disso, essa variável permite analisar a taxa de utilização dos serviços, que também reflete na efetiva utilização dos recursos alocados.

Essas métricas são influenciadas pela quantidade de papéis comuns entre múltiplas coreografias de serviços que devem ser implantadas diante da variação do

número de serviços candidatos disponíveis, uma vez que elas concorrem pelo mesmo conjunto de serviços candidatos.

Para definir a utilidade da QoS agregada requisitada por coreografia de serviços, é utilizada a soma dos valores requisitados para todos os atributos de QoS dos papéis que compõem uma coreografia, como mostra a Equação 8-1. Por sua vez, a utilidade da QoS agregada oferecida por coreografia de serviços é calculada utilizando a soma dos valores de QoS agregada de todos os serviços selecionados por coreografia, como mostra a Equação 8-2.

$$\psi_{req} = \sum_{h=1}^{p} \mu_{req_h}, \ \ onde \ \ \mu_{req_h} = \sum_{j=1}^{k} r'_{h,j} \times \rho_j; \tag{8-1}$$

$$\psi_{ofer} = \sum_{h=1}^{p} \mu_{ofer_h}, \ \ onde \ \ \mu_{ofer_h} = \sum_{j=1}^{k} q'_{i,j} \times \rho_j; \tag{8-2}$$

onde μ_{req_h} representa o valor da utilidade da QoS agregada requisitada pelo h-ésimo papel e μ_{ofer_h} representa o valor da utilidade da QoS agregada ofertada pelo serviço selecionado para satisfazer o h-ésimo papel. $r'_{h,j}$ representa o valor normalizado para o j-ésimo atributo de QoS do h-ésimo papel de uma dada coreografia, $q'_{i,j}$ representa o valor normalizado para o j-ésimo atributo de QoS do serviço selecionado s_i e ρ_j é o peso do j-ésimo atributo de QoS dentre os k atributos. Por fim, p denota o número de papéis presentes em uma coreografia de serviços.

Para facilitar a análise dos dados, também são calculadas a utilidade da QoS agregada média requisitada por papel de cada coreografia de serviços (Equação 8-3) e a utilidade da QoS agregada média dos serviços contratados para todas as coreografias submetidas para implantação (Equação 8-4):

$$\Psi_{req} = \frac{\sum_{g=1}^{z} \frac{\psi_{req_g}}{p_g}}{z}; \tag{8-3}$$

$$\Psi_{ofer} = \frac{\sum_{g=1}^{z} \frac{\psi_{ofer_g}}{b_g}}{z}; \tag{8-4}$$

onde z é o número de coreografias de serviços no ambiente, p_g é o número de papéis da g-ésima coreografia de serviços e b_g é o número de serviços selecionados pela g-ésima coreografia. A utilidade da QoS agregada média requisitada por papel de cada coreografia de serviços e a utilidade da QoS agregada média dos serviços selecionados são calculadas utilizando-se os valores das métricas de QoS normalizados para eliminar a diferença de escala entre as várias métricas de QoS.

A Equação 8-5 determina a capacidade contratada Π, calculada como a soma da capacidade dos serviços contratados por todas as coreografias submetidas para

implantação:

$$\Pi = \sum_{i=1}^{b} cap_i; \tag{8-5}$$

onde b é o número de serviços contratados e cap_i é o valor da capacidade contratada do i-ésimo serviço selecionado. A taxa de utilização, calculada pela Equação 6-2 (Capítulo 6), mede a qualidade de uma abordagem de seleção de serviços em função da eficiência na utilização dos serviços contratados.

Com a utilidade da QoS agregada e a capacidade contratada, o custo dos serviços selecionados para satisfazer os requisitos dos provedores de coreografias pode ser definido. O custo é definido pela Equação 4-8, onde o custo de um serviço é baseado na utilidade da QoS ofertada, representado pela variável μ_{ofer}, e a capacidade máxima contratada junto ao provedor de serviços, representada pela variável cap_i.

Para o propósito desta avaliação, optamos por comparar a abordagem proposta nesta tese com outras abordagens para seleção de serviços encontradas na literatura. Como discutido no Capítulo 3, existem várias abordagens similares à abordagem proposta nesta tese. Dentre os trabalhos similares, estamos interessados em abordagens de seleção de serviços sensível à QoS e à capacidade para múltiplas coreografias com diferentes valores de requisições de QoS e carga para cada coreografia. Dentre os trabalhos que tratam desses aspectos, as estratégias de avaliação de QoS, *best-QoS* e a *best-fit*, foram utilizadas como base para comparação, uma vez que a análise da QoS têm impacto direto no custo dos serviços selecionados para satisfazer os requisitos dos provedores de coreografias. Diante disso, comparamos a abordagem proposta nesta tese, denominada *SSMCP*, com outras duas abordagens para a seleção de serviços encontradas na literatura:

- ***Abordagem BF 1***: proposta por Wu *et al.* [139], essa abordagem realiza a seleção de serviços utilizando a estratégia de avaliação *best-fit* (BF) com sensibilidade à capacidade. Esta abordagem seleciona serviços para várias composições de serviços, embora um grupo de composições sejam processadas em sequência. Este trabalho, ao utilizar a estratégia *best-fit*, permite selecionar serviços que atendam as requisições sem levar ao desperdício de recursos, além de aumentar a disponibilidade dos serviços candidatos disponíveis. Este trabalho foi escolhido porque dentre os trabalhos que utilizam a estratégia *best-fit*, ele é o único que realiza a seleção de serviços para várias composições.
- ***Abordagem BQ 2***: proposta por Ardagna e Mirandola [7], essa abordagem realiza a seleção de serviços utilizando a estratégia de avaliação *best-QoS* (BQ) com sensibilidade à capacidade. A abordagem considera múltiplas composições de serviços no mesmo ambiente, mas processa apenas uma composição de serviços

por vez utilizando uma estratégia de ordenação. Ao utilizar a estratégia *best-QoS*, essa abordagem busca aumentar a garantia da QoS requisitada pelos provedores de coreografias. Para não levar à subutilização dos serviços selecionados ao empregar a estratégia *best-QoS*, os autores realizam a seleção de serviços levando em consideração um fluxo de requisições dos usuários sobre diferentes composições que possuem um conjunto de papéis em comum, aumentando assim a taxa de utilização dos serviços. Embora, existam outros trabalhos que utilizam a estratégia *best-QoS* para realizar a seleção de serviços para múltiplas composições, a abordagem proposta por Ardagna e Mirandola [7] é a que possui os objetivos mais próximos desta tese.

A abordagem *SSMCP*, proposta nesta tese, também realiza a seleção de serviços utilizando a estratégia *best-fit* de avaliação de QoS com sensibilidade à capacidade, mas ela processa um conjunto de coreografias de serviços ao mesmo tempo, buscando diminuir os custos dos serviços selecionados, ao mesmo tempo que garante a QoS requisitada.

8.1.3 Análise e interpretação dos resultados

As execuções da síntese de serviços dos experimentos foram realizadas em uma máquina com a seguinte configuração: CPU Intel® Core™ i5 2.67GHz, 6GB RAM, Ubuntu 14.04.

Na *Abordagem BF 1* e na *Abordagem BQ 2* as múltiplas coreografias são processadas sequencialmente, selecionadas aleatoriamente para serem processadas, enquanto que a abordagem *SSMCP* processa um conjunto de coreografias de serviços combinadas. Depois de cada abordagem realizar a seleção de serviços, são geradas as coreografias com serviços concretos, substituindo os respectivos papéis. Então, uma vez selecionados os serviços, calculamos o valor da utilidade da QoS agregada e capacidade contratada.

Como dito anteriormente, utilidade da QoS agregada e capacidade contratada são as duas principais métricas analisadas na seleção de serviços para as coreografias utilizadas nestes experimentos. De maneira geral, a quantidade de papéis em comum entre as coreografias de serviços e o número de serviços candidatos por papel são os parâmetros que impactam nos resultados para utilidade da QoS agregada e capacidade contratada. Em virtude disso, para facilitar a apresentação, os resultados para utilidade da QoS agregada e capacidade contratada para as diferentes abordagens são discutidos analisando-se quantidade de papéis em comum entre as coreografias de serviços e considerando-se a variação do número de serviços candidatos por papel. Por fim, uma análise do custo em relação aos resultados de ambas métricas é apresentada.

O gráfico da Figura 8.2 apresenta os resultados para a utilidade da QoS agregada média de todos os serviços selecionados para atender ao conjunto de coreografias submetidas para cada abordagem, considerando a quantidade de papéis em comum entre as coreografias. Para interpretar esses resultados é importante ressaltar que a utilidade da QoS agregada de um serviço selecionado varia entre 0 e 3, uma vez que são utilizados três atributos de QoS com valores normalizados. Neste experimento, uma vez que a QoS e a carga requisitadas por cada coreografia são iguais para todas as abordagens, quanto menor o valor da utilidade da QoS agregada, melhor o resultado, pois significa que menos recursos foram utilizados para executar os serviços, ao mesmo tempo que os requisitos definidos para as coreografias foram satisfeitos.

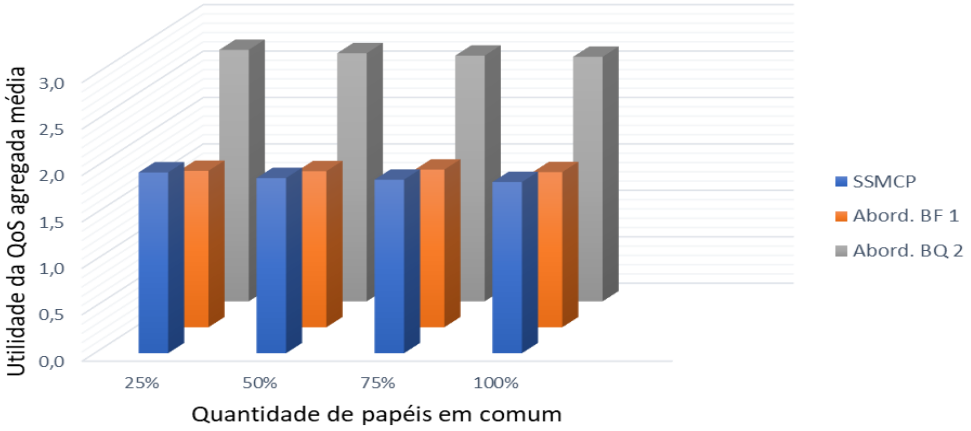


Figura 8.2: *Utilidade da QoS agregada média de todos os serviços selecionados em relação a quantidade de papéis em comum entre as coreografias considerando cada abordagem.*

Como pode ser visto no gráfico, *SSMCP* e *Abordagem BF 1* apresentam valores mais baixos em relação à *Abordagem BQ 2* para todas as quantidades de papéis em comum. Na *SSMCP* e na *Abordagem BF 1*, ao utilizarem a estratégia de avaliação de QoS *best-fit*, os serviços selecionados são avaliados de acordo com a oferta de QoS mais próxima dos requisitos de QoS dos provedores de coreografias, o que leva a valores mais baixos para a utilidade da QoS agregada média, conforme esperado. Tal fato, leva à diminuição do custo dos serviços em relação aos recursos associados, quando analisado apenas a oferta de QoS pelos serviços selecionados. Enquanto isso, na *Abordagem BQ 2*, os serviços selecionados são os que possuem os maiores valores para os atributos de QoS, mesmo que os requisitos de QoS dos provedores de coreografias não demandem tanto.

Em uma outra visualização desses resultados, os dois gráficos da Figura 8.3

mostram a utilidade da QoS agregada média de todos os serviços selecionados em relação à quantidade de papéis em comum (gráfico (a): *25%*; e gráfico (b): *100%*).

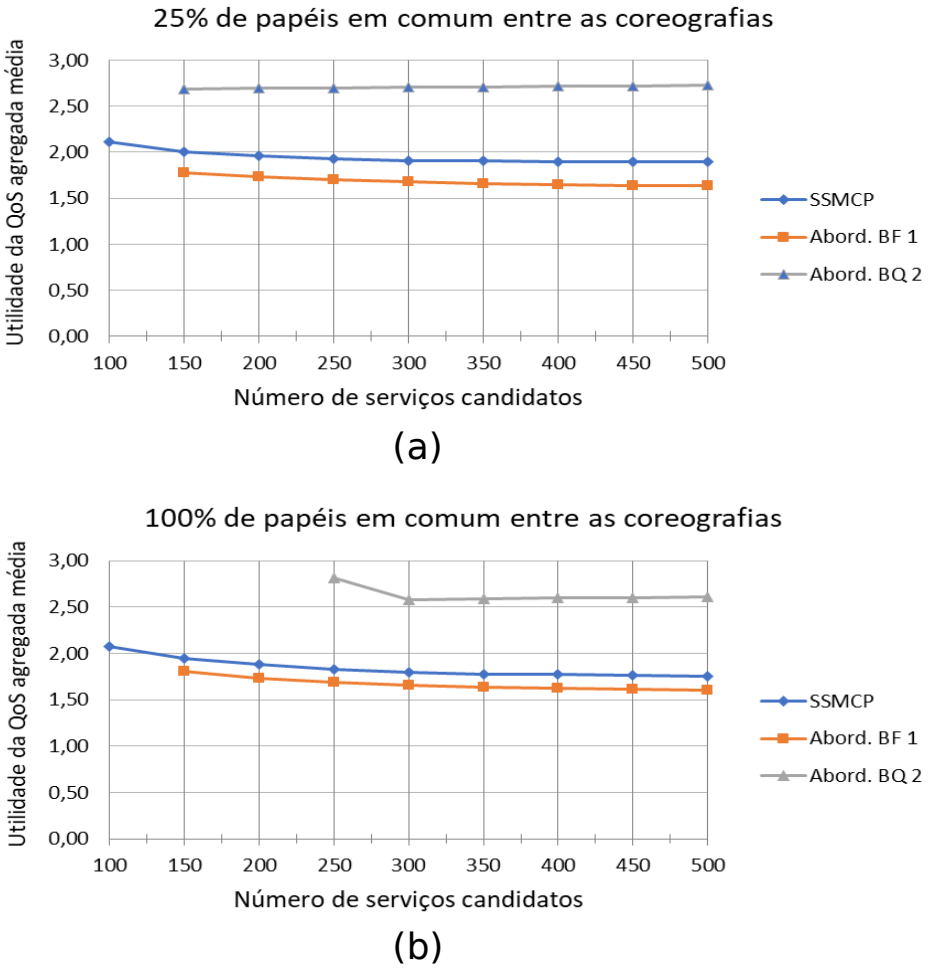


Figura 8.3: *Utilidade da QoS agregada média de todos os serviços selecionados em relação ao número de serviços candidatos por papel para cada abordagem, considerando diferentes valores para os papéis em comum entre as coreografias: a)* 25% *de papéis em comum; b)* 100% *de papéis em comum.*

Como pode ser visto pelos gráficos da Figura 8.3, em ambos casos as abordagens tendem a manter valores bem próximos da utilidade da QoS agregada média mesmo com o aumento do número de serviços candidatos. Este fato é importante porque a quantidade de papéis em comum entre as coreografias de serviços tem relação direta com a quantidade de serviços candidatos por papel: quanto maior a quantidade de papéis em comum entre as coreografias de serviços, necessita-se de uma maior quantidade de serviços candidatos por papel para conseguir implantar todas as coreografias de serviços.

A ausência de valores nos gráficos da Figura 8.3 indica situações em que uma abordagem não consegue implantar todas as coreografias de serviços, como o que

ocorreu para a *Abordagem BF 1* e a *Abordagem BQ 2*. Com a quantidade papéis em comum com *25%*, ambas abordagens não conseguiram implantar todas as coreografias com *100* serviços candidatos por papel. Já a quantidade de papéis em comum com *100%*, a *Abordagem BF 1* também não conseguiu implantar todas as coreografias com *100* serviços candidatos, enquanto a *Abordagem BQ 2* somente conseguiu implantar todas as coreografias com *250* serviços candidatos.

A Tabela 8.1 mostra os valores da capacidade máxima contratada pelos serviços selecionados com 25% e 100% de papéis em comum entre as coreografias de serviços para cada abordagem em relação ao incremento do número de serviços candidatos por papel.

	SSMCP	Abord. BF 1	Abord. BQ 2
100	9964	-	-
150	10151	14274	8110
200	10302	15126	7993
250	10492	15814	7946
300	10716	16246	7841
350	10659	16534	7864
400	10551	16992	7867
450	10492	17257	7846
500	10458	17585	7895

	SSMCP	Abord. BF 1	Abord. BQ 2
100	9300	-	-
150	9370	9896	-
200	9671	10572	-
250	10004	11105	7773
300	10106	11616	7818
350	10137	11815	7779
400	10171	12039	7767
450	10117	12519	7794
500	10122	12728	7821

Tabela 8.1: *Capacidade máxima contratada pelos serviços selecionados em relação ao número de serviços candidatos por papel. A tabela da esquerda representa o custo com quantidade de papéis em comum com 25% e a tabela da direita representa o custo com quantidade de papéis em comum com 100%.*

Conforme esperado, a *Abordagem BQ 2* possui o valor mais baixo para essa variável, enquanto a *Abordagem BF 1* possui valor mais alto. Por exemplo, dados 500 serviços candidatos por papel com 25% de papéis em comum, os serviços selecionados pelas 100 coreografias na *Abordagem BQ 2* contratam um valor de *7895* para a capacidade máxima contratada, enquanto na *Abordagem BF 1* esse valor chega a *17585*. Vale ressaltar que a carga requisitada para todas as abordagens é a mesma, logo, quanto maior o valor da capacidade contratada, maior é o desperdício.

A escolha da estratégia de avaliação da QoS dos serviços candidatos em cada abordagem tem influência direta na capacidade contratada de cada serviço selecionado. A *Abordagem BQ 2*, ao buscar selecionar os serviços com a melhor QoS, tende a aumentar o compartilhamento dos serviços entre requisições de diferentes coreografias de serviços sobre um mesmo papel. Enquanto isso, a *SSMCP* e a *Abordagem BF 1* tendem a aumentar a distribuição dos serviços selecionados entre as coreografias. Isso reflete na capacidade contratada junto aos serviços pelas coreografias. Entretanto, a

SSMCP, mesmo empregando a estratégia *best-fit*, consegue valores mais baixos para a capacidade máxima contratada do que a *Abordagem BF 1* devido ao uso da heurística que busca aumentar a taxa de utilização dos serviços candidatos selecionados.

Para uma melhor análise do uso dos recursos, os gráficos da Figura 8.4 mostram a taxa de utilização média dos serviços selecionados em relação ao número de serviços candidatos, considerando diferentes valores para a quantidade de papéis em comum para cada uma das abordagens. A taxa de utilização é baseada na capacidade máxima de processamento de um serviço selecionado e na soma das cargas das coreografias de serviços atendidas por um serviço contratado.

De acordo com os gráficos da Figura 8.4, a *Abordagem BQ 2* possui a taxa de utilização mais alta, que incrementa à medida em que aumenta a quantidade de papéis em comum entre as coreografias de serviços. Isso acontece porque, ao aumentar a quantidade de papéis em comum, também ocorre o aumento do número de coreografias que requisitam um mesmo subconjunto de serviços candidatos. Logo, os serviços desse subconjunto são selecionados enquanto a capacidade máxima de cada serviço conseguir atender a carga agregada. Como pode ser observado no gráfico, com a quantidade de papéis em comum em *75%* e *100%*, a taxa de utilização fica acima de *90%*.

A *Abordagem BF 1* possui a pior taxa de utilização dentre as abordagens analisadas. Além disso, o resultado é pior quando a quantidade de papéis em comum entre as coreografias de serviços é mais baixa, juntamente com o incremento do número de serviços candidatos por papel. Isso ocorre porque na *Abordagem BF 1* quanto maior o número de serviços candidatos disponíveis maior também é a distribuição dos serviços selecionados entre as coreografias de serviços. Com a quantidade de papéis em comum em *25%* e com o incremento do número de serviços candidatos, a taxa de utilização fica abaixo de *40%*.

A *SSMCP* possui uma taxa de utilização intermediária em comparação com as outras abordagens, onde a melhor taxa de utilização ocorre com a maior quantidade papéis em comum juntamente com um número baixo de serviços candidatos por papel. Isso ocorre porque com o aumento do número de papéis em comum e um conjunto menor de serviços candidatos, aumenta a efetividade da abordagem no aumento da taxa de utilização dos serviços com o emprego da extensão proposta ao algoritmo húngaro.

Como esperado, os resultados mostram que a utilidade da QoS agregada e capacidade dos serviços selecionados são conflitantes. A *Abordagem BF 1* possui os melhores valores em relação à utilidade da QoS agregada dos serviços selecionados, o que impacta positivamente no custo dos serviços. Ao mesmo tempo, possui valores elevados para capacidade contratada, refletindo na pior taxa de utilização dos serviços

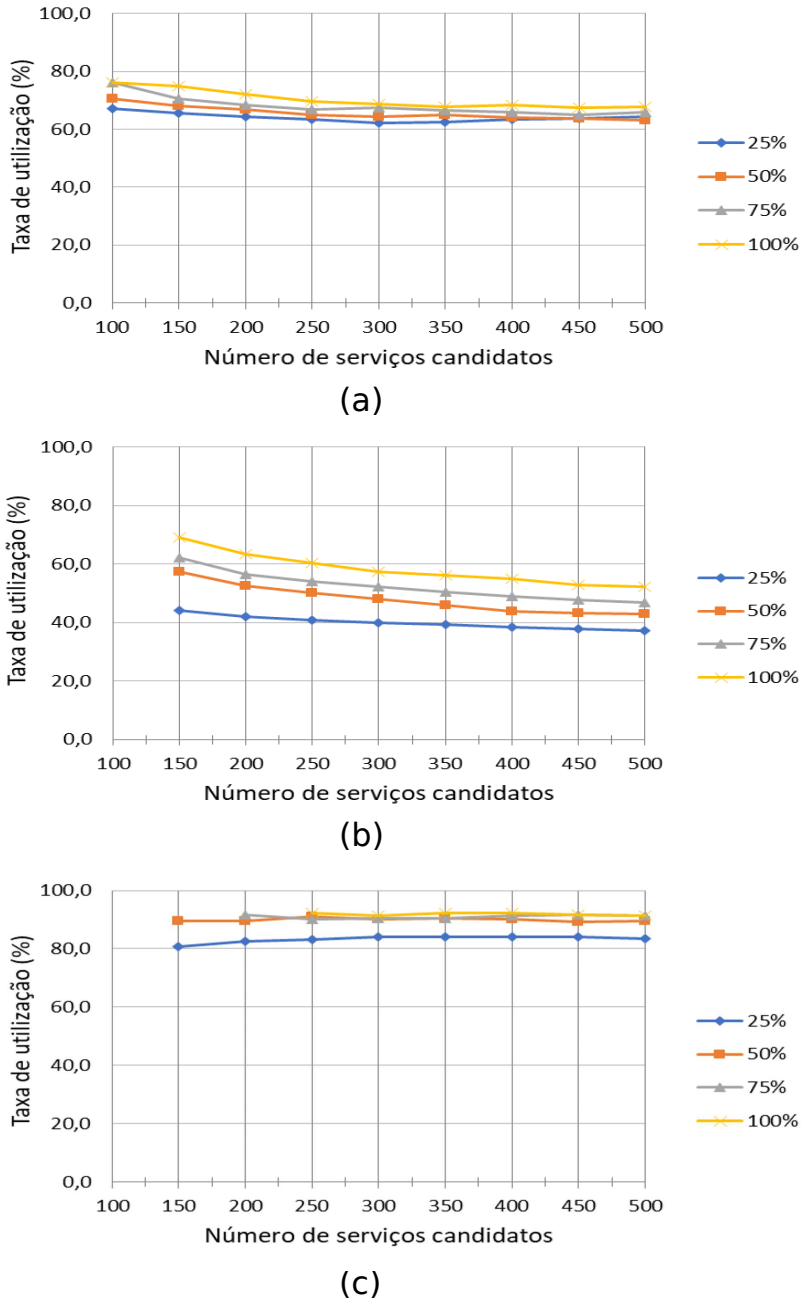


Figura 8.4: *Taxa de utilização dos serviços selecionados por serviços candidatos considerando diferentes quantidades de papéis em comum para cada abordagem: a)* SSMCP*; b)* Abordagem BF 1*; c)* Abordagem BQ 2.

selecionados, o que impacta negativamente no custo dos serviços. Por outro lado, a *Abordagem BQ 2*, que possui valores mais altos em relação a utilidade da QoS agregada dos serviços selecionados, impactando negativamente o custo dos serviços, possui os menores valores para capacidade contratada, refletindo na melhor taxa de utilização dos serviços selecionados, impactando positivamente no custo dos serviços. Já *SSMCP* possui valores para a utilidade da QoS agregada dos serviços selecionados próximos

da *Abordagem BF 1*, enquanto os valores da capacidade contratada são próximos da *Abordagem BQ 2*.

De forma a facilitar a comparação entre as três abordagens que possuem valores conflitantes entre utilidade de QoS e capacidade dos serviços contratados, optamos por utilizar uma função de custo que engloba ambas as métricas. A Equação 4-8 mostra a função de custo baseada na utilidade de QoS agregada e na capacidade contratada por serviço. Para facilitar a geração dos resultados, o preço, baseado na utilidade de QoS agregada, varia entre $R\$\ 0,01$ e $R\$\ 0,96$. Esse preço é baseado em modelos adotados por provedores de serviços, onde o serviço com a pior QoS terá o preço de $R\$\ 0,01$ e o serviço com a melhor QoS terá o preço de $R\$\ 0,96$. Para calcular o preço de cada serviço, esse valor, baseado na utilidade de QoS agregada, deve ser multiplicado pela capacidade do serviço. Por simplicidade, ignoramos informações sobre o tempo de utilização dos serviços, uma vez que consideramos que todas as coreografias de serviços iniciam e terminam no mesmo instante de tempo.

A Tabela 8.2 mostra os valores do custo dos serviços contratados para implantar todas as coreografias em relação ao número de serviços candidatos por papel, considerando a quantidade de papéis em comum com 25% e 100% para cada abordagem. Como esperado, baseado nos resultados individuais das variáveis de utilidade de QoS agregada e capacidade dos serviços contratados, a *SSMCP* possui o melhor custo, uma vez que ela consegue o melhor equilíbrio entre ambas variáveis. Nesse caso, podemos concluir que *SSMCP* tem o menor desperdício de recursos alocados para execução dos serviços selecionados, com aumento da quantidade papéis em comum.

	SSMCP	Abord. BF 1	Abord. BQ 2
100	597,8	-	-
150	558,3	713,7	770,4
200	566,6	756,3	799,3
250	577,1	790,7	794,6
300	589,3	731,1	784,1
350	586,2	744,1	786,4
400	580,3	764,6	786,7
450	577,1	776,5	784,6
500	575,2	791,3	789,5

	SSMCP	Abord. BF 1	Abord. BQ 2
100	604,5	-	-
150	552,2	564,2	-
200	531,9	581,4	-
250	530,2	565,2	738,4
300	505,3	528,7	742,7
350	506,8	531,6	739,0
400	503,5	541,7	737,8
450	510,8	563,3	740,4
500	506,1	572,7	742,9

Tabela 8.2: *Custo total dos serviços contratados em relação ao número de serviços candidatos por papel. A tabela da esquerda representa o custo com quantidade de papéis em comum em 25% e a tabela da direita representa o custo com quantidade de papéis em comum em 100%.*

A *Abordagem BF 1* possui um custo mais elevado do que a *SSMCP* devido à baixa taxa de utilização dos serviços, ou seja, ela possui um alto valor da capacidade

contratada. Como o custo de um serviço é baseado na QoS oferecida e na quantidade de requisições que ele consegue atender por período de tempo, um serviço com capacidade residual elevada possui uma alta taxa de desperdício de recursos. Já a *Abordagem BQ 2* possui o pior custo dentre as abordagens discutidas, mesmo conseguindo a melhor taxa de utilização dos serviços selecionados. Isso ocorre devido ao alto valor da utilidade de QoS agregada dos serviços selecionados, o que eleva o custo dos recursos necessários para executar os serviços.

A *SSMCP* e a *Abordagem BF 1* apresentam melhor custo com o aumento da quantidade de papéis em comum. Isso ocorre devido à diminuição da capacidade contratada com o aumento do compartilhamento de serviços entre as coreografias de serviços, como representado pelos gráficos da Figura 8.4. Na *Abordagem BQ 2* o custo tem pouca variação para as diferentes quantidades de papéis em comum.

Além de ter o melhor custo, a *SSMCP* também consegue ter a melhor taxa de implantação das coreografias de serviços quando o número de serviços candidatos é baixo. A Tabela 8.3 mostra a porcentagem de coreografias de serviços implantadas com sucesso usando as três abordagens, variando a quantidade de serviços candidatos por papel e com quantidade de papéis em comum em *100%*.

Tabela 8.3: *Número de coreografias de serviços que foram implantadas usando cada uma das três abordagens, considerando um total de 100 coreografias de serviços submetidas.*

Número de Serviços Candidatos	Abordagem BF 1	Abordagem BQ 2	SSMCP
100	81.6%	63.7%	100%
150	100.0%	77.5%	100%
200	100.0%	92.1%	100%
250	100.0%	100.0%	100%

No experimento, a *Abordagem BQ 2* não pôde implantar todas as coreografias de serviços porque tal abordagem busca selecionar primeiro os serviços com os valores dos atributos de QoS mais altos para cada coreografia de serviços, na ordem de submissão, independente do requisitos de QoS. Assim, se as primeiras coreografias de serviços a serem processadas tiverem valores mais baixos para os atributos de QoS requisitados, a abordagem seleciona serviços com a melhor QoS, causando o uso ineficiente de recursos em relação aos serviços selecionados. Como resultado, se as coreografias de serviços processadas no final da sequência de submissão tiverem valores mais altos para os atributos de QoS, pode haver apenas serviços candidatos com valores de QoS mais baixos, o que resulta na impossibilidade de implantar tais coreografias de serviços. Apenas com o aumento para *250* serviços candidatos por papel é que

a *Abordagem BQ 2* conseguiu implantar todas as coreografias de serviços. Portanto, a *Abordagem BQ 2* não garante que os serviços com a melhor QoS sejam reservados para solicitações que tenham requisitos de QoS elevados.

Já a *Abordagem BF 1* não pôde implantar todas as coreografias com *100* serviços candidatos por papel. Apenas com incremento do número de serviços candidatos todas as coreografias foram implantadas. Isso ocorre porque essa abordagem também favorece coreografias processadas no início da sequência, em detrimentos das coreografias de serviços processadas no final. Em outras palavras, com o aumento do número de coreografias, as que são processadas no final não podem ser implantadas devido à indisponibilidade de serviços adequados.

Em contraste, *SSMCP* atenua o problema de ordenação das coreografias de serviços submetidas a serem processadas, tendo uma visão global para otimizar a seleção de serviços. De acordo com a Tabela 8.3, a abordagem *SSMCP* foi capaz de implantar todas as coreografias apresentadas em todos os cenários de serviços candidatos. Isso se deve a dois fatores no processo de seleção: 1) a abordagem realiza uma análise de todos os serviços das múltiplas coreografias que competem pelo mesmo conjunto de serviços candidatos; 2) a abordagem busca o melhor ajuste entre a QoS requisitada e a QoS oferecida pelos serviços, reservando serviços com a melhor QoS para coreografias de serviços mais exigentes.

8.2 Tempo de execução da síntese de serviços

A análise do tempo de execução da síntese de serviços avalia a eficiência com que a *SSMCP* resolve o problema de seleção de serviços em relação ao número de coreografias de serviços, números de papéis por coreografia e o número de serviços candidatos por papéis. O tempo de execução é o tempo necessário para executar o algoritmo de síntese de serviços e, portanto, decidir quais serão os serviços candidatos alocados para cada papel de cada coreografia de serviços.

Como forma de avaliar esse aspecto e comprovar a factibilidade da abordagem proposta nesta tese, realizamos o experimento descrito nessa seção.

8.2.1 Definição dos experimentos

O objetivo deste experimento é medir o tempo necessário para obter o resultado da síntese de serviços. Essa característica é importante porque a síntese de serviços deve ser realizada em tempo de execução para lidar com violações do acordo entre o provedor de coreografias e provedor de serviços quando, por exemplo, ocorre indisponibilidade de um serviço previamente alocado ou ocorre variação para além da

carga estimada. Dessa forma, quanto mais tempo for necessário para solucionar a violação, maior será o período em que os serviços são utilizados com qualidade aquém da necessária, acarretando em penalidades e insatisfação dos usuários.

Para analisar este aspecto, o foco do experimento é verificar se o tempo necessário para obter a seleção de serviços para múltiplas coreografias de serviços é aceitável, considerando os contextos em que a síntese de serviços é utilizada.

8.2.2 Métricas de avaliação

A principal métrica analisada nesse experimento é o **tempo de processamento da síntese de serviços**. Na medição dessa métrica, consideramos o tempo necessário para realizar a seleção de serviços ao receber um grafo de dependências abstrato, ignorando o tempo necessário para geração do grafo de dependências abstrato.

O tempo de processamento da síntese de serviços é obtido em função de três parâmetros: quantidade de coreografias de serviços na síntese, quantidade de papéis por coreografias de serviços e quantidade de serviços candidatos por papéis disponíveis. Diante disso, o experimento considera coreografias de serviços sintéticas seguindo parâmetros e padrões dos experimentos anteriores.

Nosso objetivo em adotar esta estratégia foi realizar a análise considerando os piores casos do problema. Sendo assim, a síntese de serviços foi realizada variando cada um dos parâmetros enquanto os demais possuem valores fixos. Para análise da quantidade de coreografias foi realizado o experimento variando de 20 a 100, com incremento de 20 coreografias de serviços a cada nova iteração. Quanto aos papéis de cada coreografia, o experimento considerou a variação da quantidade de papéis por coreografia de 10 a 50. O número de serviços candidatos disponíveis por papel, variou de 100 a 500, com incremento de 50 serviços candidatos em cada iteração.

A quantidade de papéis em comum entre as coreografias de serviços foi fixada em *100%*. Isso acontece porque quanto maior for o número de papéis em comum entre as coreografias de serviços, maior o conflito de requisições das coreografias sobre um mesmo grupo de serviços candidatos, o que eleva o tempo de processamento da busca por serviços adequados para todas as coreografias de serviços, ou seja, pode-se chegar ao pior caso.

8.2.3 Análise e interpretação dos resultados

Para permitir a execução do experimento foi implementado um *script* responsável por realizar chamadas à síntese de serviços considerando as variações na entrada. Para a execução deste experimento foi utilizada uma máquina com a seguinte

configuração: CPU Intel® Core™ i5 2.67GHz, 6GB RAM, Ubuntu 14.04. Para permitir maior confiabilidade, para cada iteração os experimentos foram repetidos 100 vezes.

O gráfico da Figura 8.5 apresenta os resultados do tempo de processamento em função da quantidade de serviços candidatos por papel considerando diferentes números de coreografias de serviços.

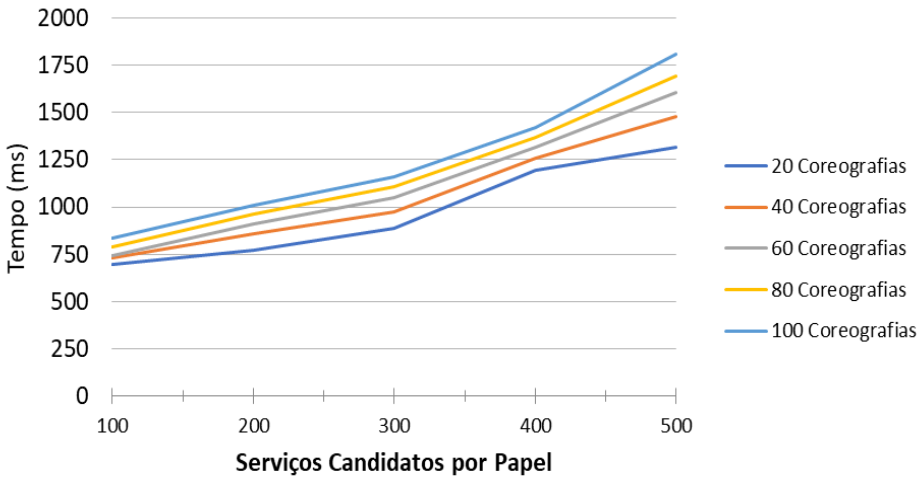


Figura 8.5: *Tempo de processamento da* SSMCP *em função da quantidade de serviços candidatos por papel, considerando diferentes quantidades de coreografias de serviços.*

Os resultados obtidos mostram que o tempo de processamento da *SSMCP* aumenta com o incremento do número de serviços candidatos por papel, que é um resultado esperado, ou seja, um maior número de serviços candidatos por papel requer mais esforço computacional, portanto, um tempo de processamento maior. Quanto maior o número de serviços candidatos por papel, maior é o espaço de busca. Vale notar que ao lidar com *50* tipos de serviço disponíveis, ao realizar a seleção com *500* serviços candidatos por papel, tem-se *25.000* serviços candidatos. De acordo com os dados, ao aumentar o número de serviços candidatos por papel de *100* para *500*, o tempo de processamento é incrementado em *89%* para *20* coreografias submetidas para implantação e *160%* para *100* coreografias submetidas para implantação.

O gráfico da Figura 8.6 apresenta os resultados do tempo de processamento em função do número de coreografias, considerando diferentes quantidades de serviços candidatos por papel. De acordo com o gráfico, ao aumentar o número de coreografias de serviços em cinco vezes, o tempo de processamento é incrementado em *19%* para *100* serviços candidatos por papel e *37%* para *500* serviços candidatos por papel. Os resultados obtidos mostram que o incremento do número de coreografias tem pouco impacto no tempo de processamento.

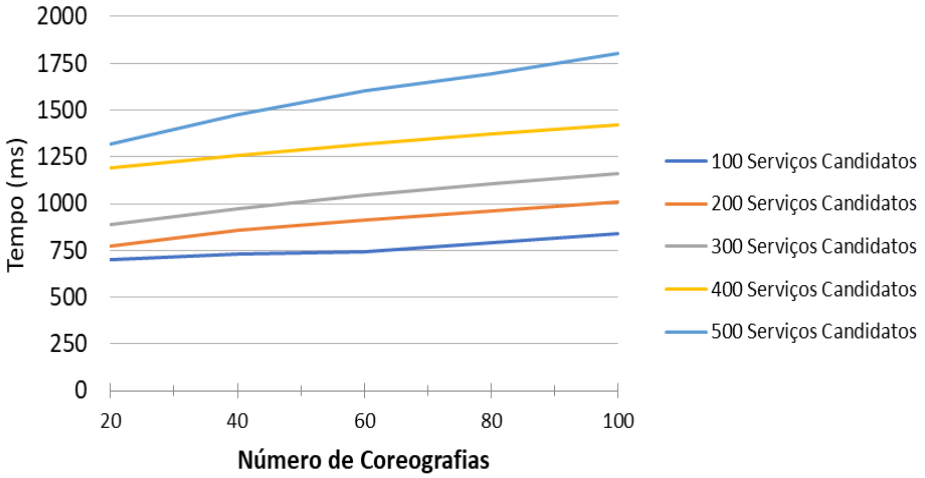


Figura 8.6: *Tempo de processamento da* SSMCP *em função da quantidade de coreografias de serviços, considerando diferentes quantidade de serviços candidatos.*

O mesmo é verdadeiro para o número de papéis, como mostra o gráfico da Figura 8.7, que apresenta os resultados do tempo de processamento em relação ao número de coreografias de serviços considerando a variação da quantidade de papéis. Nesse experimento, foi considerado intervalo de *10* a *50* papéis por coreografia. O número de serviços candidatos é fixo em *200*. De acordo com os dados, ao incrementar o número de papéis das coreografias em cinco vezes, o tempo de processamento é incrementado em *14%* para *20* coreografias e *33%* para *100* coreografias.

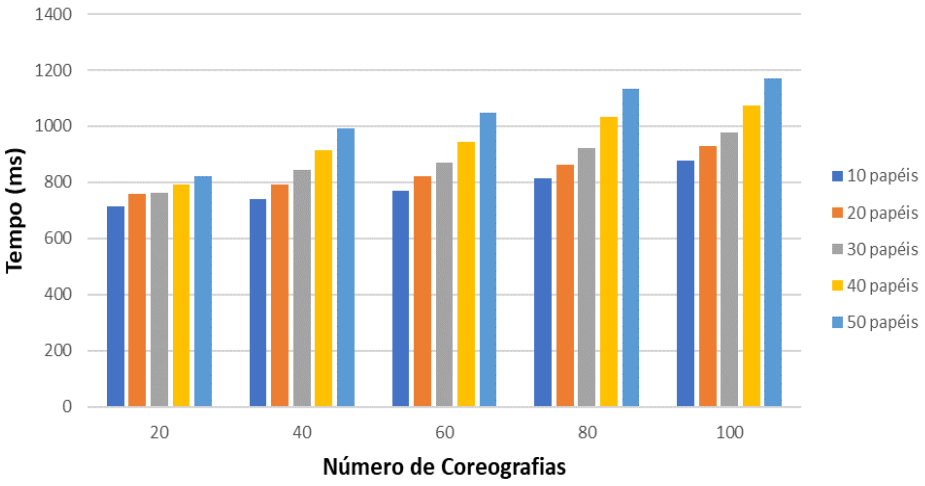


Figura 8.7: *Tempo de processamento da* SSMCP *em função do números de coreografias de serviços, considerando diferentes quantidades de papéis.*

Dado o cenário considerado nesta tese, onde os provedores de coreografias implantam as coreografias de serviços para posteriormente disponibilizar as aplica-

ções aos usuários, no pior cenário é possível realizar a seleção de serviços em um tempo próximo de 1.2 segundos, considerando o cenário utilizado.

Além disso, o tempo de processamento da *SSMCP* aumenta quase linearmente, juntamente com o crescimento do número de serviços candidatos por papel. Entretanto, para confirmar essa tendência de linearidade é necessário aumentar o número de experimentos. Em geral, nossa abordagem executa em *0,018s* em média por coreografia de serviços quando se tem *100* coreografias com *10* papéis cada e *500* serviços candidatos por papel. Mesmo não tendo realizado em nosso trabalho nenhum experimento analisando o tempo de processamento com outras abordagens, há uma série de resultados na literatura que indicam que o tempo de processamento da *SSMCP* para seleção de serviços para cada coreografia de serviços é aceitável [6, 84, 141].

8.3 Limitações dos experimentos

Dadas as métricas e padrões dos experimentos realizados nesta tese, juntamente com as suposições apresentadas, podemos concluir que a abordagem proposta para seleção de serviços permite que um conjunto de coreografias de serviços sejam implantadas de maneira eficiente, satisfazendo um conjunto de requisitos de QoS e levando em consideração o custo da utilização dos recursos utilizados pelos serviços selecionados. Entretanto, os experimentos sofrem de algumas limitações que podem impactar nos resultados apresentados. As principais limitações dos experimentos são discutidas nesta seção.

O conjunto de dados de entrada utilizados nos experimentos são dados sintéticos. Mesmo que os dados gerados sejam baseados em trabalhos importantes na área de seleção de serviços, esse conjunto de dados pode não representar fielmente cenários reais. Por exemplo, algumas métricas de QoS podem ser correlacionadas, onde o valor de uma métrica pode impactar no valor de outra, e com diferentes distribuições em domínios distintos. Em nosso experimento, o valor de cada métrica é gerado independente das demais métricas. O mesmo vale para a carga estimada por coreografia, que pode variar em diferentes cenários, impactando na seleção de serviços.

Outra limitação é em relação à organização dos serviços candidatos por tipos de serviço, uma vez que para cada tipo de serviço foram gerados entre 100 e 500 serviços candidatos. Não foi avaliada a correlação entre serviços candidatos de um mesmo tipo de serviço. Neste caso, foram gerados os valores de oferta de QoS e capacidade para cada serviço de forma independente. Além disso, o número real de serviços candidatos por papel pode ser mais restrito do que os valores adotados na literatura e nesta tese, uma vez que alguns papéis podem não ter 500 serviços candidatos disponíveis, e sim um valor bem inferior. Entretanto, um provedor de

serviços pode disponibilizar várias instâncias de uma implementação de serviço, onde cada instância é tratada como um serviço distinto, o que torna o valor do número de serviços candidatos por papel adequado com a realidade.

Como pôde ser visto nos resultados, ao utilizar a *SSMCP*, a oferta de QoS dos serviços selecionados são mais próximos dos requisitos de QoS dos provedores de coreografias, levando à diminuição do custo dos serviços, relacionado aos recursos utilizados para garantir a QoS ofertada. Entretanto, como os valores de QoS ofertados são mais próximos dos requisitos de QoS, devido ao uso da estratégia de avaliação *best-fit*, a ocorrência de variação na QoS oferecida pode ocasionar em violações de QoS. Nesses casos, pode haver penalidades, impactando no custo dos serviços, o que não foi levado em consideração nos experimentos. Vale ressaltar que abordagens que utilizam a estratégia *best-QoS* podem diminuir a chance de violações de QoS, embora tais violações também sejam possíveis, o que também acarretaria no aumento dos custos.

A análise do tempo de execução da síntese de serviços avalia a eficiência com que a *SSMCP* resolve o problema de seleção de serviços. O tempo de execução medido foi o tempo necessário para executar o algoritmo de síntese de serviços, desde a chegada de um grafo de dependências abstrato até a geração do grafo de dependências concreto correspondente. Nesta análise do tempo de execução não foi avaliado o tempo necessário para gerar os grafos de processos e combiná-los em um grafo de dependências abstrato. Embora esse processo de geração do grafo de dependências seja considerável, argumentamos que esse processo só precisa ser realizado quando uma coreografia é submetida pela primeira vez, diferentemente da síntese de serviços que poderá ser realizada várias vezes para um mesmo grafo de dependências.

Para o propósito de nossa avaliação, comparamos a abordagem proposta nesta tese com outras duas abordagens, que foram selecionadas dentre as abordagens similares discutidas no Capítulo 3. A escolha dessas duas abordagens se deve ao fato que ambas realizam a seleção de serviços sensível à QoS e à capacidade para múltiplas coreografias com diferentes valores de cargas para as coreografias, ao mesmo tempo que discutem o emprego de uma estratégia de avaliação de QoS, *best-QoS* ou *best-fit*. Além disso, esses dois trabalhos possuem objetivos próximos da abordagem proposta nesta tese. Até onde é de nosso conhecimento, a abordagem proposta por Wu *et al.* [139] é a única que realiza a seleção de serviços para várias composições utilizando a estratégia *best-fit*, enquanto que a abordagem proposta por Ardagna e Mirandola [7], que realiza a seleção de serviços para várias composições utilizando a estratégia *best-QoS*, possui outras abordagens similares na literatura. Logo, outras abordagens de seleção de serviços para múltiplas composições que utilizam estratégia *best-QoS* poderiam ser utilizadas para fins de comparação.

Consideramos que estas limitações não invalidam a avaliação desenvolvida, uma vez que seguimos modelos amplamente adotados na literatura para geração dos experimentos. Além disso, algumas dessas limitações possibilitam a definição de trabalhos futuros, discutidos no próximo capítulo.

8.4 Conclusões dos experimentos

Neste capítulo foram apresentados alguns experimentos para avaliar a abordagem proposta para a síntese de serviços. A avaliação foi realizada com base no protótipo descrito no capítulo anterior e nas coreografias de serviços sintéticas, baseadas no cenário apresentados no decorrer da tese. A realização dos experimentos teve como objetivo analisar a qualidade da abordagem proposta e verificar a viabilidade de seu uso. Como pôde ser visto nos resultados dos experimentos, nossa abordagem possui um melhor custo, em relação à utilização dos recursos pelos serviços selecionados, em comparação com outras abordagens.

Abordagens existentes que não lidam diretamente com papéis em comum entre múltiplas coreografias de serviços no processo de seleção, tendem a elevar o desperdício de recursos em alguns cenários e violações de QoS em outros. Nesse último caso, as abordagens apenas confiam no SLA para resolver violações de QoS, bem como utilizar a multiplexação estatística [125].

Por outro lado, a abordagem proposta nesta tese usa uma seleção eficiente dos serviços para minimizar tais violações, evitando assim os custos relacionados como compensação do usuário e superdimensionamento de recursos onde um serviço é implantado. Para isso, a *SSMCP* realiza uma análise global de todas as requisições de QoS juntamente com a carga prevista de forma a selecionar serviços que satisfaçam os requisitos e minimizem os custos associados aos recursos utilizados. Nesse sentido, as principais métricas que a *SSMCP* lida na seleção de serviços são a utilidade da QoS agregada e a capacidade contratada dos serviços selecionados.

Como pôde ser visto nos resultados, ao utilizar a *SSMCP*, a oferta de QoS dos serviços selecionados é mais próxima dos requisitos de QoS dos provedores de coreografias, levando à diminuição do custo dos serviços, relacionado aos recursos utilizados para garantir a QoS ofertada. Além disso, ao aplicar a *SSMCP*, a taxa de utilização, que mede o quanto da capacidade contratada é realmente utilizada, possui um valor médio entre *60%* e *80%*. Valores altos para a taxa de utilização, como apresentado pela *Abordagem BQ 2*, levam à minimização do desperdício em relação à capacidade contratada. Entretanto, qualquer variação na carga estimada pode levar à sobrecarga dos serviços e, por consequência, à violação da QoS oferecida. Além disso, [117] e [57] discutem que o melhor equilíbrio entre o desempenho de cada máquina e o custo em

relação ao consumo de energia[1] é entre *60%* e *80%*. Diante disso, a *SSMCP* consegue manter um baixo desperdício da capacidade contratada e uma margem aceitável em relação à variação da carga estimada.

O aumento na quantidade de serviços candidatos por papel é a variável que mais impacta na sobrecarga do tempo de processamento necessário para obter o resultado da síntese de serviços. Entretanto, o tempo de processamento da *SSMCP* aumenta quase linearmente com o incremento do número de serviços candidatos por papel. No experimento realizado, ao aumentar o número de serviços candidatos em cinco vezes, fixando os demais parâmetros, causou uma sobrecarga média adicional de apenas $0,96s$. Com isso, argumentamos que o uso da abordagem é factível para o cenário apresentado nesta tese. Além disso, como mostrado no resultado, nossa abordagem ainda é capaz de aumentar o número de coreografias de serviços satisfeitas.

Diante dos resultados dos experimentos, podemos observar que em cenários com várias coreografias de serviços, que devem ser implantadas ao mesmo tempo, concorrendo pelo mesmo conjunto de serviços candidatos, o processo de seleção de serviços utilizando a abordagem proposta nesta tese apresenta bons resultados mesmo com um número pequeno de serviços candidatos. Além disso, a nossa abordagem é ainda melhor com o aumento do número de papéis em comum entre coregrafias, conseguindo aumentar a taxa de utilização dos serviços selecionados. Portanto, em cenários com múltiplas coreografias de serviços que devem ser implantadas, o uso da abordagem *SSMCP*, em relação às demais abordagens, é capaz de garantir um aumento do número de coreografias implantadas ao mesmo tempo que o custo dos serviços selecionados é minimizado.

As demais abordagens sofrem de limitações ao não realizarem a seleção de serviços analisando as coreografias combinadas. Por exemplo, com um número baixo de serviços candidatos, tanto a *Abordagem BF 1* quanto a *Abordagem BQ 2* não conseguiram implantar todas as coreografias submetidas. Entretanto, em cenários onde a QoS ofertada pelos serviços candidatos é homogênea e a capacidade passa a ser o principal parâmetro de avaliação, a *Abordagem BQ 2* torna-se também uma boa alternativa por conseguir aumentar a taxa de utilização dos serviços selecionados.

[1] Geralmente, o custo para manter em execução a máquina física é estimado pelo seu consumo de energia [51].

CAPÍTULO 9

Conclusão

A adoção cada vez mais frequente de composições de serviços vem mudando a forma como sistemas são concebidos, projetados e implementados. Neste novo cenário, coreografias de serviços constituem um modelo de composição promissor, por permitir que a coordenação dos serviços seja realizada sem a necessidade de um controlador central. O processo de construção de uma coregrafia, envolve, primeiramente, a especificação da coreografia, onde são identificadas as funcionalidades exigidas para cada papel e seu inter-relacionamento com os demais papéis. Em seguida, são selecionados os serviços que serão utilizados para desempenhar cada papel presente em uma coreografia.

A seleção eficiente de serviços sensíveis à QoS e à capacidade para múltiplas coreografias de serviços impõe diversos desafios, como a avaliação multidimensional de QoS e a classificação dos serviços candidatos, o uso eficiente de recursos associados aos serviços selecionados e a minimização do custo.

Nesta tese, apresentamos uma abordagem de seleção de serviços que seleciona um conjunto de serviços para desempenhar papéis especificados por múltiplas coreografias, garantindo os requisitos de QoS e minimizando o custo associado aos serviços selecionados. A minimização do custo aos provedores de coreografias é realizada através de uma análise da QoS e carga requisitadas, juntamente com a análise da oferta de QoS e da capacidade dos serviços candidatos, uma vez que o custo de um serviço é diretamente relacionado à oferta de QoS e à capacidade do serviço.

Este capítulo resume as contribuições desta tese (Seção 9.1) e discute oportunidades para trabalhos científicos futuros (Seção 9.2).

9.1 Contribuições do trabalho

A garantia dos requisitos de QoS é um dos principais desafios no contexto de coreografias de serviços, uma vez que a QoS oferecida tem um papel crucial na seleção de serviços. Isso ocorre porque diferentes atributos de QoS podem ser usados na seleção de serviços como uma forma de diferenciação na preferência dos provedo-

res de coreografias em relação aos serviços candidatos para um mesmo papel. Sendo assim, a seleção dos serviços deve ser realizada visando não apenas os requisitos funcionais, mas também requisitos de QoS oriundos do modelo de negócio ou resultantes da busca por soluções que minimizem os **custos** em relação aos **recursos computacionais** associados aos serviços selecionados. Para garantir que essas propriedades sejam oferecidas no processo de seleção de serviços, é preciso selecionar de forma eficiente serviços para as coreografias, evitando à **subutilização** e à **sobrecarga** dos serviços selecionados. Se por um lado, a subutilização gera desperdício de recursos ao superdimensionar os recursos utilizados pelos serviços selecionados, por outro lado, quando um serviço não consegue garantir a QoS devido a uma sobrecarga, o usuário deve ser compensado financeiramente por tais violações de QoS, aumentando o custo dos provedores de coreografias.

Esses problemas tornam a satisfação da seleção de serviços ainda mais desafiadora, sobretudo ao considerarmos que um mesmo serviço pode ser compartilhado entre diversas coreografias com diferentes requisitos de QoS e carga. Para tanto, esta tese propôs uma abordagem de seleção de serviços que se beneficia de uma visão global de um conjunto de coreografias com papéis compartilhados, sujeitos a requisitos de QoS e carga diferentes, que devem ser implantadas em conjunto. A abordagem proposta garante o uso eficiente dos serviços, reduzindo o custo associado aos recursos selecionados, além de garantir a satisfação dos requisitos de QoS dos provedores de coreografias. As demais contribuições do trabalho são discutidas a seguir.

- *Definição de uma nova notação para representação de coreografias de serviços*: Nesta tese, apresentamos uma abordagem para representação de coreografias de serviços que permite associar requisitos de QoS e carga aos papéis individuais e à coreografia como um todo, em uma representação chamada de **grafo de processo sensível à QoS**.
- *Definição de uma notação para representação combinada de múltiplas coreografias de serviços*: A partir de um conjunto de grafos de processo sensível à QoS, apresentamos uma representação combinada de múltiplas coreografias com respectivos requisitos de QoS e carga, chamada de **grafo de dependências sensível à QoS**. O objetivo dessa especificação é unificar a representação dos papéis comuns entre coreografias e, assim, estabelecer uma visão global dos serviços a serem selecionados e, dessa forma, tornar explícitos os efeitos de seu compartilhamento.
- *Proposta de método de classificação dos serviços através dos seus múltiplos requisitos de QoS*: Devido à multidimensionalidade das métricas de QoS associadas aos serviços, desenvolvemos um método de classificação dos requisitos, que proporciona maior flexibilidade para a avaliação dos atributos de QoS, por meio

de uma **função de utilidade**. A avaliação dos requisitos dos provedores de coreografias, e sua associação com serviços candidatos, utiliza a estratégia ***best-fit*** de avaliação de QoS, como meio de reduzir o desperdício dos recursos onde os serviços são executados, ao mesmo tempo que os requisitos de QoS são assegurados. Com esta estratégia de avaliação, a seleção de serviços sensível à QoS pretende garantir que os recursos não sejam desperdiçados e que os requisitantes utilizem serviços que ofereçam QoS próxima do necessário, evitando uma grande diferença entre QoS ofertada e requisitada.

- *Formalização do problema de síntese de serviços*: Propomos a formalização do problema da síntese de serviços como um problema de otimização e apresentamos uma solução que estende o algoritmo húngaro na busca por emparelhamentos sobre grafos bipartidos.
- *Proposta de uma estratégia para seleção de serviços*: Alicerçados nos modelos de coreografias de serviços e função de utilidade, desenvolvemos uma **estratégia de síntese de serviços** que seleciona os serviços mais apropriados para cada papel das coreografias envolvidas na síntese, enquanto assegura que o compartilhamento de serviços não comprometerá a QoS requisitada, além de reduzir o custo associado ao uso dos recursos alocados. Embora não foram realizados experimentos em relação ao uso da abordagem proposta nesta tese para orquestrações de serviços, existem evidências que esta abordagem pode ser utilizada também para seleção de serviços para orquestrações.
- *Definição de uma arquitetura que considera todas as atividades relacionadas à seleção de serviços*: Com base na abordagem desenvolvida, propusemos uma **arquitetura que coordena todas as atividades**, desde a especificação das coreografias de serviços até a seleção de serviços. Implementamos parte dessa arquitetura em um protótipo que considera as atividades relacionadas à seleção dos serviços.

Dadas as principais contribuições, também foi realizado uma avaliação sobre a abordagem proposta. Os resultados dos experimentos de avaliação realizados indicam a efetividade e o desempenho da abordagem de seleção de serviços proposta e sua adequação aos objetivos propostos. A abordagem proposta nesta tese, ao utilizar a estratégia *best-fit* de avaliação de QoS, busca reduzir o custo dos recursos associados aos serviços selecionados ao evitar uma grande assimetria entre a QoS ofertada e a QoS requisitada. Entretanto, essa estratégia tende a aumentar o número de serviços candidatos selecionados, diminuindo o compartilhamento dos serviços selecionados entre as coreografias e a taxa de utilização dos mesmos, impactando negativamente no custo dos serviços. Logo, nossa abordagem tratou esse *trade-off* relaxando os valores da QoS requisitada, ou seja, aumentando a distância entre a QoS requisitada e a

QoS ofertada, ao aplicar uma extensão no algoritmo húngaro. Os resultados indicam que a abordagem proposta seleciona serviços cujas ofertas de QoS é mais próxima dos requisitos de QoS dos provedores de coreografias, levando à diminuição do custo dos serviços, relacionado aos recursos utilizados para garantir a QoS ofertada. Além disso, a abordagem maximiza a taxa de utilização dos serviços selecionados, evitando a dilapidação de recursos ao manter um baixo desperdício da capacidade contratada.

Ao selecionar serviços para múltiplas coreografias, combinadas de forma eficiente em relação a oferta de QoS e capacidade, concluímos que *i)* o uso das técnicas propostas reduz o custo de utilização dos recursos pelos serviços se comparado com as abordagens relacionadas utilizadas nos experimentos, onde um grupo de composições são processadas em sequência por tais abordagens; e *ii)* o tempo necessário para obter a solução da síntese de serviços é aceitável. Com base nestes resultados obtidos, argumentamos que o uso da abordagem de seleção de serviços proposta pode beneficiar os provedores de coreografias interessados em implantar um conjunto de coreografias de serviços, caso a seleção de serviços possa ser centralizada. Ao adotar essa abordagem de seleção de serviços, é possível implantar um conjunto de coreografias de serviços de maneira eficiente, inclusive maximizando o número de coreografias implantadas, pois se leva em consideração a interferência causada pelo compartilhamento de serviços entre elas, possibilitando satisfazer todos os requisitos de QoS enquanto é obtida a redução dos custos associados.

9.1.1 Publicações decorrentes da tese

A elaboração desta tese resultou em duas publicações [52], [78]. Nesse sentido, a pesquisa desenvolvida durante o doutorado teve como foco principal a seleção de serviços sensível à QoS para coreografias de serviços individuais. Como resultado, identificamos também um conjunto de desafios que devem ser considerados ao lidar com múltiplas coreografias, em especial, em relação ao compartilhamento de serviços entre as coreografias que devem ser implantadas. Essa pesquisa produziu a formalização do modelo para representação de coreografias de serviços, que culminou na representação do grafo de dependências descrito no seguinte artigo:

- GOMES, R.; LIMA, J.; COSTA, F.; ROCHA, R.; GEORGANTAS, N.. **A Model-Based Approach for the Pragmatic Deployment of Service Choreographies**. In: Antonio Celesti; Philipp Leitner. (Org.). *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015*, Taormina, Italy, September 15-17, 2015, Revised Selected Papers. 1ed.: Springer International Publishing, 2016, v. 567, p. 153-165.

Após formalizar os modelos para representação de coreografias de serviços, concretizamos uma abordagem inicial para seleção de serviços sensível à QoS para múltiplas coreografias. Para este fim, também propusemos uma função de utilidade de QoS que descreve os requisitos multidimensionais dos provedores de coreografia e uma abordagem de seleção de serviços que maximiza o ajuste entre a QoS solicitada pelos provedores de coreografia e a QoS oferecida pelos serviços, independentemente da ordem de processamento das coreografias. Além disso, uma arquitetura foi proposta (apresentada no Capítulo 7). Esse resultado foi descrito no seguinte artigo:

- LIMA, J.; COSTA, F.; ROCHA, R.. **An Approach for QoS-Aware Selection of Shared Services for Multiple Service Choreographies**. In: *10th IEEE International Symposium on Service-Oriented System Engineering (SOSE)*, Oxford, UK, 2016. 10 p.

O objetivo desse artigo foi enfatizar o uso de uma arquitetura capaz de permitir englobar todos os modelos e métodos propostos de maneira conjunta e coordenada para realizar a seleção de serviços para múltiplas coreografias, com provável compartilhamento de serviços entre elas.

9.2 Trabalhos futuros

Mesmo que o principal objetivo desta tese tenha sido atingido, identificamos algumas possibilidades de extensão deste trabalho. Além disso, detectamos alguns pontos que devem ser melhor investigados como desdobramento dos objetivos iniciais da pesquisa desenvolvida. Nesta seção discutimos os principais.

O principal foco da abordagem apresentada nesta tese é a seleção de serviços adequados para satisfazer um conjunto de requisitos de QoS impostos sobre um conjunto de coreografias. Contudo, um importante aspecto que deve ser considerado é garantir que esses requisitos se mantenham satisfeitos durante a encenação das coreografias, uma vez que os requisitos de QoS podem ser violados em tempo de execução por diferentes motivos.

De imediato, existem aqueles que são inerentes do cenário considerado, como flutuação natural na carga das coreografias implantadas. Em complemento, a dinamicidade presente em ambientes de execução faz com que coreografias de serviços implantadas tenham que lidar com mudanças que podem ocorrer nos recursos disponibilizados. Como exemplos de mudanças nessa categoria, pode haver alteração nos serviços oferecidos ou indisponibilidade de um serviço, bem como podem surgir novas oportunidades, como o oferecimento de novos serviços. De modo similar, novas coreografias que possuem papéis comuns com aquelas previamente implantadas podem

ser submetidas, impactando no compartilhamento dos serviços selecionados. Além disso, é natural supor que poderão ocorrer alterações nos requisitos de QoS estabelecidos pelos provedores de coreografias, causando mudanças diretas nas coreografias de serviços já implantadas.

Um processo de adaptação deveria se atentar às coreografias diretamente relacionadas à causa da adaptação, buscando preservar as coreografias não relacionadas. Uma questão a ser analisada é se um grafo de dependências completo precisa ser enviado à síntese de serviços toda vez que uma adaptação é iniciada ou uma nova coreografia precisa ser implantada, ou se apenas os vértices do grafo de dependências relacionados à ativação da adaptação. Para tal, diversos desafios associados à estabilidade da síntese de serviços em executar repetidamente a abordagem proposta, considerando mudanças pontuais na entrada original, precisam ser investigados para que a abordagem seja plenamente adaptada para lidar com mudanças em tempo de execução.

Há também adaptações relacionadas à solução proposta, uma vez que a solução usa modelos estáticos para oferta de QoS e capacidade dos serviços. Entretanto, a QoS ofertada e a capacidade dos serviços podem ser variáveis devido ao ambiente de execução dos serviços. Por exemplo, serviços implantados em ambientes de nuvem que se beneficiam da variabilidade de tipos de recursos, assim como da elasticidade em seu uso. Uma forma de lidar com esse aspecto é analisar o comportamento dos serviços utilizando, por exemplo, algoritmos de aprendizado. Ao se definir modelos dinâmicos para prever a QoS ofertada e capacidade dos serviços candidatos, uma abordagem de adaptação proativa pode ser considerada.

A variabilidade da QoS ofertada e da capacidade dos serviços candidatos também impactam o modelo de custo estático adotado nesta tese. Logo, uma análise sobre a relação entre a variabilidade da QoS e capacidade e o modelo de custo deve ser realizada. Isso se deve ao fato que o modelo de custo desta tese é baseado na oferta de QoS e capacidade estática dos serviços. Além disso, a variabilidade da QoS de serviços compartilhados, trazem desafios na garantia da QoS requisitada por diferentes coreografias que utilizam um mesmo serviços, afetando inclusive o custo global.

Em alguns cenários reais os provedores de serviços podem fazer descontos em relação ao custo financeiro dos serviços oferecidos em determinadas épocas do ano ou por um determinado período de tempo. Provedores de serviços também podem dar descontos quando um grupo de serviços é contratado, ou seja, quanto mais serviços de um provedor forem contratados, maior será o desconto. Em tais casos, um modelo de custo variável, levando em consideração diversas políticas de preço, pode gerar impacto na síntese de serviços. Esse problema pode ainda ser mais desafiador ao considerar que a QoS e a capacidade também são variáveis.

A grande tendência no desenvolvimento baseado em serviços é propor o uso de serviços com escopo cada vez mais bem definido, assumindo a responsabilidade por tarefas reduzidas. Dessa forma, as aplicações passam a ser desenvolvidas como um conjunto de serviços pequenos, cada um funcionando em seu próprio processo e se comunicando por meio de mecanismos de baixa sobrecarga [100]. Além de benefícios como separação de interesses e modularização, esta estratégia de desenvolvimento, conhecida como estilo arquitetural de microsserviços [115], permite que serviços escalem de maneira independente, permitindo melhor uso dos recursos. Um trabalho de pesquisa no contexto da síntese de serviços é realizar a seleção de microsserviços levando em consideração que diferentes microsserviços podem compartilhar os mesmos recursos. O compartilhamento de recursos entre microsserviços tem por objetivo a redução de custo e, por consequência, o custo global de um grupo de coreografias a serem implantadas. Sendo assim, a síntese de serviços poderia ser repensada para levar em consideração, além da garantia da QoS ofertada e da capacidade de um microsserviço compartilhado, a minimização do custo ao lidar com compartilhamento de recursos entre microsserviços selecionados, o que acrescenta uma nova variável a ser considerada durante a seleção de serviços.

Referências Bibliográficas

[1] Ahmed, T.; Srivastava, A. **Service choreography: Present and future**. In: *Services Computing (SCC), 2014 IEEE International Conference on*, p. 863–864. IEEE, 2014.

[2] Al-Masri, E.; Mahmoud, Q. H. **QoS-based discovery and ranking of Web services**. In: *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, p. 529–534. IEEE, 2007.

[3] Al-Masri, E.; Mahmoud, Q. H. **Investigating Web services on the World Wide Web**. In: *Proceedings of the 17th international conference on World Wide Web*, p. 795–804. ACM, 2008.

[4] Alrifai, M.; Risse, T. **Combining global optimization with local selection for efficient QoS-aware service composition**. In: *Proceedings of the 18th international conference on World Wide Web*, p. 881–890. ACM, 2009.

[5] Alrifai, M.; Risse, T.; Nejdl, W. **A hybrid approach for efficient Web service composition with end-to-end QoS constraints**. *ACM Transactions on the Web (TWEB)*, 6(2):7, 2012.

[6] Alrifai, M.; Skoutas, D.; Risse, T. **Selecting skyline services for QoS-based Web service composition**. In: *Proceedings of the 19th international conference on World Wide Web*, p. 11–20. ACM, 2010.

[7] Ardagna, D.; Mirandola, R. **Per-flow optimal service selection for Web services based processes**. *Journal of Systems and Software*, 83(8):1512–1523, 2010.

[8] Ardagna, D.; Pernici, B. **Adaptive service composition in flexible processes**. *Software Engineering, IEEE Transactions on*, 33(6):369–384, 2007.

[9] Autili, A. D. S. M.; Di Ruscio, D.; Inverardi, P. **Synthesizing an automata-based representation of bpmn2 choreography diagrams**. *ModComp at MoDELS*, 14, 2014.

[10] Barker, A.; Walton, C. D.; Robertson, D. **Choreographing Web services**. *Services Computing, IEEE Transactions on*, 2(2):152–166, 2009.

[11] Barros, A.; Dumas, M.; Oaks, P. **A critical overview of the Web services choreography description language (WS-CDL)**. *BPTrends Newsletter*, 3:1–24, 2005.

[12] Bartoletti, M.; Degano, P.; Ferrari, G.-L.; Zunino, R. **Semantics-based design for secure Web services**. *IEEE Transactions on Software Engineering*, 34(1):33–49, 2008.

[13] Bartolini, C.; Bertolino, A.; De Angelis, G.; Ciancone, A.; Mirandola, R. **Non-functional analysis of service choreographies**. In: *Principles of Engineering Service Oriented Systems (PESOS), 2012 ICSE Workshop on*, p. 8–14. IEEE, 2012.

[14] Baryannis, G.; Danylevych, O.; Karastoyanova, D.; Kritikos, K.; Leitner, P.; Rosenberg, F.; Wetzstein, B. **Service composition**. In: Papazoglou, M.; Pohl, K.; Parkin, M.; Metzger, A., editores, *Service Research Challenges and Solutions for the Future Internet*, volume 6500 de **Lecture Notes in Computer Science**, p. 55–84. Springer Berlin Heidelberg, 2010.

[15] Beloglazov, A.; Abawajy, J.; Buyya, R. **Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing**. *Future generation computer systems*, 28(5):755–768, 2012.

[16] Bocciarelli, P.; D'Ambrogio, A. **A bpmn extension for modeling non functional properties of business processes**. In: *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, p. 160–168. Society for Computer Simulation International, 2011.

[17] Bonatti, P. A.; Festa, P. **On optimal service selection**. In: *Proceedings of the 14th international conference on World Wide Web*, p. 530–538. ACM, 2005.

[18] Bouguettaya, A.; Singh, M.; Huhns, M.; Sheng, Q. Z.; Dong, H.; Yu, Q.; Neiat, A. G.; Mistry, S.; Benatallah, B.; Medjahed, B.; others. **A service computing manifesto: the next 10 years**. *Communications of the ACM*, 60(4):64–72, 2017.

[19] Buyya, R.; Garg, S. K.; Calheiros, R. N. **SLA-oriented resource provisioning for cloud computing: challenges, architecture, and solutions**. In: *Cloud and Service Computing (CSC), 2011 International Conference on*, p. 1–10. IEEE, 2011.

[20] Canfora, G.; Di Penta, M.; Esposito, R.; Villani, M. L. **An approach for QoS-aware service composition based on genetic algorithms**. In: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, p. 1069–1075. ACM, 2005.

[21] Canfora, G.; Di Penta, M.; Esposito, R.; Villani, M. L. **A framework for QoS-aware binding and re-binding of composite Web services**. *Journal of Systems and Software*, 81(10):1754–1769, 2008.

[22] Cardellini, V.; Casalicchio, E.; Grassi, V.; Iannucci, S.; Presti, F. L.; Mirandola, R. **Moses: A framework for QoS driven runtime adaptation of service-oriented systems**. *Software Engineering, IEEE Transactions on*, 38(5):1138–1159, 2012.

[23] Cardellini, V.; Casalicchio, E.; Grassi, V.; Lo Presti, F. **Flow-based service selection for Web service composition supporting multiple QoS classes**. In: *Web Services, 2007. ICWS 2007. IEEE International Conference on*, p. 743–750. IEEE, 2007.

[24] Cardellini, V.; Casalicchio, E.; Grassi, V.; Lo Presti, F.; Mirandola, R. **QoS-driven runtime adaptation of service oriented architectures**. In: *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, p. 131–140. ACM, 2009.

[25] Cardoso, J.; Sheth, A.; Miller, J.; Arnold, J.; Kochut, K. **Quality of service for workflows and Web service processes**. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308, 2004.

[26] Catalano, M.; Lo Casto, B.; Migliore, M. **Car sharing demand estimation and urban transport demand modelling using stated preference techniques**. 2008.

[27] Chen, F.; Dou, R.; Li, M.; Wu, H. **A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing**. *Computers & Industrial Engineering*, 99:423–431, 2016.

[28] Chen, Q.; Li, X.; Wang, Y. **Sla-driven cost-effective monitoring based on criticality for multi-tenant service-based systems**. *IEEE Access*, 2018.

[29] Chen, Y.; Huang, J.; Lin, C.; Hu, J. **A partial selection methodology for efficient QoS-aware service composition**. *IEEE Transactions on Services Computing*, 8(3):384–397, 2015.

[30] Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S.; others. **Web services description language (wsdl) 1.1**, 2001.

[31] Coulouris, G.; Dollimore, J.; Kindberg, T.; Blair, G. **Distributed systems: concepts and design**. Pearson education, 2012.

[32] Cuesta, C. E.; Cáceres, P.; Vela, B.; Cavero, J. M. **Comobility: a mobile platform for transport sharing**. *Mobile Computing*, p. 22, 2013.

[33] Curbera, F. **Component contracts in service-oriented architectures**. *Computer*, 40(11):74–80, 2007.

[34] de Medeiros, R. W.; Rosa, N. S.; Pires, L. F. **A metamodel for modeling cost behavior in service composition**. In: *Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on*, p. 84–91. IEEE, 2014.

[35] Decker, G.; Kopp, O.; Barros, A. **An introduction to service choreographies**. *Information Technology*, 50(2/2008):122–127, 2008.

[36] Decker, G.; Kopp, O.; Leymann, F.; Weske, M. **BPEL4Chor: Extending BPEL for modeling choreographies**. In: *Web Services, 2007. ICWS 2007. IEEE International Conference on*, p. 296–303. IEEE, 2007.

[37] Deng, Y.; Masoud Sadjadi, S.; Clarke, P. J.; Hristidis, V.; Rangaswami, R.; Wang, Y. **CVM – a communication virtual machine**. *Journal of Systems and Software*, 81(10):1640–1662, 2008.

[38] Diaz, A. P.; Batista, D. M. **A methodology to define QoS and sla requirements in service choreographies**. In: *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on*, p. 201–205. IEEE, 2012.

[39] Diestel, R. **Graph theory**. © Springer-Verlag New York, 2000.

[40] Dranidis, D.; Ramollari, E.; Kourtesis, D. **Run-time verification of behavioural conformance for conversational Web services**. In: *Web Services, 2009. ECOWS'09. Seventh IEEE European Conference on*, p. 139–147. IEEE, 2009.

[41] Duesterwald, E.; Bala, V. **Software profiling for hot path prediction: Less is more**. *ACM SIGOPS Operating Systems Review*, 34(5):202–211, 2000.

[42] Eckert, J.; Ertogrul, D.; Papageorgiou, A.; Repp, N.; Steinmetz, R. **The impact of service pricing models on service selection**. In: *Internet and Web Applications and Services, 2009. ICIW'09. Fourth International Conference on*, p. 316–321. IEEE, 2009.

[43] Eclipse. **BPMN2 Modeler**. https://www.eclipse.org/bpmn2-modeler/, 2017.

[44] Engler, L. **BPELgold: choreography on the service bus**. 2009.

[45] Fleming, P. J.; Wallace, J. J. **How not to lie with statistics: the correct way to summarize benchmark results**. *Communications of the ACM*, 29(3):218–221, 1986.

[46] Fok, C.-L.; Julien, C.; Roman, G.-C.; Lu, C. **Challenges of satisfying multiple stakeholders: quality of service in the internet of things**. In: *Proceedings of the 2nd workshop on software engineering for sensor network applications*, p. 55–60. ACM, 2011.

[47] Foster, H.; Uchitel, S.; Kramer, J. M. J. **Model-based analysis of obligations in Web service choreography**. In: *Advanced International Conference on Telecommunications / International Conference on Internet and Web Applications and Services*, volume 00, p. 149, 2006.

[48] Furtado, T.; Francesquini, E.; Lago, Nelson and, F. **A middleware for reflective Web service choreographies on the cloud**. In: *Proceedings of the 13th Workshop on Adaptive and Reflective Middleware*, p. 9. ACM, 2014.

[49] Furtado, T.; Francesquini, E.; Lago, N.; Kon, F. **Towards an enactment engine for dynamically reconfigurable and scalable choreographies**. In: *Services (SERVICES), 2014 IEEE World Congress on*, p. 325–332. IEEE, 2014.

[50] Geoffrion, A. M.; Nauss, R. **Exceptional paper parametric and postoptimality analysis in integer linear programming**. *Management Science*, 23(5):453–466, 1977.

[51] Gomes, R. **QoS-Aware Composition of Adaptive Service-Oriented Systems**. Tese de doutorado, Universidade de Federal de GoiÃ¡s, Brasil, 2017.

[52] Gomes, R.; Lima, J.; Costa, F.; Da Rocha, R.; Georgantas, N. **A model-based approach to pragmatic service choreography deployment**. In: *on Proceedings of the 2nd Workshop on Seamless Adaptive Multi-cloud Management of Service-based Applications*, 2015.

[53] Group, W. W.-C. W. **Web services choreography description language version 1.0.** https://www.w3.org/TR/ws-cdl-10/, 2005.

[54] Hamida, A. B.; Kon, F.; Oliva, G. A.; Dos Santos, C. E. M.; Lorré, J.-P.; Autili, M.; De Angelis, G.; Zarras, A.; Georgantas, N.; Issarny, V.; others. **An integrated development and runtime environment for the future internet**. In: *The Future Internet*, p. 81–92. Springer, 2012.

[55] He, Q.; Han, J.; Yang, Y.; Grundy, J.; Jin, H. **QoS-driven service selection for multi-tenant SaaS**. In: *Cloud computing (cloud), 2012 ieee 5th international conference on*, p. 566–573. IEEE, 2012.

[56] He, Q.; Yan, J.; Jin, H.; Yang, Y. **Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction**. *IEEE Transactions on Software Engineering*, 40(2):192–215, 2014.

[57] HoseinyFarahabady, M.; Lee, Y. C.; Zomaya, A. Y.; Tari, Z. **A QoS-aware resource allocation controller for function as a service (FaaS) platform**. In: *International Conference on Service-Oriented Computing*, p. 241–255. Springer, 2017.

[58] Huo, Y.; Qiu, P.; Zhai, J.; Fan, D.; Peng, H. **Multi-objective service composition model based on cost-effective optimization**. *Applied Intelligence*, p. 1–19, 2017.

[59] Hussain, O. K.; Hussain, F. K.; others. **IaaS cloud selection using MCDM methods**. In: *e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on*, p. 246–251. IEEE, 2012.

[60] ITU-T. **Definitions of terms related to quality of service - recommendation ITU-T E.800**, 2008.

[61] Jaeger, M. C.; Mühl, G. **QoS-based selection of services: the implementation of a genetic algorithm**. In: *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*, p. 1–12. VDE, 2007.

[62] Jin, H.; Zou, H.; Yang, F.; Lin, R.; Shuai, T. **Using bipartite graph for resolving multiple requests conflicts**. In: *Service Sciences (IJCSS), 2012 International Joint Conference on*, p. 46–50. IEEE, 2012.

[63] Jin, H.; Zou, H.; Yang, F.; Lin, R.; Shuai, T.; Zhao, X. **Large-scale multi-user service selection based on hybrid method**. *Advances in Information Sciences and Service Sciences, AICIT*, 4(8):361–380, 2012.

[64] Jongtaveesataporn, A.; Takada, S. **Rapid selection of services based on QoS**. In: *Proceedings of the 8th International Conference on Bioinspired Information and Communications Technologies*, p. 223–230. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[65] Kaewbanjong, K.; Intakosum, S. **QoS attributes of Web services: a systematic review and classification**. *Journal of Advanced Management Science Vol*, 3(3), 2015.

[66] Kang, G.; Liu, J.; Tang, M.; Liu, X.; Fletcher, K. K. **Web service selection for resolving conflicting service requests**. In: *Web Services (ICWS), 2011 IEEE International Conference on*, p. 387–394. IEEE, 2011.

[67] Keller, A.; Ludwig, H. **The WSLA framework: specifying and monitoring service level agreements for Web services**. *Journal of Network and Systems Management*, 11(1):57–81, 2003.

[68] Khanouche, M. E.; Amirat, Y.; Chibani, A.; Kerkar, M.; Yachir, A. **Energy-centered and QoS-aware services selection for internet of things**. *IEEE Transactions on Automation Science and Engineering*, 13(3):1256–1269, 2016.

[69] Ko, H.-G.; Ko, I.-Y.; Cho, J.-H. **Adaptive service selection according to the service density in multiple QoS aspects**.

[70] Kon, F.; Santana, E. F. Z. **Cidades Inteligentes: conceitos, plataformas e desafios**. In: *Jornadas de Atualização em Informática (JAI) 2016*, p. 1–48. Sociedade Brasileira de Computação, 2016.

[71] Kritikos, K.; Plexousakis, D. **Requirements for QoS-based Web service description and discovery**. *Services Computing, IEEE Transactions on*, 2(4):320–337, 2009.

[72] Kuhn, H. W. **The hungarian method for the assignment problem**. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[73] Leitner, P.; Hummer, W.; Dustdar, S. **Cost-based optimization of service compositions**. *IEEE Transactions on Services Computing*, 6(2):239–251, 2013.

[74] Leonardo, L.; Lago, N.; Gerosa, M. A.; Kon, F. **Um middleware para encenaçao automatizada de coreografias de serviços Web em ambientes**

de computaçao em nuvem. In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2013)*, 2013.

[75] Leymann, F. **Web Services Flow Language (WSFL 1.0)**. *IBM Corporation*, 2001. Disponível em: http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[76] Li, J.; Zhao, Y.; Liu, M.; Sun, H.; Ma, D. **An adaptive heuristic approach for distributed QoS-based service composition**. In: *Computers and Communications (ISCC), 2010 IEEE Symposium on*, p. 687–694. IEEE, 2010.

[77] Li, W.; Delicato, F. C.; Pires, P. F.; Lee, Y. C.; Zomaya, A. Y.; Miceli, C.; Pirmez, L. **Efficient allocation of resources in multiple heterogeneous wireless sensor networks**. *Journal of Parallel and Distributed Computing*, 74(1):1775–1788, 2014.

[78] Lima, J. C.; Rocha, R. C.; Costa, F. M. **An approach for QoS-aware selection of shared services for multiple service choreographies**. In: *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, p. 221–230. IEEE, March 2016.

[79] Liu, Y.; Ngu, A. H.; Zeng, L. Z. **QoS computation and policing in dynamic Web service selection**. In: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, p. 66–73. ACM, 2004.

[80] Liu, Z.-Z.; Chu, D.-H.; Jia, Z.-P.; Shen, J.-Q.; Wang, L. **Two-stage approach for reliable dynamic Web service composition**. *Knowledge-Based Systems*, 97:123–143, 2016.

[81] Liu, Z.-Z.; Xue, X.; Shen, J.-q.; Li, W.-R. **Web service dynamic composition based on decomposition of global QoS constraints**. *The International Journal of Advanced Manufacturing Technology*, 69(9-12):2247–2260, 2013.

[82] Maamar, Z.; Sheng, Q. Z.; Benatallah, B. **On composite Web services provisioning in an environment of fixed and mobile computing resources**. *Information Technology and Management*, 5(3-4):251–270, 2004.

[83] Mabrouk, N. B.; Beauche, S.; Kuznetsova, E.; Georgantas, N.; Issarny, V. **Qos-aware service composition in dynamic service oriented environments**. In: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, p. 123–142. Springer, 2009.

[84] Mabrouk, N. B.; Georgantas, N.; Issarny, V. **Multi-objective service composition in ubiquitous environments with service dependencies**. *arXiv preprint arXiv:1611.09169*, 2016.

[85] Mardukhi, F.; Nematbakhsh, N.; Zamanifar, K.; Barati, A. **QoS decomposition for service composition using genetic algorithm**. *Applied Soft Computing*, 13(7):3409–3421, 2013.

[86] Maros, I. **Computational techniques of the simplex method**, volume 61. Springer Science & Business Media, 2012.

[87] Martello, S.; Pisinger, D.; Vigo, D. **The three-dimensional bin packing problem**. *Operations Research*, 48(2):256–267, 2000.

[88] Mazza, R.; Lockerbie, J.; Lavazza, L.; Sioli, L.; Ripa, G.; Panza, G. **Choreos deliverable d7: Mobile-enabled coordination of people-requirements specification and use case definition**. http://www.choreos.eu/bin/download/Download/Deliverables/CHOReOS_-WP07D7.1Mobile-enabled_coordination_of_people_requirements_-specification_and_use_case_definitionVB.pdf, 2012.

[89] Menascé, D. A. **QoS issues in Web services**. *Internet Computing, IEEE*, 6(6):72–75, 2002.

[90] Mendling, J.; Lassen, K. B.; Zdun, U.; others. **Transformation strategies between block-oriented and graph-oriented process modelling languages**. In: *Multikonferenz Wirtschaftsinformatik*, volume 2, p. 297–312. unknown, 2006.

[91] Michlmayr, A.; Rosenberg, F.; Leitner, P.; Dustdar, S. **Comprehensive QoS monitoring of Web services and event-based sla violation detection**. In: *Proceedings of the 4th international workshop on middleware for service oriented computing*, p. 1–6. ACM, 2009.

[92] Michlmayr, A.; Rosenberg, F.; Leitner, P.; Dustdar, S. **End-to-end support for QoS-aware service selection, binding, and mediation in vresco**. *IEEE Transactions on Services Computing*, 3(3):193–205, 2010.

[93] Mills-Tettey, G. A.; Stentz, A.; Dias, M. B. **The dynamic hungarian algorithm for the assignment problem with changing costs**. Relatório técnico CMU-RI-TR-07-27, Pittsburgh, PA, July 2007.

[94] Moghaddam, M.; Davis, J. G. **Service selection in Web service composition: A comparative review of existing approaches**. In: *Web Services Foundations*, p. 321–346. Springer, 2014.

[95] Nanda, M. G.; Chandra, S.; Sarkar, V. **Decentralizing execution of composite Web services**. In: *ACM Sigplan Notices*, volume 39, p. 170–187. ACM, 2004.

[96] Ngoko, Y.; Cérin, C.; Goldman, A. **Improving the quality of online search services: on the service multi-selection problem**. In: *Services Computing (SCC), 2016 IEEE International Conference on*, p. 243–250. IEEE, 2016.

[97] Nguyen, X. T.; Kowalczyk, R.; Han, J. **Using dynamic asynchronous aggregate search for quality guarantees of multiple Web services compositions**. In: *Service-Oriented Computing–ICSOC 2006*, p. 129–140. Springer, 2006.

[98] Oliveira, T. **Efficient processing of multiway spatial join queries in distributed systems**. 2017.

[99] OMG. **Documents Associated with Business Process Model and Notation (BPMN) Version 2.0**. http://www.omg.org/spec/BPMN/2.0/, 2011.

[100] Pahl, C. **Containerization and the PaaS Cloud**. *IEEE Cloud Computing*, 2(3):24–31, 2015.

[101] Palade, A.; Cabrera, C.; White, G.; Clarke, S. **Stigmergic service composition and adaptation in mobile environments**. In: *16th International Conference on Service Oriented Computing (ICSOC)*, 2018.

[102] Papazoglou, M. P.; Dubray, J.-j. **A survey of Web service technologies**. 2004.

[103] Papazoglou, M. P.; Traverso, P.; Dustdar, S.; Leymann, F. **Research directions in service-oriented computing**. *4a IC-SOC 2006*, p. 28, 2006.

[104] Parejo, J. A.; Segura, S.; Fernandez, P.; Ruiz-Cortés, A. **QoS-aware Web services composition using grasp with path relinking**. *Expert Systems with Applications*, 41(9):4211–4223, 2014.

[105] Peng, X.; Changsong, L. **ESCA: Evolution-strategy based service composition algorithm for multiple QoS constrained cloud applications**. *International Journal of Future Generation Communication and Networking*, 7(1):249–260, 2014.

[106] Ramacher, R.; Mönch, L. **Cost-minimizing service selection in the presence of end-to-end qos constraints and complex charging models**. In: *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, p. 154–161. IEEE, 2012.

[107] Rosa, N. S.; Cunha, P. R.; Justo, G. R. **Process NFL: A language for describing non-functional properties**. In: *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, p. 3676–3685. IEEE, 2002.

[108] Rosario, S.; Benveniste, A.; Jard, C. **Flexible probabilistic QoS management of transaction based Web services orchestrations**. In: *Web Services, 2009. ICWS 2009. IEEE International Conference on*, p. 107–114. IEEE, 2009.

[109] Rosenberg, F.; Enzi, C.; Michlmayr, A.; Platzer, C.; Dustdar, S. **Integrating quality of service aspects in top-down business process development using WS-CDL and WS-BPEL**. In: *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, p. 15–15. IEEE, 2007.

[110] Saeedi, K.; Zhao, L.; Sampaio, P. R. F. **Extending BPMN for supporting customer-facing service quality requirements**. In: *Web Services (ICWS), 2010 IEEE International Conference on*, p. 616–623. IEEE, 2010.

[111] Sathya, M.; Swarnamugi, M.; Dhavachelvan, P.; Sureshkumar, G. **Evaluation of QoS based Web-service selection techniques for service composition**. *International Journal of Software Engineering*, 1(5):73–90, 2010.

[112] Schuller, D.; Lampe, U.; Eckert, J.; Steinmetz, R.; Schulte, S. **Cost-driven optimization of complex service-based workflows for stochastic QoS parameters**. In: *Web Services (ICWS), 2012 IEEE 19th International Conference on*, p. 66–73. IEEE, 2012.

[113] Shen, Y.; Yang, X.; Wang, Y.; Ye, Z. **Optimizing QoS-aware services composition for concurrent processes in dynamic resource-constrained environments**. In: *Web Services (ICWS), 2012 IEEE 19th International Conference on*, p. 250–258. IEEE, 2012.

[114] Sheng, Q. Z.; Qiao, X.; Vasilakos, A. V.; Szabo, C.; Bourne, S.; Xu, X. **Web services composition: A decades overview**. *Information Sciences*, 280:218–238, 2014.

[115] Sill, A. **The design and architecture of microservices**. *IEEE Cloud Computing*, 3(5):76–80, 2016.

[116] Sommerville, I. **Engenharia de software**. Pearson, 2012.

[117] Srikantaiah, S.; Kansal, A.; Zhao, F. **Energy aware consolidation for cloud computing**. In: *Proceedings of the 2008 conference on Power aware computing and systems*, volume 10, p. 1–5. San Diego, California, 2008.

[118] Strunk, A. **QoS-aware service composition: a survey**. In: *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, p. 67–74. IEEE, 2010.

[119] Su, J.; Bultan, T.; Fu, X.; Zhao, X. **Towards a theory of Web service choreographies**. In: *Web Services and Formal Methods*, p. 1–16. Springer, 2008.

[120] Su, K.; Liangli, M.; Xiaoming, G.; Yufei, S. **An efficient parameter-adaptive genetic algorithm for service selection with end-to-end QoS constraints**. *Journal of Computational Information Systems*, 10(2):581–588, 2014.

[121] Szwarcfiter, J. L.; Markenzon, L. **Estruturas de dados e seus algoritmos**, volume 2. Livros Técnicos e Científicos, 1994.

[122] Taher, L.; El Khatib, H.; Basha, R. **A framework and QoS matchmaking algorithm for dynamic Web services selection**. In: *Second International Conference on Innovations in Information Technology (IIT05)), Dubai, UAE*, 2005.

[123] Temglit, N.; Chibani, A.; Djouani, K.; Nacer, M. A. **A distributed agent-based approach for optimal QoS selection in Web of object choreography**. *IEEE Systems Journal*, 2017.

[124] Tian, M.; Gramm, A.; Naumowicz, T.; Ritter, H.; Freie, J. **A concept for QoS integration in Web services**. In: *Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on*, p. 149–155. IEEE, 2003.

[125] Tse, D. N. C.; Gallager, R. G.; Tsitsiklis, J. N. **Statistical multiplexing of multiple time-scale markov streams**. *IEEE Journal on Selected Areas in Communications*, 13(6):1028–1038, 1995.

[126] Vincent, H.; Issarny, V.; Georgantas, N.; Francesquini, E.; Goldman, A.; Kon, F. **CHOReOS: scaling choreographies for the internet of the future**. In: *Middleware'10 Posters and Demos Track*, p. 8. ACM, 2010.

[127] W3C. **Web services description requirements**. https://www.w3.org/TR/2002/WD-ws-desc-reqs-20021028/ws-desc-reqs.pdf, 2002.

[128] Walsh, W. E.; Tesauro, G.; Kephart, J. O.; Das, R. **Utility functions in autonomic systems**. In: *Autonomic Computing, 2004. Proceedings. International Conference on*, p. 70–77. IEEE, 2004.

[129] Wang, H.; Wang, X.; Hu, X.; Zhang, X.; Gu, M. **A multi-agent reinforcement learning approach to dynamic service composition**. *Information Sciences*, 363:96–119, 2016.

[130] Wang, L.; Shen, J.; Di, C.; Li, Y.; Zhou, Q. **Towards minimizing cost for composite data-intensive services**. In: *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, p. 293–298. IEEE, 2013.

[131] Wang, S.; Hsu, C.-H.; Liang, Z.; Sun, Q.; Yang, F. **Multi-user Web service selection based on multi-QoS prediction**. *Information Systems Frontiers*, 16(1):143–152, 2014.

[132] Wang, W.; Li, B.; Liang, B. **Towards optimal capacity segmentation with hybrid cloud pricing**. In: *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, p. 425–434. IEEE, 2012.

[133] Wang, X.; Liu, J.; Cao, B.; Tang, M. **A global optimal service selection approach based on QoS and load-aware in cloud environment**. In: *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*, p. 762–768. IEEE, 2013.

[134] Wang, Y.; He, Q.; Zhang, X.; Ye, D.; Yang, Y. **Efficient QoS-aware service recommendation for multi-tenant service-based systems in cloud**. *IEEE Transactions on Services Computing*, 2017.

[135] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. **Experimentation in software engineering: an introduction**. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[136] Wolf, K. **Does my service have partners?** In: *Transactions on Petri Nets and Other Models of Concurrency II*, p. 152–171. Springer, 2009.

[137] Wolke, A.; Tsend-Ayush, B.; Pfeiffer, C.; Bichler, M. **More than bin packing: dynamic resource allocation strategies in cloud data centers**. *Information Systems*, 52:83–95, 2015.

[138] Wu, L.; Garg, S. K.; Buyya, R. **SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments**. In: *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, p. 195–204. IEEE Computer Society, 2011.

[139] Wu, Q.; Zhu, Q.; Jian, X.; Ishikawa, F. **Broker-based SLA-aware composite service provisioning**. *Journal of Systems and Software*, 96:194–201, 2014.

[140] Xu, L.; Jennings, B. **A cost-minimizing service composition selection algorithm supporting time-sensitive discounts**. In: *Services Computing (SCC), 2010 IEEE International Conference on*, p. 402–408. IEEE, 2010.

[141] Xu, X.; Liu, Z.; Wang, Z.; Sheng, Q. Z.; Yu, J.; Wang, X. **S-ABC: A paradigm of service domain-oriented artificial bee colony algorithms for service selection and composition**. *Future Generation Computer Systems*, 68:304–319, 2017.

[142] Xu, Y.; Saifullah, A.; Chen, Y.; Lu, C.; Bhattacharya, S. **Near optimal multi-application allocation in shared sensor networks**. In: *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*, p. 181–190. ACM, 2010.

[143] Yahia, E. B. H. **A language-based approach for Web service composition**. Tese de doutorado, Université de Bordeaux, 2017.

[144] Yang, K.; Galis, A.; Chen, H.-H. **QoS-aware service selection algorithms for pervasive service composition in mobile wireless environments**. *Mobile Networks and Applications*, 15(4):488–501, 2010.

[145] Ye, Z.; Zhou, X.; Bouguettaya, A. **Genetic algorithm based QoS-aware service compositions in cloud computing**. In: *International Conference on Database Systems for Advanced Applications*, p. 321–334. Springer, 2011.

[146] Yoon, K. P.; Hwang, C.-L. **Multiple attribute decision making: an introduction**, volume 104. Sage publications, 1995.

[147] Yu, Q.; Liu, X.; Bouguettaya, A.; Medjahed, B. **Deploying and managing Web services: issues, solutions, and directions**. *The International Journal on Very Large Data Bases*, 17(3):537–572, 2008.

[148] Yu, T.; Lin, K.-J. **Service selection algorithms for Web services with end-to-end QoS constraints**. *Information Systems and E-Business Management*, 3(2):103–126, 2005.

[149] Yu, T.; Zhang, Y.; Lin, K.-J. **Efficient algorithms for Web services selection with end-to-end QoS constraints**. *ACM Transactions on the Web (TWEB)*, 1(1):6, 2007.

[150] Zaha, J. M.; Barros, A.; Dumas, M.; ter Hofstede, A. **Lets dance: a language for service behavior modeling**. In: *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, p. 145–162. Springer, 2006.

[151] Zaha, J. M.; Dumas, M.; Ter Hofstede, A.; Barros, A.; Decker, G. **Service interaction modeling: bridging global and local views**. In: *Enterprise Distributed Object Computing Conference, 2006. EDOC'06. 10th IEEE International*, p. 45–55. IEEE, 2006.

[152] Zeng, L.; Benatallah, B.; Dumas, M.; Kalagnanam, J.; Sheng, Q. Z. **Quality driven Web services composition**. In: *Proceedings of the 12th international conference on World Wide Web*, p. 411–421. ACM, 2003.

[153] Zeng, L.; Benatallah, B.; Ngu, A. H.; Dumas, M.; Kalagnanam, J.; Chang, H. **QoS-aware middleware for Web services composition**. *Software Engineering, IEEE Transactions on*, 30(5):311–327, 2004.

[154] Zheng, Z.; Wu, X.; Zhang, Y.; Lyu, M. R.; Wang, J. **QoS ranking prediction for cloud services**. *IEEE transactions on parallel and distributed systems*, 24(6):1213–1222, 2013.

[155] Zhu, W.; Yin, B.; Gong, S.; Cai, K.-Y. **An approach to Web services selection for multiple users**. *IEEE Access*, 5:15093–15104, 2017.

APÊNDICE A

Representação do grafo de processo sensível à QoS utilizando XML referente à coreografia para consulta de rota da Figura 5.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<processGraph>
 <!-- Vértices do grafo -->
 <vertices>
    <!-- Vértice inicial -->
    <vertex id="START" type="START">
   <outgoings>
    <outgoing>edge_1</outgoing>
   </outgoings>
  </vertex>
    <vertex id="APP" type="ROLE" >
   <!-- Descrição de um papel -->
   <roleDescriptor/>
   <!-- Saídas de um vértice -->
   <outgoings>
    <outgoing>edge_2</outgoing>
   </outgoings>
  </vertex>
    <vertex id="TRANSITO" type="ROLE">
   <roleDescriptor>
    <name>TRANSITO</name>
        <logicalOperations>
     <logicalOperation>
      <name>requisicaoRota</name>
            <uri>/requisicaoRota/{id}</uri>
     </logicalOperation>
        </logicalOperations>
```

```
28      </roleDescriptor>
29    <outgoings>
30     <outgoing>edge_3</outgoing>
31     <outgoing>edge_4</outgoing>
32      </outgoings>
33     </vertex>
34     <vertex id="LOCALIZACAO" type="ROLE">
35    <roleDescriptor>
36         <name>LOCALIZACAO</name>
37         <logicalOperations>
38      <logicalOperation>
39       <name>consulta</name>
40             <uri>/consulta/{id}</uri>
41      </logicalOperation>
42         </logicalOperations>
43      </roleDescriptor>
44      <outgoings/>
45     </vertex>
46     <vertex id="TRAFEGO" type="ROLE">
47    <roleDescriptor>
48     <name>TRAFEGO</name>
49         <logicalOperations>
50      <logicalOperation>
51       <name>condicoesVia</name>
52             <uri>/condicoesVia/{id}</uri>
53      </logicalOperation>
54         </logicalOperations>
55      </roleDescriptor>
56    <outgoings>
57     <outgoing>edge_7</outgoing>
58      </outgoings>
59     </vertex>
60     <vertex id="METEOROLOGIA" type="ROLE">
61    <roleDescriptor>
62     <name>METEOROLOGIA</name>
63         <logicalOperations>
64      <logicalOperation>
65       <name>previsaoTempo</name>
66             <uri>/previsaoTempo/{id}</uri>
67      </logicalOperation>
68         </logicalOperations>
```

```
69      </roleDescriptor>
70    <outgoings>
71     <outgoing>edge_8</outgoing>
72      </outgoings>
73     </vertex>
74     <!-- Conectores entre serviços -->
75     <vertex id="ANDF_1" type="ANDF">
76    <outgoings>
77     <outgoing>edge_5</outgoing>
78     <outgoing>edge_6</outgoing>
79    </outgoings>
80     </vertex>
81     <vertex id="ANDJ_1" type="ANDJ">
82    <outgoings>
83     <outgoing>edge_9</outgoing>
84    </outgoings>
85     </vertex>
86     <!-- Vértice final -->
87   <vertex id="END_1" type="END" />
88  </vertices>
89  <!-- Arestas do grafo de processo-->
90   <edges>
91    <!-- Requisição a uma operação -->
92    <edge id="edge_1" vertexSource="START" vertexTarget="APP"
        operation="" />
93      <edge id="edge_2" vertexSource="APP" vertexTarget="
        TRANSITO"
94             operation="consulta" />
95      <edge id="edge_3" vertexSource="TRANSITO" vertexTarget="
        LOCALIZACAO"
96             operation="consulta" />
97      <edge id="edge_4" vertexSource="TRANSITO" vertexTarget="
        ANDF1" />
98      <edge id="edge_5" vertexSource="ANDF1" vertexTarget="
        TRAFEGO"
99             operation="condicoesVia"/>
100   <edge id="edge_6" vertexSource="ANDF1" vertexTarget="
        METEOROLOGIA"
101            operation="previsaoTempo"/>
102     <edge id="edge_7" vertexSource="TRAFEGO" vertexTarget="
        ANDJ" />
```

```
103     <edge id="edge_8" vertexSource="METEOROLOGIA"
            vertexTarget="ANDJ" />
104     <edge id="edge_9" vertexSource="ANDJ" vertexTarget="END"
            />
105   </edges>
106   <!-- Especificação de requisitos de QoS -->
107   <requirements>
108     <qosrequirement role="TRANSITO" operation="requisicaoRota
          ">
109      <qosAttributes>
110       <attribute>
111        <metric> br.ufg.inf.metric.ResponseTime </metric>
112        <relationalOp> <= </relationalOp>
113        <targetValue> 10.0 </targetValue>
114        <weightValue> 1.0 </weightValue>
115        <classificationValue> Alvo </classificationValue>
116           </attribute>
117           <attribute>
118        <metric> br.ufg.inf.metric.Disponibilidade </metric>
119        <relationalOp> >= </relationalOp>
120        <targetValue> 99.7 </targetValue>
121        <weightValue> 1.0 </weightValue>
122        <classificationValue> Alvo </classificationValue>
123           </attribute>
124           <attribute>
125        <metric> br.ufg.inf.metric.Reputação </metric>
126        <relationalOp> >= </relationalOp>
127        <targetValue> 95 </targetValue>
128        <weightValue> 1.0 </weightValue>
129        <classificationValue> Alvo </classificationValue>
130           </attribute>
131      </qosAttributes>
132      <!-- Carga estimada -->
133      <load>50</load>
134       </qosrequirement>
135     <qosrequirement role="LOCALIZACAO" operation="consulta">
136      <qosAttributes>
137       <attribute>
138        <metric> br.ufg.inf.metric.ResponseTime </metric>
139        <relationalOp> <= </relationalOp>
140        <targetValue> 6.5 </targetValue>
```

```
141         <weightValue> 1.0 </weightValue>
142         <classificationValue> Alvo </classificationValue>
143            </attribute>
144            <attribute>
145         <metric> br.ufg.inf.metric.Disponibilidade </metric>
146         <relationalOp> >= </relationalOp>
147         <targetValue> 99.8 </targetValue>
148         <weightValue> 1.0 </weightValue>
149         <classificationValue> Alvo </classificationValue>
150            </attribute>
151            <attribute>
152         <metric> br.ufg.inf.metric.Reputação </metric>
153         <relationalOp> >= </relationalOp>
154         <targetValue> 98 </targetValue>
155         <weightValue> 1.0 </weightValue>
156         <classificationValue> Alvo </classificationValue>
157            </attribute>
158      </qosAttributes>
159      <!-- Carga estimada -->
160      <load>50</load>
161       </qosrequirement>
162         <qosrequirement role="TRAFEGO" operation="
              previsaoTempo">
163      <qosAttributes>
164       <attribute>
165        <metric> br.ufg.inf.metric.ResponseTime </metric>
166        <relationalOp> <= </relationalOp>
167        <targetValue> 4.2 </targetValue>
168        <weightValue> 1.0 </weightValue>
169        <classificationValue> Alvo </classificationValue>
170           </attribute>
171           <attribute>
172        <metric> br.ufg.inf.metric.Disponibilidade </metric>
173        <relationalOp> >= </relationalOp>
174        <targetValue> 99.5 </targetValue>
175        <weightValue> 1.0 </weightValue>
176        <classificationValue> Alvo </classificationValue>
177           </attribute>
178           <attribute>
179        <metric> br.ufg.inf.metric.Reputação </metric>
180        <relationalOp> >= </relationalOp>
```

```
181         <targetValue> 90 </targetValue>
182         <weightValue> 1.0 </weightValue>
183         <classificationValue> Alvo </classificationValue>
184            </attribute>
185       </qosAttributes>
186       <!-- Carga estimada -->
187       <load>50</load>
188        </qosrequirement>
189          <qosrequirement role="METEOROLOGIA" operation="
               condicoesVia">
190       <qosAttributes>
191        <attribute>
192         <metric> br.ufg.inf.metric.ResponseTime </metric>
193         <relationalOp> <= </relationalOp>
194         <targetValue> 3.4 </targetValue>
195         <weightValue> 1.0 </weightValue>
196         <classificationValue> Alvo </classificationValue>
197            </attribute>
198            <attribute>
199         <metric> br.ufg.inf.metric.Disponibilidade </metric>
200         <relationalOp> >= </relationalOp>
201         <targetValue> 99.6 </targetValue>
202         <weightValue> 1.0 </weightValue>
203         <classificationValue> Alvo </classificationValue>
204            </attribute>
205            <attribute>
206         <metric> br.ufg.inf.metric.Reputação </metric>
207         <relationalOp> >= </relationalOp>
208         <targetValue> 80 </targetValue>
209         <weightValue> 1.0 </weightValue>
210         <classificationValue> Alvo </classificationValue>
211            </attribute>
212       </qosAttributes>
213       <!-- Carga estimada -->
214       <load>50</load>
215        </qosrequirement>
216    </requirements>
217 </processGraph>
```