Universidade Federal
do Rio de Janeiro
Escola Politécnica

COMPARTILHAMENTO DE RECURSOS EM NUVENS COLABORATIVAS

Hugo de Freitas Siqueira Sadok Menna Barreto

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores:
Miguel Elias Mitre Campista
Luís Henrique Maciel Kosmalski Costa

Rio de Janeiro
Setembro de 2016

COMPARTILHAMENTO DE RECURSOS EM NUVENS COLABORATIVAS

Hugo de Freitas Siqueira Sadok Menna Barreto

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

Hugo de Freitas Siqueira Sadok Menna Barreto

Orientador:

Prof. Miguel Elias Mitre Campista, D.Sc.

Orientador:

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Examinador:

Prof. Pedro Braconnot Velloso, Dr.

Examinador:

Prof. Daniel Sadoc Menasché, Ph.D.

Rio de Janeiro

Setembro de 2016

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

DEDICATÓRIA

Aos meus pais.

Aos meus irmãos.

Às minhas avós.

AGRADECIMENTO

Insuficientes agradecimentos aos meus pais Marcelo e Márcia por tudo que aprendi e pelo apoio que recebi até nos momentos mais difíceis. Agradeço ao meu irmão Bruno por me dar suporte e por me suportar. À minha irmã Luna que mesmo em início de vida já significa tanto para mim. Agradeço também às minhas avós Tina, que já se foi, e Carmen que até hoje me faz uma pessoa melhor.

Sou muito grato também aos professores Miguel e Luís, por me orientarem desde o início da graduação e por nunca desistirem de mim. Agradeço também pelas valiosas contribuições dadas a este trabalho. Obrigado aos professores membros da banca Pedro Velloso e Daniel Menasché por aceitarem o convite e se dedicarem a avaliar o meu trabalho.

Também sou grato aos colegas do GTA, presentes e passados pela ajuda imensurável e pela companhia. Uma lista não exaustiva inclui, em ordem alfabética: Daniel Neto, Fábio Oliveira, Felipe Ferreira, Filipe Barretto, Igor Alvarenga, Igor Quintanilha, Jean Fonseca, José Geraldo Júnior, Leonardo Cardoso, Lucas Maurício, Martin Lopez, Pedro Cruz, Pedro Henrique Guimarães, Pedro Pisa, Rodrigo Couto, Tatiana Sciammarella e Vitor Borges.

Agradeço aos meus amigos de graduação por estarem juntos desde o início, alguns desses com passagem também pelo GTA: Antonio Lobato, Carlos Oliveira, Eduardo Barretto, Lucas Cinelli, Thiago Cosenza e Ulisses Figueiredo.

Um muito obrigado também aos professores do DEL/UFRJ por todo o conhecimento que adquiri nesses últimos anos, essenciais para a realização deste trabalho.

RESUMO

O modelo de computação em nuvem se consolidou devido à possibilidade de elasticidade de recursos trazida aos usuários e às vantagens da agregação e economia de escala obtidas pelos provedores de nuvem. No entanto, muitos usuários da nuvem ainda dispõem de recursos computacionais próprios que alternam entre ociosos e insuficientes. Assim, as vantagens da elasticidade, características das nuvens, são muitas vezes usadas como auxílio à infraestrutura própria, nas chamadas nuvens híbridas. Uma alternativa mais barata é o uso de nuvens colaborativas. As nuvens colaborativas são formadas por recursos ociosos dos próprios usuários, o que permite o uso do sistema sem custos adicionais com provedores de nuvem. As vantagens da agregação de recursos, antes desfrutadas pelos provedores passa a beneficiar os usuários. No entanto, ao contribuir com seus recursos e utilizar recursos de outros, os usuários podem entrar em conflito. O sistema deve então garantir que os usuários de fato contribuam seus recursos ociosos, oferecendo vantagens na alocação quando a demanda for maior que a oferta. Nesse sentido, este projeto final propõe um modelo de cooperação, através de um cenário Bayesiano, em que os usuários constantemente decidem a quantidade de recursos que irão fornecer ou requisitar da nuvem colaborativa. O mecanismo proposto incentiva a participação ao mesmo tempo que garante o uso eficiente dos recursos. Simulações conduzidas neste projeto com o auxílio de dados de utilização da nuvem do Google mostram um grande aumento na parcela de requisições satisfeitas. Além disso, é possível observar como a configuração de parâmetros do mecanismo podem afetar a correlação entre recursos fornecidos e obtidos pelos usuários, assim como a estabilidade das alocações.

Palavras-Chave: alocação de recursos, teoria dos jogos, computação em nuvem, nuvem colaborativa.

ABSTRACT

The cloud computing model was consolidated as a consequence of the resource elasticity brought to end users, and the aggregation and advantages of economies of scale obtained by the providers. However, many cloud users still possess their own computational resources often alternating between idle and insufficient. As a consequence, the aggregation and elasticity advantages typical of clouds, are base building blocks of a collaborative cloud among those users. Collaborative clouds are composed of idle resources from users, which yields system utilization without additional costs with cloud providers. However, when users contribute their resources and use the resources of third parties, conflicts may arise. The system must be able to decide who to provide resources for, when the demand is greater than the offer, as well as ensure that users will indeed contribute with their idle resources. This work proposes a collaborative model, using a Bayesian setting, in which users constantly decide the amount of resources to either contribute or request from the collaborative cloud. The proposed mechanism guarantees the Individual Rationality and the Strict Pareto Efficiency. Simulations made in this project using Google cloud dataset show a non-negligible increase in the fraction of fulfilled requests. Moreover, it demonstrates how changes in the mechanism parameters affect the correlation between contributed and received resources, as well as the allocation stability.

Key-words: resource allocation, game theory, cloud computing, collaborative cloud.

SIGLAS

CPU - *Central Processing Unit* - Unidade Central de Processamento

GTA - Grupo de Teleinformática e Automação

IaaS - *Infrastructure as a Service* - Infraestrutura como Serviço

PaaS - *Platform as a Service* - Plataforma como Serviço

PID - Plataforma IaaS Distribuída

P2P - *Peer-to-peer* - Par-a-par

RAM - *Random-Access Memory* - Memória de Acesso Aleatório

SaaS - *Software as a Service* - Software como Serviço

UFRJ - Universidade Federal do Rio de Janeiro

VM - *Virtual Machine* - Máquina Virtual

Sumário

1 Introdução **1**

1.1 Proposta e Objetivos do Projeto 2

1.2 Metodologia 3

1.3 Organização do Texto 4

2 Fundamentos Teóricos **5**

2.1 Teoria dos Jogos 5

2.1.1 Jogos na Forma Normal 5

2.1.2 Jogos Repetidos 7

2.1.3 Jogos Bayesianos 9

2.1.4 Projeto de Mecanismo 9

3 Trabalhos Relacionados **11**

3.1 Divisão de Recursos e Competição 11

3.2 Federação de Provedores de Nuvem 13

4 Modelo Proposto **14**

4.1 Modelo 14

4.1.1 Conjunto de Resultados Possíveis 16

4.1.2 Tipos de Usuários 16

4.1.3 Utilidades dos Usuários 17

4.2 Função Mecanismo 20

4.2.1 Requisitos Formais 20

4.2.2 Credibilidades dos Usuários 21

4.2.3 Conjunto de Alocações Desejáveis 22

4.2.4 Definição da Função Mecanismo 23

4.2.5 Algoritmo para o Problema da Função Mecanismo 25
4.2.6 Demonstração dos Requisitos 30

5 Avaliação da Proposta **33**
5.1 Metodologia 33
5.2 Resultados 35

6 Conclusões **44**

Bibliografia **46**

Lista de Figuras

2.1 Utilidade dos prisioneiros de acordo com as ações escolhidas. O primeiro número é a utilidade do Prisioneiro 1 com a combinação das ações, enquanto que o segundo número é a utilidade do Prisioneiro 2. 6

3.1 Reciprocidade indireta formada por um ciclo de contribuição de recursos. O usuário A fornece recursos a B mas pode ser posteriormente recompensado por D. 12

4.1 Utilidade de um usuário i (u_i) conforme a alocação (o_i) para o caso em que ele precisa de 2 recursos extras ($\theta_i = 2$) e para o caso em que ele pode fornecer 2 recursos ($\theta_i = -2$). Os usuários são indiferentes a receber mais recursos do que precisam. 18

4.2 Alocações geradas pelo mecanismo a partir das ações e credibilidades dos usuários A até E. Ações positivas representam demanda, enquanto negativas representam oferta. Credibilidades negativas indicam que o usuário contribuiu mais no passado. 24

4.3 4 exemplos de alocação de recursos usando o mecanismo. A distribuição de credibilidade é a mesma para todos os exemplos, assim como as requisições por recursos. A cada exemplo, a quantidade de recursos sendo disponibilizada aumenta. 27

5.1 Cada evento representa uma transição de estado dos trabalhos e tarefas na nuvem do Google. Adaptado de [27]. 34

5.2 Necessidades de dois usuários arbitrários com comportamentos bem diferentes. Nos gráficos é possível ver os valores das necessidades de CPU e memória, bem como suas médias. As simulações consideram que os usuários possuem de 10% a 150% da média desses recursos. 35

5.3 Distribuição do total de recursos utilizados no sistema em todas as iterações. Observe que, para uma parcela de até 0,5 da média de CPU ou 0,4 da média de memória, o sistema está completamente sobrecarregado. 36
5.4 Distribuição das credibilidades de CPU e memória dos usuários no fim das alocações. Cada faixa representa um valor de δ diferente, com parcela de recursos atribuídos variando conforme a legenda. 37
5.5 Parcela de necessidades por recursos atendidas para diferentes parcelas de recursos atribuídos. Não há diferença significativa de necessidades atendidas para os diferentes valores de δ. 38
5.6 Comparação das necessidades atendidas ao usar ou não o mecanismo. A parcela de necessidades atendidas é consideravelmente maior ao usar o mecanismo, independentemente do δ utilizado. 39
5.7 Média das utilidades de todos os usuários em todas as iterações para diferentes parcelas de recursos atribuídos. Pela forma como a utilidade é definida, a sua média no sistema independe do valor de δ. 40
5.8 Correlação de Pearson entre os recursos contribuídos e recebidos no mecanismo. Observe como $\delta = 0{,}999$ resulta na melhor correlação para todas as parcelas de recursos atribuídos. 40
5.9 Mediana das Autocorrelações *lag-1* das alocações de CPU e memória de todos os usuários. 42

Lista de Tabelas

4.1 Resumo das notações usadas nas definições. 19

Capítulo 1

Introdução

O paradigma de computação em nuvem move o provisionamento de recursos computacionais para centros de dados acessíveis pela rede. Além das vantagens trazidas pela terceirização da operação da infraestrutura, a nuvem fornece aos usuários maior flexibilização do dimensionamento da infraestrutura necessária às suas aplicações [1, 2]. Sem a nuvem, os usuários precisam dimensionar seus recursos para atender os possíveis picos de demanda, causando ociosidade na maior parte do tempo. Além disso, os provedores de nuvem conseguem oferecer serviços a preços competitivos por se beneficiarem da agregação e heterogeneidade de uso de vários usuários [2]. Entretanto, mesmo com o cenário de crescente popularização da computação em nuvem, muitos usuários (empresas, laboratórios, etc.) possuem sua própria infraestrutura computacional, seja por possuir uma base já instalada, seja por uma política de aquisição de equipamentos da instituição. A computação em nuvem neste caso se torna uma forma desses usuários obterem mais recursos quando necessário, ou compartilhar recursos de forma eficiente.

Algumas arquiteturas em nuvem são utilizadas para prover elasticidade, dentre elas está a nuvem híbrida [3]. Nuvens híbridas integram a infraestrutura local à nuvem, de forma que os usuários possam obter recursos adicionais em casos de pico de demanda. Uma pesquisa feita em 2016 com 1060 profissionais de TI mostrou que 71% usavam uma nuvem híbrida [4]. Embora os usuários possam obter recursos extras em caso de pico, é comum a presença de recursos ociosos. Uma alternativa à nuvem híbrida é o paradigma de nuvem colaborativa. Uma nuvem colaborativa é

formada por máquinas dedicadas, cedidas pelos próprios usuários da nuvem colaborativa. Esses usuários frequentemente estão geograficamente separados, dando um aspecto distribuído à nuvem. Os integrantes possuem acesso aos recursos ociosos dos demais, em seus momentos de pico, e contribuem com seus próprios recursos, quando ociosos. A nuvem colaborativa traz os benefícios da agregação e da heterogeneidade de uso, características das nuvens tradicionais, ao mesmo tempo em que permite que os usuários usem sua própria infraestrutura, como nas nuvens híbridas. Diferente do conceito de grades de computação, em que os usuários combinam recursos com um objetivo comum [5], os usuários da nuvem colaborativa podem ter objetivos distintos. Um exemplo de nuvem colaborativa é o Projeto PID [6].

Embora as vantagens da agregação de recursos na nuvem colaborativa sejam facilmente vislumbradas, ainda é essencial definir como os recursos da nuvem devem ser atribuídos a cada usuário. Igualmente essencial para o funcionamento da nuvem colaborativa é a necessidade dos usuários contribuírem com seus recursos ociosos.

1.1 Proposta e Objetivos do Projeto

O objetivo deste trabalho é propor uma solução que incentive a colaboração dos usuários com seus recursos ociosos, como CPU e memória, e escalone recursos entre os usuários em caso de sobrecarga da nuvem. No entanto, o projeto da política de atribuição de recursos de acordo com a doação de recursos deve ser feito com cuidado. Dependendo de como for a solução, pode ser difícil ou até mesmo impossível a entrada de novos usuários, já que estes nunca contribuíram com o sistema. Assim, o mecanismo proposto neste projeto define o escalonamento de recursos em caso de sobrecarga sem obstruir a entrada de novos usuários, mas dando prioridade aos usuários com as melhores contribuições no passado. Os objetivos específicos do mecanismo proposto são:

1. incentivar a contribuição de todos os recursos ociosos;
2. permitir que os usuários possam ter acesso aos próprios recursos, assim que desejarem; e
3. permitir a entrada de novos usuários.

Deve-se notar que o objetivo da proposta não é um escalonador de recursos no sentido tradicional do termo. Ao invés de escalonar máquinas virtuais ou tarefas entre servidores, o objetivo deste trabalho é escalonar recursos entre os usuários. A decisão de onde esses recursos serão alocados deve ser feita por outro escalonador, que pode ser escolhido de acordo com as características da nuvem. Características relevantes incluem distribuição geográfica e tipo comum das tarefas. Por exemplo, uma nuvem com grande espalhamento geográfico, onde a maior parte das aplicações são servidores *web*, pode se beneficiar de alocações que espalhem os recursos, aumentando assim a resiliência dessas aplicações.

1.2 Metodologia

Este projeto propõe um modelo onde cada usuário possui alguma necessidade por recursos e atinge um nível de satisfação diferente dependendo do que obtiver do sistema. A ideia básica do mecanismo se apoia na teoria dos jogos.

As requisições dos usuários por recursos e o mecanismo de alocação são modelados como jogos repetidos. Em cada iteração os usuários devem informar a quantidade de recursos que desejam fornecer ou pedir ao sistema. Baseado nas ofertas e demandas recebidas, o mecanismo pode escolher a quantidade de recursos que de fato irá fornecer a cada usuário. No entanto, a decisão do mecanismo é guiada não só pelos desejos dos usuários mas também pela credibilidade de cada um.

As credibilidades dos usuários são definidas em função das contribuições e requisições passadas. Quanto mais recursos um usuário forneceu ao sistema no passado melhor será sua credibilidade e mais fácil será dele obter recursos quando desejar. Para facilitar a recuperação da credibilidade, quando existem mais recursos sendo fornecidos ao sistema do que requisitados, os recursos dos usuários de pior credibilidade são preferencialmente utilizados.

Vale notar que as credibilidades são definidas com valores positivos e negativos, de forma que a média das credibilidades de todos os usuários em qualquer iteração é zero. Isso auxilia na entrada de novos usuários, que serão integrados ao sistema com

credibilidade zero. Além disso, como forma de reagir a mudanças no comportamento dos usuários, as contribuições passadas são amortecidas com o tempo no cálculo da credibilidade. O modelo também inclui uma função utilidade para representar a satisfação dos usuários com os recursos obtidos. Quanto mais recursos um usuário obtém do sistema, maior é sua utilidade. Contudo, os usuários são indiferentes ao receber mais recursos do que de fato precisam.

Após a apresentação e avaliação teórica do mecanismo, seu desempenho é analisado com o uso de um conjunto de dados da nuvem do Google. Os dados possuem todas as requisições por recursos dos usuários da nuvem do Google durante 29 dias. A partir disso foi possível inferir as necessidades desses usuários. Além disso, como este trabalho trata de uma nuvem colaborativa, cada um dos usuários deve possuir seus próprios recursos, que podem então ser fornecidos à nuvem. Para simular esse comportamento, considerou-se que cada usuário possui uma quantidade de recursos em relação a média de suas necessidades. Foram feitas avaliações com quantidades de recursos variando de 10% a 150% da média das necessidades. O mecanismo proposto garante que nenhum usuário obterá vantagens abandonando o mecanismo ao mesmo tempo que maximiza a soma das utilidades de todos os usuários. Os resultados das simulações do modelo proposto mostram um aumento do número de requisições satisfeitas de menos de 30%, sem a utilização do mecanismo, para cerca de 98%, no caso em que os usuários receberam recursos equivalentes a 90% da média das necessidades.

1.3 Organização do Texto

O restante do trabalho está organizado da seguinte forma. O Capítulo 2 introduz os conceitos básicos necessários à compreensão da proposta. No Capítulo 3 são apresentados os principais trabalhos relacionados encontrados na literatura. O Capítulo 4 mostra a formulação do modelo de competição por recursos e o mecanismo proposto para o sistema. O Capítulo 5 mostra uma avaliação do mecanismo usando dados da nuvem do Google. Finalmente, o Capítulo 6 conclui o trabalho e aponta direções futuras.

Capítulo 2

Fundamentos Teóricos

Este capítulo apresenta conceitos preliminares, notações e nomenclaturas utilizadas ao longo deste trabalho.

2.1 Teoria dos Jogos

A Teoria dos Jogos é o estudo de modelos de conflito ou cooperação entre agentes inteligentes, que devem tomar decisões com o intuito de maximizar seu retorno [7]. Nesse sentido, o compartilhamento de recursos em uma nuvem distribuída pode ser visto como um caso a ser analisado com o uso da teoria dos jogos. A nuvem colaborativa é formada por usuários que procuram sempre atender suas demandas por recursos e que entram em conflito quando precisam de mais recursos do que a nuvem pode oferecer. Esta seção discute conceitos e definições básicos de teoria dos jogos que serão úteis na descrição do modelo e do mecanismo proposto para o compartilhamento de recursos.[1]

2.1.1 Jogos na Forma Normal

Os cenários discutidos em Teoria dos Jogos são chamados "jogos". Os jogos são formados por jogadores, ações e utilidades. A forma mais simples de definir um jogo é a forma normal $(\mathcal{N}, \mathcal{A}, u)$ [10] em que:

- $\mathcal{N} = \{1, \ldots, N\}$ é um conjunto finito de jogadores, indexado por $i \in [1, N]$.

[1] Ao leitor interessado em se aprofundar mais em teoria dos jogos recomenda-se os cursos [8, 9], além das referências citadas ao longo deste capítulo.

- $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_N$ onde $\mathcal{A}_i$ é um conjunto de ações disponíveis para o jogador i. A escolha da ação de todos os jogadores é representada por $a = (a_1, \ldots, a_n) \in \mathcal{A}$.

- $u = (u_1, \ldots, u_N)$ onde $u_i : \mathcal{A} \mapsto \mathbb{R}$, é a função utilidade do jogador i, a qual define a satisfação deste com as ações escolhidas por todos os jogadores.

Um jogo na forma normal é comumente exemplificado com o Dilema dos Prisioneiros. Nesse dilema, existem dois jogadores (prisioneiros) com duas opções de ação: confessar o crime ou mentir. Dependendo das ações escolhidas, os prisioneiros terão satisfações diferentes, representadas pelo resultado das suas respectivas funções utilidades. Quanto maior for a satisfação, maior o valor retornado pela função utilidade. A Figura 2.1 mostra as satisfações dos prisioneiros para diferentes combinações de ações. Tome como exemplo o quadrante inferior esquerdo, este é o caso em que o Prisioneiro 1 mente e o Prisioneiro 2 confessa. Nesse caso o resultado da função utilidade para o Prisioneiro 1 é -5 e para o Prisioneiro 2 é -1. Observe que a melhor estratégia a ser usada por ambos os prisioneiros é confessar, independente da escolha do outro. No entanto, se os prisioneiros escolherem esta estratégia, eles estarão no quadrante superior esquerdo, recebendo utilidade -4.

		Prisioneiro 2	
		Confessar	Mentir
Prisioneiro 1	Confessar	-4,-4	-1,-5
	Mentir	-5,-1	-2,-2

Figura 2.1: Utilidade dos prisioneiros de acordo com as ações escolhidas. O primeiro número é a utilidade do Prisioneiro 1 com a combinação das ações, enquanto que o segundo número é a utilidade do Prisioneiro 2.

Este exemplo ilustra como boas estratégias, do ponto de vista individual, podem gerar resultados não ótimos. Ao pensar individualmente, os prisioneiros acabam levando a uma situação de equilíbrio que é pior do que o caso em que os dois mentem.[2] Esta situação de equilíbrio é chamada de Equilíbrio de Nash. Formalmente, o Equilíbrio de Nash pode ser definido com auxílio do conceito de melhor resposta [10].

Definição 2.1. *Dado um jogo* $(\mathcal{N}, \mathcal{A}, u)$*, seja* a_{-i} *uma seleção de ações de todos os jogadores excluindo* i*, ou seja,* $a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$*. Uma ação* a_i^* *é uma* ***melhor resposta*** *a* a_{-i} *se, e somente se,* $\forall a_i \in \mathcal{A}_i, u_i(a_i^*, a_{-i}) \geq u_i(a_i, a_{-i})$*.*

Em outras palavras, uma ação de um jogador é uma melhor resposta às ações dos outros se esta sempre trouxer a este jogador uma utilidade melhor, ou tão boa quanto, as demais ações possíveis. No exemplo, dado que o Prisioneiro 2 escolheu confessar, confessar é uma melhor resposta para o Prisioneiro 1. Do mesmo modo, dado que o Prisioneiro 2 escolheu mentir, confessar também é uma melhor resposta para o Prisioneiro 1. Usando o conceito de melhor resposta é possível então definir o Equilíbrio de Nash.

Definição 2.2. *Uma seleção de ações* $a = (a_1, \ldots, a_n)$ *é um* ***Equilíbrio de Nash*** *se, e somente se, para todo* i*,* a_i *é uma melhor resposta a* a_{-i}*.*

No exemplo, os dois confessarem é um Equilíbrio de Nash já que é uma melhor resposta a qualquer ação do outro prisioneiro.

2.1.2 Jogos Repetidos

Nem todos os jogos funcionam a partir de apenas uma decisão de cada jogador. O mesmo jogo pode ser repetido um número arbitrário de vezes. Considerando-se que o jogo será repetido k vezes e que os jogadores conhecem k, surgem duas questões principais:

- Como calcular a utilidade dos jogadores;

[2]Vale notar as semelhanças entre o Dilema dos Prisioneiros e o processo de delação premiada, bastante comentado na mídia atualmente devido à repercussão da Operação Lava Jato. `https://www.nexojornal.com.br/expresso/2016/05/25/Como-a-Teoria-dos-Jogos-explica-o-comportamento-dos-delatores-da-Lava-Jato`

- Como a existência de várias iterações do jogo altera as estratégias.

No caso de um número finito de iterações, a soma das utilidades de cada jogador em todas as iterações pode ser usada para calcular a utilidade total obtida por cada jogador. Entretanto, no caso de infinitas iterações, não faz sentido utilizar a soma das utilidades já que esta em geral também será infinita. Nesse caso, a recompensa futura descontada é uma forma comum de medir a utilidade de cada jogador em jogos repetidos infinitamente [10].

Definição 2.3. *Dada uma sequência infinita de utilidades* $u_i(0), u_i(1), \ldots$ *para o jogador* i*, e um fator de desconto* $0 < \delta < 1$*, a* ***recompensa futura descontada*** *de* i *é dada por* $\sum_{j=0}^{\infty} \delta^j u_i(j)$*.*

Uma interpretação da recompensa futura descontada é que, em geral, as utilidades perto do presente tem valor maior que as utilidades futuras. Em um caso extremo, quando δ se aproxima de 0, os jogadores só se importam com as utilidades presentes. Quanto mais δ se aproxima de 1, mais os valores futuros são considerados.

As estratégias dos jogadores também podem mudar com a presença de várias rodadas. Considere novamente o dilema dos prisioneiros, representado na Figura 2.1 na página 6. Foi visto que confessar era o equilíbrio de Nash, no entanto, caso o mesmo jogo ocorra infinitas vezes, pode ser possível que os jogadores adotem outra estratégia. Se o fator de desconto δ for grande o suficiente, os valores futuros passam a ter maior importância podendo existir um número infinito de equilíbrios de Nash possíveis.

Um dos equilíbrios possíveis é a estratégia ***tit for tat*** (olho por olho) [11]. A estratégia consiste em cooperar no primeiro movimento e então fazer exatamente o que o adversário fez na rodada anterior. Uma estratégia alternativa, que também configura um equilíbrio, é a ***grim trigger*** (gatilho severo) [12], a qual também consiste dos jogadores começarem cooperando um com o outro. No entanto, caso em qualquer rodada o adversário deixe de cooperar, o jogador também deixará de cooperar em todas as rodadas futuras.

No dilema dos prisioneiros isso equivaleria aos prisioneiros mentirem em todos os jogos, mas caso algum deles decida confessar, o outro passa a confessar para sempre. Isso mostra como a presença de várias iterações pode trazer novos equilíbrios ao jogo.

2.1.3 Jogos Bayesianos

Existem casos onde a forma do jogo não se dá de maneira tão clara como foi apresentado até agora. Um jogo Bayesiano é uma forma de representar a incerteza dos jogadores quanto ao jogo que participam. Um jogo Bayesiano pode ser definido como uma tupla $(\mathcal{N}, \mathcal{A}, \Theta, p, u)$ [10] onde:

- $\mathcal{N}$ é um conjunto finito de jogadores, onde $|\mathcal{N}| = N$.
- $\mathcal{A} = A_1 \times \ldots \times A_N$ onde A_i é um conjunto de ações disponíveis para o jogador i.
- $\Theta = \Theta_1 \times \ldots \times \Theta_N$ onde Θ_i é um conjunto de tipos possíveis para o jogador i.
- p é uma distribuição de probabilidade dos tipos Θ, conhecida por todos os jogadores.
- $u = (u_1, \ldots, u_N)$ onde $u_i : \mathcal{A} \times \Theta \mapsto \mathbb{R}$ é a função utilidade do jogador i.

Diferente dos jogos na forma normal, agora são incluídos os tipos de cada jogador (Θ), junto a uma distribuição de probabilidade destes tipos (p). Além disso, as funções utilidade agora dependem não só das ações de todos os jogadores, mas também dos tipos dos jogadores. A noção de tipo é bastante ampla mas intuitivamente representa o conhecimento privado de cada jogador, de forma que somente os usuários conhecem seus próprios tipos. Podendo ser um candidato, no caso de uma eleição, ou a quantidade de recursos que o jogador precisa, no caso de um sistema de compartilhamento de recursos. Dessa forma, é natural que a utilidade de cada jogador dependa da informação dos tipos.

2.1.4 Projeto de Mecanismo

Existem ainda casos, como o deste trabalho, onde há a possibilidade de projetar o jogo apresentado aos jogadores. Neste caso, separa-se o jogo em um cenário e um mecanismo. Ao mecanismo cabe somente definir possíveis ações que poderão ser

tomadas pelos jogadores e escolher uma saída baseada nessas ações [13]. Exemplos comuns de mecanismo são sistemas de eleição. Um mecanismo de eleição bastante comum permite a cada eleitor a escolha de um candidato, escolhendo como vencedor o candidato com a maior quantidade de votos. Os candidatos possíveis são as ações fornecidas aos eleitores e o candidato vencedor é a saída do mecanismo. Cada eleitor tem como tipo uma ordem de preferência entre os candidatos possíveis e obterá uma utilidade diferente dependendo de qual for a saída do mecanismo. O maior desafio no projeto de um mecanismo é a escolha da função que produzirá uma saída adequada baseada nas ações dos jogadores.

Muitos mecanismos também possuem na saída uma determinação de pagamentos que devem ser feitos entre os jogadores. Isso pode ajudar a compensar jogadores prejudicados pela decisão do mecanismo. No entanto, isso assume a presença de **utilidade transferível**, ou seja, a utilidade dos jogadores pode ser livremente redistribuída entre eles. Em geral, essa hipótese é válida quando a utilidade é representada por algum tipo de moeda [10].

Este trabalho propõe um mecanismo com um cenário baseado em jogos Bayesianos. As várias iterações do mecanismo caracterizam um jogo repetido. Os detalhes do mecanismo serão apresentados no Capítulo 4.

Capítulo 3

Trabalhos Relacionados

Este capítulo apresenta os principais trabalhos relacionados encontrados na literatura. Os trabalhos foram divididos entre os que avaliam o compartilhamento de recursos de uma forma mais generalizada e os que avaliam a federação de provedores de nuvem.

3.1 Divisão de Recursos e Competição

O problema de compartilhamento de recursos e de competição tratado neste trabalho assemelha-se ao problema enfrentado por sistemas par-a-par (*peer-to-peer* - P2P). Do mesmo modo que em sistemas par-a-par, os usuários podem fornecer e receber recursos dos outros nós participantes. No entanto, o comportamento egoísta dos nós pode comprometer o desempenho do sistema.

Buragohain *et al.* [14] analisaram o problema das redes par-a-par usando ideias de teoria dos jogos. A solução buscou aumentar a reciprocidade de contribuição de recursos entre pares de nós, ao mesmo tempo lidando com o problema de falta de confiança entre os usuários. Foram utilizadas técnicas de jogos com aprendizado, de forma que os usuários podiam punir outros usuários egoístas com base nas contribuições passadas. Um problema da proposta é a falta de conhecimento dos nós que entram no sistema, os quais não sabem em que nós confiar até que tenham tido um certo número de iterações com a rede. No contexto de redes *ad hoc*, Velloso *et al.* [15] trataram o problema de falta de confiança com auxílio de troca de informações. Na

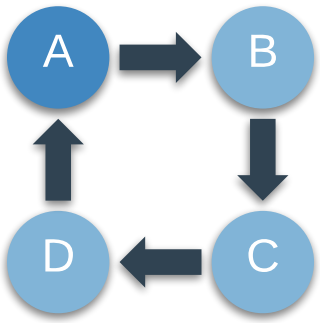


Figura 3.1: Reciprocidade indireta formada por um ciclo de contribuição de recursos. O usuário A fornece recursos a B mas pode ser posteriormente recompensado por D.

proposta, cada nó avalia os vizinhos com base não só nas suas observações, mas também nas avaliações obtidas dos demais vizinhos. Alternativamente, Andrade *et al.* [16] e Cirne *et al.* [17] trataram do problema de colaboração em grades. A proposta usou o conceito de rede de favores, em que cada usuário guarda a quantidade de recursos recebida e fornecida a cada outro usuário. Os usuários então decidem a quantidade de recursos que devem contribuir aos demais, baseados nesses valores. Falcão *et al.* [18] também propuseram uma solução usando rede de favores mas para o cenário de uma federação de provedores privados de nuvem. Além disso, como forma de prover justiça, propuseram que os usuários limitassem a quantidade de recursos providos aos demais, de acordo com a quantidade de recursos recebidos. Um problema comum dos trabalhos citados até então é a ausência da chamada reciprocidade indireta. Como exemplificado na Figura 3.1, quando há reciprocidade indireta, um usuário pode contribuir recursos para um usuário, mas ser recompensado por outro. Menasché *et al.* [19] propuseram formas de melhorar o desempenho das redes com reciprocidade direta, ao reduzir ciclos de contribuição. No entanto, o uso de banda dos usuários pode chegar a dobrar em relação às redes com a presença de reciprocidade indireta.

Neste trabalho, como a arquitetura da nuvem exige a presença de uma entidade central, onde os recursos podem ser requisitados, essa entidade central também pode ser usada para reger as alocações de recursos, permitindo que haja reciprocidade indireta. A entidade central também permite que os novos usuários sofram menor impacto dos usuários mal comportados, uma vez que os comportamentos passados

de todos os usuários são conhecidos pela entidade.

3.2 Federação de Provedores de Nuvem

Existem trabalhos que tratam da possibilidade de provedores de nuvem formarem algum tipo de coalizão possibilitando o aumento da utilidade global. Lu *et al.* [20] propuseram uma federação de provedores de nuvem que cooperam para atender usuários internos e externos. A proposta procura ajudar os provedores a decidir a quantidade de recursos que eles devem compartilhar e qual a melhor coalizão para cada provedor. Entretanto, a proposta apenas sugere ações aos provedores: cada provedor ainda é responsável por definir um preço para os recursos que compartilha. Samaan [21] focou em definir preços e a quantidade de recursos que o provedor deve contribuir para um mercado de excedentes (*spot market*). Para isso, foi usado um modelo com jogo repetido. Para garantir o equilíbrio no jogo foi utilizada a estratégia de *grim trigger*[1]. Alternativamente, Niyato *et al.* [22] focaram na análise de formação de coalizão entre os provedores de nuvem. Foram analisadas combinações de coalizão entre três provedores e foi constatado que se houver custos para a formação da coalizão, alguns provedores podem não obter vantagem ao participar. O modelo da proposta também exigiu o uso de utilidade transferível[2].

Como neste trabalho não há utilidade transferível, não é necessário que os usuários definam um preço para os recursos que fornecem. Além disso, o mecanismo proposto incentiva a verdade e portanto não são necessárias medidas severas como *grim trigger*.

[1] A estratégia *grim trigger* foi explicada na Seção 2.1.2 na página 8.

[2] O conceito de utilidade transferível foi introduzido na Seção 2.1.4 na página 10.

Capítulo 4

Modelo Proposto

Este capítulo apresenta o modelo de compartilhamento de recursos e o mecanismo utilizado para a alocação. O mecanismo possibilita que cada usuário sempre possa ter acesso a todos os recursos que possui quando desejar. Ainda, o sistema incentiva a entrada de novos usuários e a contribuição dos membros participantes.

Em cada iteração do mecanismo, os usuários têm a decisão de fornecer ou requisitar recursos de acordo com suas próprias necessidades. Se um usuário tem recursos ociosos, este pode fornecê-los ao sistema. Se um usuário precisa de mais recursos do que possui, este pede os recursos extras, mantendo o controle sobre os recursos que já possui.

Os incentivos dados aos usuários são discutidos em termos de suas satisfações, representadas por suas respectivas funções de utilidade. Intuitivamente, para que os usuários façam parte do mecanismo, a satisfação que obtêm ao participar deve ser maior do que seria caso não participassem. Do mesmo modo, para que um usuário contribua recursos, a satisfação que este espera obter ao contribuir deve ser maior do que o caso em que ele não contribui seus recursos ociosos.

4.1 Modelo

No modelo proposto, a utilização do sistema é modelada em intervalos de tempo de mesma duração. Isso permite que as alocações de recursos passadas sejam comparáveis, o que posteriormente será útil na definição das credibilidades dos usuários.

Esses intervalos devem ser determinados e podem ser da ordem de minutos. Os usuários decidem a quantidade de recursos a compartilhar ou requisitar em cada intervalo de tempo. O mecanismo de compartilhamento assume um único tipo de recurso. Havendo mais de um tipo de recurso, deve ser utilizado um mecanismo independente para cada tipo. Poderia ser proposto um sistema que considerasse todos os tipos de recursos ao mesmo tempo, porém seria necessário determinar uma relação de valor entre cada tipo, o que nem sempre é possível.

Define-se então um cenário de Jogo Bayesiano que ocorrerá em cada intervalo, cujas notações utilizadas nas definições estão resumidas na Tabela 4.1 na página 19. O cenário contém as variáveis que não podem ser controladas pelo mecanismo e é definido por uma tupla $(\mathcal{N}, \mathcal{O}, \Theta, p, u)$ [10], onde:

- $\mathcal{N}$ é um conjunto finito de usuários tal que $|\mathcal{N}| = N$.
- $\mathcal{O}$ é um conjunto de resultados possíveis (todas as alocações de recursos possíveis).
- $\Theta = \Theta_1 \times \ldots \times \Theta_N$ onde Θ_i é um conjunto de tipos possíveis para o usuário i. O tipo de usuário representa as informações privadas de cada usuário, enquanto Θ representa as combinações de tipos de todos os usuários.
- p é uma distribuição de probabilidade dos tipos Θ, conhecida por todos os usuários.
- $u = (u_1, \ldots, u_N)$ onde $u_i : \mathcal{O} \times \Theta \mapsto \mathbb{R}$ é a função utilidade do usuário i que define a satisfação deste. A satisfação depende de uma alocação de recursos em $\mathcal{O}$ e de uma combinação de tipos em Θ.

Para este cenário, define-se um mecanismo que representa o sistema e é responsável por definir a alocação de recursos em cada intervalo desde que limitado ao conjunto $\mathcal{O}$. O mecanismo é definido pelo par $(\mathcal{A}, \mathcal{M})$, onde:

- $\mathcal{A} = A_1 \times \ldots \times A_N$: onde A_i é um conjunto de ações disponíveis para o usuário i.
- $\mathcal{M} : \mathcal{A} \mapsto \mathcal{O}$: é a função mecanismo que mapeia as ações dos usuários em $\mathcal{A}$ em uma alocação de recursos em $\mathcal{O}$.

Juntos, o cenário e o mecanismo formam um jogo completo. De posse da estrutura do jogo, os elementos podem ser definidos com mais detalhes.

4.1.1 Conjunto de Resultados Possíveis

O conjunto $\mathcal{O}$ define as alocações de recursos possíveis. Uma alocação possível deve fornecer recursos de certos usuários a outros usuários sem que novos recursos sejam criados. No entanto, é possível que o sistema aja de forma ineficiente obtendo recursos de usuários e não os repassando a outros. Além disso, um usuário não pode fornecer mais recursos do que possui. Os termos positivos da alocação representam recursos sendo fornecidos ao usuário, enquanto termos negativos representam recursos retirados do usuário. Considerando que r_i representa a quantidade total de recursos que o usuário i possui ($r_i \in \mathbb{N}$), $\mathcal{O}$ pode ser definida como:

$$\mathcal{O} = \left\{ (o_1, \ldots, o_N) \in \mathbb{Z}^N \middle| (\forall i \in \mathcal{N}, o_i \geq -r_i) \wedge \sum_{i \in \mathcal{N}} o_i \leq 0 \right\}. \tag{4.1}$$

Note que na Equação 4.1, $o_i \geq -r_i$ porque os usuários não podem contribuir mais recursos do que possuem. Além disso, $\sum_{i \in \mathcal{N}} o_i \leq 0$ já que, se a soma das alocações fosse positiva, haveriam mais usuários recebendo recursos do que fornecendo. O caso em que $\sum_{i \in \mathcal{N}} o_i < 0$ ainda é possível uma vez que o mecanismo poderia bloquear recursos de um usuário sem necessariamente fornecê-los a outros.

4.1.2 Tipos de Usuários

O tipo de um usuário i, $\theta_i \in \Theta_i$, indica a quantidade de recursos que i precisa receber. Se $\theta_i > 0$, o usuário precisa de mais recursos do que possui. Caso $\theta_i < 0$, o usuário possui recursos excedentes e pode contribuir recursos para outros usuários. Formalmente, o conjunto de tipos possíveis de um usuário i, $\forall i \in \mathcal{N}$, é:

$$\Theta_i = \{\theta_i \in \mathbb{Z} | \theta_i \geq -r_i\}, \tag{4.2}$$

de forma que os usuários não podem desejar fornecer mais recursos do que de fato possuem (r_i), mas podem desejar receber um número arbitrário de recursos. O mecanismo a ser definido é direto, ou seja, o conjunto de ações possíveis $\mathcal{A}$ é o mesmo que o de tipos possíveis Θ.

4.1.3 Utilidades dos Usuários

Para a definição das funções utilidade dos usuários, é conveniente usar o conceito de independência de utilidade condicionada [23], definido como se segue:

Definição 4.1. *Um Jogo Bayesiano possui* ***independência de utilidade condicionada*** *se, para todos os usuários $i \in \mathcal{N}$, para todos os resultados $\boldsymbol{o} \in \mathcal{O}$ e para todos os pares de tipo $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$, com $\theta_i = \theta'_i$, acontece que $u_i(\boldsymbol{o}, \boldsymbol{\theta}) = u_i(\boldsymbol{o}, \boldsymbol{\theta}')$.*

Em outras palavras, a independência de utilidade condicionada assume que a utilidade de um usuário i (u_i) depende somente do seu tipo (θ_i) e não dos tipos de todos os usuários como definido anteriormente. Fazendo isso, a utilidade de i pode ser escrita como $u_i(\boldsymbol{o}, \theta_i)$.

A função utilidade de cada usuário deve representar a satisfação destes com a alocação escolhida. Existem basicamente dois tipos de usuários: usuários que precisam de recursos ($\theta_i > 0$) e usuários com recursos ociosos ($\theta_i < 0$). Um usuário que precisa de recursos tem utilidade positiva para qualquer recurso extra obtido, desde que não ultrapasse sua necessidade. Caso ultrapasse, qualquer recurso recebido será indiferente ao usuário. Um usuário com recursos ociosos será indiferente em fornecer recursos desde que ainda tenha os recursos que precisa. Considerando um resultado possível $\boldsymbol{o} \in \mathcal{O}$ e o tipo do usuário i, $\theta_i \in \Theta_i$, a função utilidade de cada usuário pode ser resumida como:

$$u_i(\boldsymbol{o}, \theta_i) = \begin{cases} \min(o_i, \theta_i), \text{ para } \theta_i \geq 0. \\ \min(0, o_i - \theta_i), \text{ para } \theta_i < 0. \end{cases} \tag{4.3}$$

Observe como na Equação 4.3, usuários que precisam de recursos ($\theta_i > 0$) terão utilidades negativas caso $o_i < 0$ e utilidades positivas caso $o_i > 0$. Quando estes usuários recebem mais recursos do que precisam ($o_i > \theta_i$) suas utilidades deixam de aumentar, sendo limitadas por θ_i. Já os usuários com recursos ociosos ($\theta_i < 0$) não conseguem utilidade positiva, mas terão utilidade negativa caso $o_i < \theta_i$, ou seja, caso sejam forçados pelo mecanismo a contribuir com mais recursos do que gostariam. Vale notar que, para quando $\theta_i = 0$, os usuários são indiferentes a receber recursos

mas terão utilidade negativa caso contribuam recursos ao sistema. Um exemplo do resultado da função utilidade para valores diferentes de o_i pode ser visto na Figura 4.1.

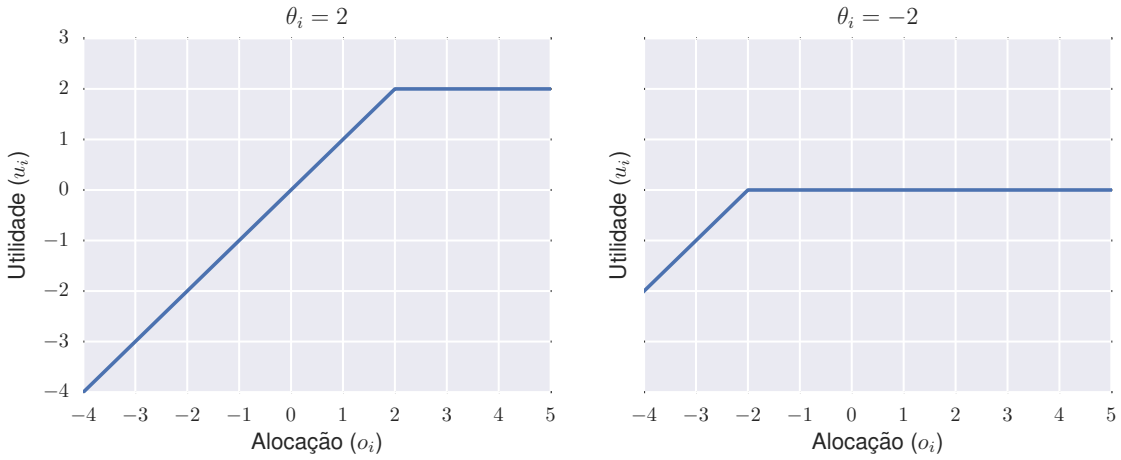


Figura 4.1: Utilidade de um usuário i (u_i) conforme a alocação (o_i) para o caso em que ele precisa de 2 recursos extras ($\theta_i = 2$) e para o caso em que ele pode fornecer 2 recursos ($\theta_i = -2$). Os usuários são indiferentes a receber mais recursos do que precisam.

Embora as utilidades dos usuários que contribuem recursos sejam nulas, espera-se que os usuários escolham estratégias que maximizem sua função utilidade, não só no intervalo atual mas na expectativa dos intervalos futuros. Tendo isso em vista, o mecanismo deve ser capaz de prover incentivos aos usuários de modo que contribuir recursos aumente a expectativa das utilidades futuras. A discussão sobre as estratégias não faz parte da definição do modelo e será vista mais à frente.

Notação	**Descrição**
N	Número de usuários
$\mathcal{N}$	Conjunto de todos os usuários
$\mathcal{O}$	Conjunto de alocações possíveis
$\boldsymbol{o}$	Uma alocação de recursos em $\mathcal{O}$ para todos os usuários
o_i	Quantidade de recursos alocados para i
Θ	Combinação de tipos possíveis de todos os usuários
Θ_i	Conjunto de tipos possíveis para o usuário i
$\boldsymbol{\theta}$	Escolha de tipos de todos os usuários
θ_i	Um tipo de i em Θ_i
r_i	Quantidade de recursos que i possui
p	Distribuição de probabilidade dos tipos em Θ
u	Funções utilidade dos usuários ($u = (u_1, \ldots, u_N)$)
$u_i(o, \theta_i)$	Função utilidade do usuário i
$\mathcal{A}$	Combinação de ações de todos os usuários
A_i	Conjunto de ações possíveis para o usuário i
$\boldsymbol{a}$	Ações escolhidas por todos os usuários
a_i	Ação escolhida por i
$\mathcal{M}(a)$	Função Mecanismo

Tabela 4.1: Resumo das notações usadas nas definições.

4.2 Função Mecanismo

Como visto na Seção 4.1, a função mecanismo decide uma alocação de recursos válida a partir dos desejos dos usuários de receber ou contribuir. Para que haja um incentivo à contribuição de recursos ociosos, o mecanismo deve garantir que essa alocação privilegie usuários com maior contribuição no passado. Para isso é usado um sistema de credibilidade, para que a contribuição passada de um usuário se reflita em sua credibilidade, sabendo que quanto melhor for a credibilidade do usuário, mais fácil será para que ele consiga recursos adicionais. No entanto, deve haver cautela para que o mecanismo também possibilite a entrada de novos usuários. Os requisitos do mecanismo serão formalizados a seguir.

4.2.1 Requisitos Formais

O primeiro requisito do mecanismo é o incentivo à participação. Pela forma como a função utilidade foi definida na Equação 4.3, utilidades positivas indicam que o usuário obteve vantagem ao participar do mecanismo. Já utilidades negativas indicam que a participação no mecanismo prejudicou o usuário. Ao garantir que os usuários não obterão utilidades negativas, isso garante que participar do mecanismo é, pelo menos, tão bom quanto não participar. Formalmente, quando isso ocorre, o mecanismo satisfaz a Racionalidade Individual [24, 25].

Definição 4.2. *Um mecanismo* $(\mathcal{A}, \mathcal{M})$ *satisfaz a* ***Racionalidade Individual*** *se para todo* i *e* $\boldsymbol{\theta}$*,* $u_i(\mathcal{M}(\boldsymbol{\theta}), \theta_i) \geq 0$*.*

$\mathcal{M}(\boldsymbol{\theta})$ representa a resposta do mecanismo para os tipos de todos os usuários. A Racionalidade Individual implica que nenhum usuário desejará abandonar o mecanismo.

Teorema 4.1. *Dado o mecanismo* $(\mathcal{A}, \mathcal{M})$*, que atua no cenário descrito na Seção 4.1. Para cada iteração, o mecanismo satisfaz a Racionalidade Individual se* $o_i \geq \min(\theta_i, 0)$ *para todo* i*.*

Demonstração. A demonstração é imediata. Considerando a função utilidade descrita pela Equação 4.3, o mecanismo deve satisfazer $u_i(\mathcal{M}(\boldsymbol{\theta}), \theta_i) \geq 0$ para todo i.

Para o caso em que $\theta_i \geq 0$: $o_i \geq 0$.

Para o caso em que $\theta_i < 0$: $o_i - \theta_i \geq 0 \therefore o_i \geq \theta_i$.

De forma compacta: $o_i \geq \min(\theta_i, 0)$. $\square$

Outro requisito é que a escolha do mecanismo maximize a soma das utilidades, isso ocorre se o mecanismo tiver Eficiência Estrita de Pareto.

Definição 4.3. *Um mecanismo* $(\mathcal{A}, \mathcal{M})$ *satisfaz a* ***Eficiência Estrita de Pareto*** *se, em equilíbrio, ele seleciona resultados* $\boldsymbol{o} \in \mathcal{O}$ *de forma que:*

$$\sum_{i \in \mathcal{N}} u_i(\boldsymbol{o}, \theta_i) \geq \sum_{i \in \mathcal{N}} u_i(\boldsymbol{o}', \theta_i), \forall \boldsymbol{o}' \in \mathcal{O}.$$

É importante, também, que a melhor estratégia dos usuários seja anunciar a verdade, de forma que não seja possível manipular o mecanismo.

Definição 4.4. *Um mecanismo* $(\mathcal{A}, \mathcal{M})$ *é* ***verdadeiro*** *se for direto e para todo* i *e* θ_i*, a estratégia no equilíbrio do usuário* i *é* $a_i = \theta_i$ *[10].*

4.2.2 Credibilidades dos Usuários

Para guiar o mecanismo, é definida uma credibilidade para cada usuário. A credibilidade de um usuário agrega as contribuições deste nas iterações anteriores. Isso é feito por meio de uma média móvel exponencial das alocações do usuário em cada instante. A credibilidade do usuário i no instante t, representada por $c_i(t)$, é definida como:

$$c_i(t) = (1 - \delta) \sum_{n=0}^{t} \delta^{t-n} o_i(n) \tag{4.4}$$

onde $\delta \in [0, 1)$ define o quanto as amostras passadas influenciam na credibilidade. Para $\delta = 0$, a credibilidade só considera a última alocação de recursos $(o_i(t))$. Quanto mais δ se aproxima de 1 mais os valores passados influenciam na credibilidade. Observe que, ao contrário do que pode ser pensado, quanto menor for a credibilidade, maior a quantidade de recursos contribuída pelo usuário, já que $o_i < 0$ indica que i fornece recursos. Assim, credibilidades negativas são melhores que as positivas.

Como o tempo é discreto, cada iteração representa um instante, e a credibilidade também pode ser definida por meio de uma equação de diferenças:

$$c_i(t) = (1 - \delta) \cdot o_i(t) + \delta \cdot c_i(t - 1), \tag{4.5}$$

com $c_i(0) = 0$. A representação por equação de diferenças é mais vantajosa para a implementação do sistema, já que o valor da credibilidade pode ser rapidamente calculado usando o valor da iteração anterior.

4.2.3 Conjunto de Alocações Desejáveis

Como dito anteriormente, as saídas do mecanismo devem pertencer ao conjunto de alocações possíveis ($\mathcal{O}$). No entanto, algumas alocações, mesmo que possíveis, não são desejadas. São então definidas algumas restrições para as alocações que podem ser geradas pelo mecanismo.

A primeira restrição é contra o desperdício de recursos. Não é permitido que mais recursos sejam contribuídos do que utilizados. Ou seja, a alocação deve obedecer $\sum_{i \in \mathcal{N}} o_i = 0$. Outra restrição imposta tem a finalidade de garantir a racionalidade individual. Para isso, basta impedir alocações que gerem utilidade negativa para qualquer usuário. Pelo Teorema 4.1, basta garantir que $o_i \geq \min(\theta_i, 0)$ para todo i. Para $a_i = \theta_i$, isso equivale a dizer que $o_i \geq \min(a_i, 0)$. Uma outra forma de desperdício de recursos é atribuir mais recursos a um usuário do que ele de fato precisa. Para impedir que isso ocorra, também com $a_i = \theta_i$, basta garantir que $o_i \leq \max(a_i, 0)$.

Uma representação das restrições acima de forma compacta é obtida com o subconjunto de alocações desejáveis $\Psi \subseteq \mathcal{O}$:

$$\Psi = \left\{ (o_1, \ldots, o_N) \in \mathcal{O} \middle| \sum_{i \in \mathcal{N}} o_i = 0 \wedge \forall i \in \mathcal{N}, \min(a_i, 0) \leq o_i \leq \max(a_i, 0) \right\}. \tag{4.6}$$

Com as novas restrições à saída do mecanismo, pode-se questionar se o mecanismo pode não ter solução possível para certas entradas $\boldsymbol{a} \in \mathcal{A}$. Portanto, é introduzido o teorema a seguir.

Teorema 4.2. *O conjunto $\Psi \subseteq \mathcal{O}$ é não vazio para todo conjunto $\mathcal{O}$ não vazio. Além disso, o conjunto Ψ contém pelo menos um elemento diferente de $(0, \ldots, 0)$ se, e somente se, existe pelo menos um par a_i, a_j de ações em $\boldsymbol{a} \in \mathcal{A}$ tal que $a_i \cdot a_j < 0$.*

Demonstração. Primeiro, demostra-se que $\mathcal{O} \neq \emptyset \implies \Psi \neq \emptyset$:

Se $\mathcal{O} \neq \emptyset$, pela Equação 4.1, $(0, \ldots, 0) \in \mathcal{O}$. Para que $o \in \Psi \subseteq \mathcal{O}$, $\min(a_i, 0) \leq o_i \leq \max(a_i, 0)$, $\forall i$. Como $\min(a_i, 0) \leq 0 \leq \max(a_i, 0)$ para todo $a_i \in \mathbb{Z}$, $(0, \ldots, 0) \in \Psi$. Logo $\Psi \neq \emptyset$.

Agora demonstra-se o resto da afirmação:

Se $a_i \cdot a_j < 0$, isso significa que um dos termos é positivo e o outro é negativo. Sem perda de generalidade, é possível supor que $a_i < 0$ e $a_j > 0$. Assim, pelas restrições de Ψ, $a_i \leq o_i \leq 0$ e $0 \leq o_j \leq a_j$. Supondo que todos os demais termos de o são nulos, para que $\sum_{n \in \mathcal{N}} o_n = 0$, $o_i = -o_j$. Logo, existe $\boldsymbol{o} \in \Psi$ tal que $\boldsymbol{o} \neq (0, \ldots, 0)$.

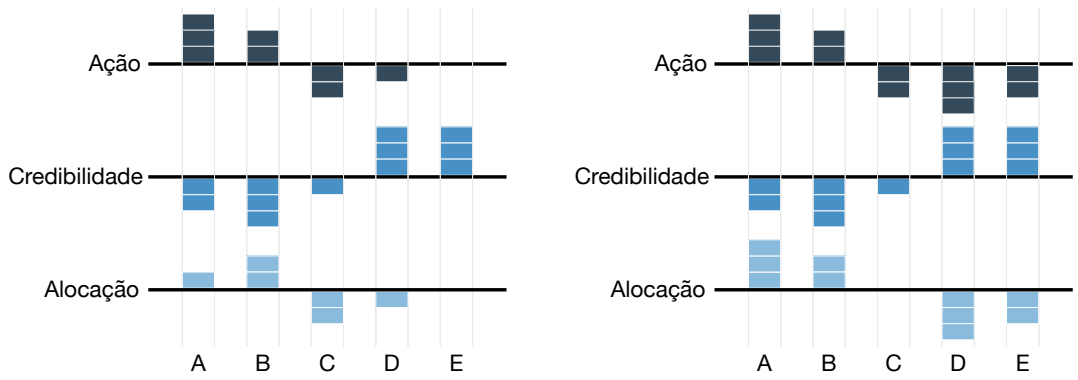
Para demonstrar a recíproca, por contradição, assuma que exista um vetor de ações $\boldsymbol{a} \in \mathcal{A}$ tal que $a_i \cdot a_j \geq 0$ para todo i e j, e um vetor de alocações $\boldsymbol{o} \in \Psi$ associado ao vetor $\boldsymbol{a}$, tal que $\boldsymbol{o} \neq (0, \ldots, 0)$. Como todos os elementos de $\boldsymbol{a}$ são do mesmo sinal ou nulos, isso implica que todos os elementos de $\boldsymbol{o}$ também são do mesmo sinal ou nulos. Assim, a única forma de garantir que $\sum_{n \in \mathcal{N}} o_n = 0$ é fazendo todas as saídas nulas, o que invalida o pressuposto de que $\boldsymbol{o} \neq (0, \ldots, 0)$.

□

Em outras palavras, o Teorema 4.2 implica que, se houver pelo menos um usuário disposto a fornecer recursos e pelo menos um outro disposto a consumir, pode ocorrer transferência de recursos respeitando as restrições de Ψ. O mecanismo deve então escolher uma alocação dentre as possíveis no conjunto de alocações desejadas.

4.2.4 Definição da Função Mecanismo

A intuição para a definição da função mecanismo está em usar a credibilidade do usuário para inferir a quantidade de recursos que este costuma contribuir ou requisitar. Os usuários que costumam contribuir mais devem receber mais. Por outro lado, usuários que costumam contribuir menos poderão compensar futuramente com maior contribuição.



(a) Demanda maior que oferta.

(b) Oferta maior que demanda.

Figura 4.2: Alocações geradas pelo mecanismo a partir das ações e credibilidades dos usuários A até E. Ações positivas representam demanda, enquanto negativas representam oferta. Credibilidades negativas indicam que o usuário contribuiu mais no passado.

Existem duas situações típicas de conflito de interesse com as quais o mecanismo deve lidar em relação à oferta e à demanda de recursos:

- A demanda de recursos é maior que a oferta, neste caso o conflito está entre os usuários que precisam de recursos;
- A oferta de recursos é maior que a demanda, neste caso o conflito está entre os usuários que podem fornecer recursos.

O mecanismo cuida dos dois casos de maneira simétrica. No caso do conflito entre os usuários que precisam de recursos, o mecanismo privilegia os usuários com melhor credibilidade. No conflito entre os usuários que fornecem recursos, a preferência é dada aos usuários com pior credibilidade. A razão para isso é que os usuários que contribuíram mais no passado devem ser preferencialmente recompensados com recursos, de forma a aumentar suas utilidades. Já os que contribuíram com menos recursos devem ter maior chance de contribuir e então melhorar suas credibilidades.

Considere um exemplo com cinco usuários (Figura 4.2a). Nesse exemplo, a demanda por recursos é maior que a oferta. Existe demanda por 5 recursos mas somente 3 são ofertados. O mecanismo cuida do conflito entre os usuários que precisam de recursos. Como o usuário B tem credibilidade melhor (mais negativa), ele contribuiu mais no passado e, portanto, recebe uma alocação de recursos maior.

O mesmo exemplo pode ser adaptado de forma que a oferta supere a demanda (Figura 4.2b). Observe que agora todas as demandas são aceitas e o usuário C deixará de contribuir com recursos, já que sua credibilidade é melhor que a de D e de E, os quais conseguem suprir a demanda sozinhos.

Essas alocações podem ser computadas matematicamente como será mostrado a seguir. Considerando o primeiro exemplo (Figura 4.2a), há um vetor de ações $\boldsymbol{a} = [3, 2, -2, -1, 0]$ e um vetor de credibilidade, calculado a partir de alocações passadas hipotéticas, $\boldsymbol{c} = [-2, -3, -1, 3, 3]$. O mecanismo usa a para definir o conjunto de alocações possíveis Ψ, conforme a Equação 4.6. Uma alocação $\boldsymbol{o} \in \Psi$ é então calculada de forma a alocar a quantidade máxima de recursos e minimizar a variância do vetor $\boldsymbol{c} + \boldsymbol{o}$. Em resumo, a função mecanismo resolve:

$$\underset{\boldsymbol{o} \in \Phi \subseteq \Psi}{\text{minimizar}} \sum_{i \in \mathcal{N}} (c_i + o_i)^2 \quad \textit{(minimizar a variância)} \tag{4.7}$$

tal que:

$$\Phi = \underset{\boldsymbol{o} \in \Psi}{\arg\max} \sum_{i \in \mathcal{N}} |o_i| \quad \textit{(alocar a quantidade máxima de recursos)} \tag{4.8}$$

Observe que a variância é escrita sem considerar a média do vetor $\boldsymbol{c} + \boldsymbol{o}$ já que as médias de ambos os vetores, $\boldsymbol{c}$ e $\boldsymbol{o}$, são nulas. Observe também que minimizar a variância de $\boldsymbol{c} + \boldsymbol{o}$, neste caso, possui o mesmo efeito que minimizar os máximos de $\boldsymbol{c} + \boldsymbol{o}$ lexicograficamente (minimizar o primeiro máximo, o segundo máximo e assim sucessivamente), quando a oferta é maior que a demanda, e maximizar os mínimos lexicograficamente, caso contrário. Embora o problema de minimização da variância seja mais simples de ser definido, o problema de otimização multiobjetivo pelo método lexicográfico representa melhor a abordagem do algoritmo a ser apresentado.

4.2.5 Algoritmo para o Problema da Função Mecanismo

Para resolver a Equação 4.7 é possível recorrer a métodos de otimização com restrições [26]. No entanto, também é possível definir um algoritmo determinístico. O método de otimização poderia ser aplicado diretamente a partir das restrições apresentadas até então. No entanto, os métodos de otimização podem ter convergência

lenta e muitas vezes não encontram a resposta correta. A seguir será apresentado o algoritmo determinístico.

Suponha que, ao invés de minimizar a variância, o objetivo seja maximizar o mínimo de $\boldsymbol{c} + \boldsymbol{o}$ enquanto as restrições permitirem. Depois disso, tentar maximizar o segundo mínimo também respeitando as restrições, e assim sucessivamente, para os mínimos seguintes. Esse tipo de otimização é o que caracteriza o método lexicográfico [26].

Devido à restrição imposta na definição do conjunto Φ (Equação 4.8), quando a demanda for maior que a oferta ($\sum_{i \in \mathcal{N}} a_i > 0$), as alocações de oferta serão definidas exatamente como as ações dos usuários (se $a_i \leq 0$ então $o_i = a_i$). Quando a oferta for maior que a demanda ($\sum_{i \in \mathcal{N}} a_i < 0$), as alocações de demanda é que serão definidas exatamente como as ações dos usuários (se $a_i \geq 0$ então $o_i = a_i$). Como, pela Equação 4.6, $\sum_{i \in \mathcal{N}} o_i = 0$, é necessário que $\sum_{i \in \mathcal{N}} \min(o_i, 0) = -\sum_{i \in \mathcal{N}} \max(o_i, 0)$.

Como os dois casos são análogos (oferta maior que demanda e demanda maior que oferta), é conveniente focar em apenas um deles. Quando $\sum_{i \in \mathcal{N}} a_i > 0$, resta ao algoritmo apenas definir a alocação da parte em conflito ($a_i > 0$). O algoritmo pode ser visto como uma alocação de recursos limitados entre os usuários que precisam ($a_i > 0$). A quantidade de recursos a ser alocada é dada pela soma das ações negativas ($-\sum_{i \in \mathcal{N}} \min(a_i, 0)$).

O algoritmo aloca todos esses recursos de forma que a variância de $\boldsymbol{c} + \boldsymbol{o}$ seja mínima, como dito anteriormente, isso é feito indiretamente. A Figura 4.3 na página seguinte mostra exemplos de alocação para diferentes quantidades de recursos. Os 4 gráficos da primeira linha mostram os valores de $\boldsymbol{c}$, $\boldsymbol{a}$, $\boldsymbol{o}$, $\boldsymbol{c} + \boldsymbol{o}$ para a alocação de 3 recursos. Os usuários estão dispostos em ordem de credibilidade. O Usuário 1 possui a melhor credibilidade enquanto o Usuário 9 possui a pior.

Observe que, como $\sum_{i \in \mathcal{N}} a_i = 7$, os usuários com $a_i < 0$ tem $o_i = a_i$. Assim, 3 recursos são alocados entre 5 usuários visando aumentar os mínimos de $\boldsymbol{c} + \boldsymbol{o}$ lexicograficamente. Como o primeiro mínimo de $\boldsymbol{c} + \boldsymbol{o}$ não pode ser alterado, devido às restrições de Φ, o algoritmo tenta aumentar o segundo mínimo. A diferença entre

Figura 4.3: 4 exemplos de alocação de recursos usando o mecanismo. A distribuição de credibilidade é a mesma para todos os exemplos, assim como as requisições por recursos. A cada exemplo, a quantidade de recursos sendo disponibilizada aumenta.

o segundo e o terceiro mínimo é de 2, o que coincide com a quantidade de recursos requeridas pelo Usuário 2, assim $o_2 = 2$. Ainda resta 1 recurso a ser alocado, este poderia ser alocado tanto para o Usuário 2 quanto para o Usuário 3, no entanto, como o Usuário 2 já adquiriu a quantidade de recursos que pediu, este recurso é alocado para 3, fazendo $o_3 = 1$.

As figuras seguintes mostram o que acontece quando mais recursos são disponibilizados pelos Usuários 1, 6 e agora também 9. Os usuários com credibilidades piores que 2 e 3 passam a ter suas ações atendidas. Com 5 recursos, 1 recurso a mais é fornecido ao Usuário 3 e 1 recurso é fornecido ao Usuário 4. Com 7 recursos, o Usuário 4 recebe mais 2 recursos. Finalmente, ao alocar 10 recursos, $\sum_{i \in \mathcal{N}} a_i = 0$, o que faz com que $\boldsymbol{o} = \boldsymbol{a}$.

O Algoritmo 1 na próxima página executa os mesmos passos descritos para os exemplos. É utilizada uma fila de prioridade Q, onde são inseridas as credibilidades dos usuários. Para a fila são definidas três operações: $\mathit{Extrair}_{Min}$, $\mathit{Encontrar}_{Min}$ e *Inserir*. $\mathit{Extrair}_{Min}$ retira e retorna o menor elemento da fila de prioridade $\mathit{Encontrar}_{Min}$ encontra o menor elemento da fila mas não o remove. *Inserir* insere um novo elemento na fila.

A cada iteração, o algoritmo calcula a diferença d entre a menor e a segunda menor credibilidades na fila. Para o usuário com a menor credibilidade é então calculada a alocação α como sendo o menor valor entre a quantidade de recursos disponíveis r, o número de recursos requisitados pelo usuário $\boldsymbol{a}[i]$ e a distância calculada d (linha 18). Caso o usuário ainda não tenha tido todas as suas necessidades atendidas, sua credibilidade descontada é reinserida na fila de prioridade (linha 24). O algoritmo termina quando todos os recursos foram alocados entre os usuários.

Para o caso em que a oferta é maior que a demanda, o algoritmo inverte o sinal das ações e das credibilidades dos usuários (linha 3) e também a alocação obtida no final do algoritmo (linha 26).

Observe que, embora as credibilidades dos exemplos sejam valores inteiros, isso não é necessariamente verdade e sim um artifício didático. As credibilidades são cal-

Algoritmo 1 Alocação de recursos entre os usuários.

Precondições: $\boldsymbol{a} \in \mathcal{A}$, $\boldsymbol{c} = [(1, c_1(t)), \ldots, (N, c_N(t))]$

$A \leftarrow \sum \boldsymbol{a}$

se $A < 0$ **então** ▷ Oferta maior que demanda.

  $\boldsymbol{a} \leftarrow -\boldsymbol{a} \quad \boldsymbol{c} \leftarrow -\boldsymbol{c}$

senão se $A = 0$ **então** ▷ Sem conflitos de oferta e demanda.

  retorne $\boldsymbol{a}$

para $i \leftarrow 1, N$ **faça**

  se $\boldsymbol{a}[i] > 0$ **então**

    Inserir$(Q, \boldsymbol{c}[i])$ ▷ Insere as credibilidades na fila de prioridade.

    $\boldsymbol{o}[i] \leftarrow 0$

  senão

    $\boldsymbol{o}[i] \leftarrow \boldsymbol{a}[i]$

$r \leftarrow -\sum_{i=1}^{N} \min(\boldsymbol{a}[i], 0)$

enquanto $r > 0$ **faça**

  $(i, u) \leftarrow \text{Extrair}_{\text{Min}}(Q)$

  $(j, v) \leftarrow \text{Encontrar}_{\text{Min}}(Q)$

  $d \leftarrow \lceil v - u \rceil$ ▷ Distância entre as credibilidades.

  se $d = 0$ **então** d $\leftarrow 1$

  $\alpha \leftarrow \min(r, \boldsymbol{a}[i], d)$

  $\boldsymbol{o}[i] \leftarrow \boldsymbol{o}[i] + \alpha$

  $r \leftarrow r - \alpha$

  se $\alpha < \boldsymbol{a}[i]$ **então** ▷ Usuário ainda precisa de mais recursos.

    $u \leftarrow u + \alpha$

    $\boldsymbol{a}[i] \leftarrow \boldsymbol{a}[i] - \alpha$

    Inserir$(Q, (i, u))$ ▷ Reinsere o usuário na fila.

se $A < 0$ **então**

  $\boldsymbol{o} \leftarrow -\boldsymbol{o}$ ▷ Re-invertendo para quando oferta maior que demanda.

retorne $\boldsymbol{o}$

culadas conforme a Equação 4.5, usando os valores de alocações passadas, podendo assumir valores reais.

4.2.6 Demonstração dos Requisitos

Os requisitos formais definidos na Seção 4.2.1 foram: racionalidade individual, eficiência estrita de Pareto e verdade. O requisito de racionalidade individual foi garantido diretamente pelas restrições de Ψ (Equação 4.6). Os demais são garantidos pelos teoremas que seguem.

Teorema 4.3. *Se o mecanismo $(\mathcal{A}, \mathcal{M})$ for verdadeiro e $\mathcal{M}$ satisfizer a restrição imposta pela Equação 4.8 então o mecanismo satisfaz a Eficiência Estrita de Pareto.*

Teorema 4.4. *O mecanismo $(\mathcal{A}, \mathcal{M})$ em que $\mathcal{M}$ satisfaz a Equação 4.8 é verdadeiro.*

Os Teoremas 4.3 e 4.4 serão demonstrados a seguir.

4.2.6.1 Eficiência Estrita de Pareto

Para a prova da eficiência estrita de Pareto é conveniente provar que as alocações negativas não geram utilidade negativa e que as alocações positivas são totalmente convertidas em utilidade para os usuários.

Lema 4.1. *Se em equilíbrio os usuários falam a verdade, o conjunto de alocações desejáveis Ψ faz com que $\forall i \in \mathcal{N}$ se $o_i \leq 0$ então $u_i = 0$ e se $o_i > 0$ então $u_i = o_i$.*

Demonstração. Repetindo a definição de Ψ:

$$\Psi = \left\{ (o_1, \ldots, o_N) \in \mathcal{O} \middle| \sum_{i \in \mathcal{N}} o_i = 0 \wedge \forall i \in \mathcal{N}, \min(a_i, 0) \leq o_i \leq \max(a_i, 0) \right\}$$

Se $o_i \leq 0$ significa que $a_i \leq o_i \leq 0$. Assim, como os usuários falam a verdade, $\theta_i \leq o_i \leq 0$. Pela Equação 4.3, $u_i = 0$.

Se $o_i > 0$ significa que $0 \leq o_i \leq a_i$. Assim, como os usuários falam a verdade, $0 \leq o_i \leq \theta_i$. Agora, pela Equação 4.3, $u_i = o_i$. □

De posse do Lema 4.1 prova-se o Teorema 4.3.

Teorema 4.3. *Se o mecanismo* $(\mathcal{A}, \mathcal{M})$ *for verdadeiro e* $\mathcal{M}$ *satisfizer a restrição imposta pela Equação 4.8 então o mecanismo satisfaz a Eficiência Estrita de Pareto.*

Demonstração. Por absurdo, vale supor que o mecanismo não satisfaz a Eficiência Estrita de Pareto. Assim, $\exists o' \in \mathcal{O}$ tal que, para alguma alocação do mecanismo $o \in \Phi$ vale:

$$\sum_{i \in \mathcal{N}} u_i(o', \theta_i) > \sum_{i \in \mathcal{N}} u_i(o, \theta_i)$$

Pelo fato de Ψ garantir que $\sum_{i \in \mathcal{N}} o_i = 0$, como o mecanismo também garante que $\sum_{i \in \mathcal{N}} |o_i|$ é máximo (Equação 4.8), isso equivale a maximizar a soma das alocações positivas $\sum_{i \in \mathcal{N}} \max(o_i, 0)$.

Como, pelo Lema 4.1, todas as alocações positivas em o resultarão em utilidades iguais às alocações. o' deve garantir que sua soma das alocações positivas seja maior que a de o. Para que a soma das alocações seja maior é necessário que mais recursos sejam alocados. No entanto, como $\sum_{i \in \mathcal{N}} \max(o_i, 0)$ é máxima dentro das restrições de Ψ, só é possível aumentar o número de alocações positivas usando mais recursos do que disponível ou fazendo que, para algum usuário i, $o_i > \theta_i$, o que não contribui para o aumento da soma das utilidades. Diante dessa impossibilidade pode-se dizer que o mecanismo satisfaz a Eficiência Estrita de Pareto. □

4.2.6.2 Mecanismo Verdadeiro

Finalmente demonstra-se que o mecanismo é verdadeiro. O mecanismo ser verdadeiro significa que a verdade é um equilíbrio de Nash em cada iteração. Embora a verdade seja um equilíbrio, é interessante notar que ela não constitui o único equilíbrio do jogo.

Teorema 4.4. *O mecanismo* $(\mathcal{A}, \mathcal{M})$ *em que* $\mathcal{M}$ *satisfaz a Equação 4.8 é verdadeiro.*

Demonstração. Como o mecanismo satisfaz a Racionalidade Individual, em qualquer momento as utilidades dos usuários serão positivas ou nulas. Pela definição da função utilidade (Equação 4.3), quando $\theta_i \leq 0$, $u_i \leq 0$ de modo que o usuário i não obterá melhor utilidade ao declarar $a_i \neq \theta_i$.

Para quando $\theta_i > 0$, chamando de a'_i uma ação mentirosa do usuário i de forma que $a'_i \neq \theta_i$. Quando $a'_i < \theta_i$ os limites impostos em Ψ passam a fazer com que $0 \leq o_i \leq a'_i < \theta_i$, limitando superiormente a quantidade de recursos atribuíveis ao usuário i e potencialmente reduzindo sua utilidade. Quando $a'_i > \theta_i$, os limites impostos por Ψ passam a ser $0 \leq o_i \leq a'_i$, o que faz com que o usuário i potencialmente receba mais recursos. No entanto, ao receber uma quantidade de recursos $o_i > \theta_i$, a utilidade de i permanece a mesma do que para quando $o_i = \theta_i$. Assim, para qualquer valor de θ_i para o usuário i, o usuário não consegue aumentar sua utilidade ao declarar uma ação $a'_i \neq \theta_i$. O que torna o mecanismo verdadeiro. $\square$

Capítulo 5

Avaliação da Proposta

5.1 Metodologia

Os dados utilizados para a análise representam 29 dias na nuvem do Google, com aproximadamente 12 500 máquinas e 930 usuários [27, 28]. Os dados foram extensivamente analisados por Reiss et al. [29]. Algumas das características encontradas foram heterogeneidade de tipos de recursos por máquina (quantidade de RAM, núcleos de CPU, etc.) e de uso (duração, recursos necessários, etc.). Essas características são também esperadas em um sistema colaborativo como o deste trabalho. Há diferença, contudo, na posse dos recursos: no Google, as máquinas são propriedades da empresa, a qual possui autoridade sobre a alocação; enquanto na proposta, cada usuário detém uma quantidade de recursos, hipoteticamente insuficiente para seus picos de uso. É necessário assim definir uma quantidade de recursos que cada usuário possui para que seja possível utilizar esses dados para a presente análise.

Os dados listam eventos de requisição oriundos de tarefas disparadas por usuários da nuvem. A Figura 5.1 na página seguinte mostra os estados possíveis para as tarefas, onde cada evento representa uma transição de estado. Em um funcionamento normal, uma tarefa é submetida pelo usuário, entrando no estado pendente. Após isso, ela é escalonada e entra em execução. Quando a execução termina, a tarefa vai para o estado morto. Depois de morta, a tarefa pode ser ressubmetida, entrando novamente no estado pendente.

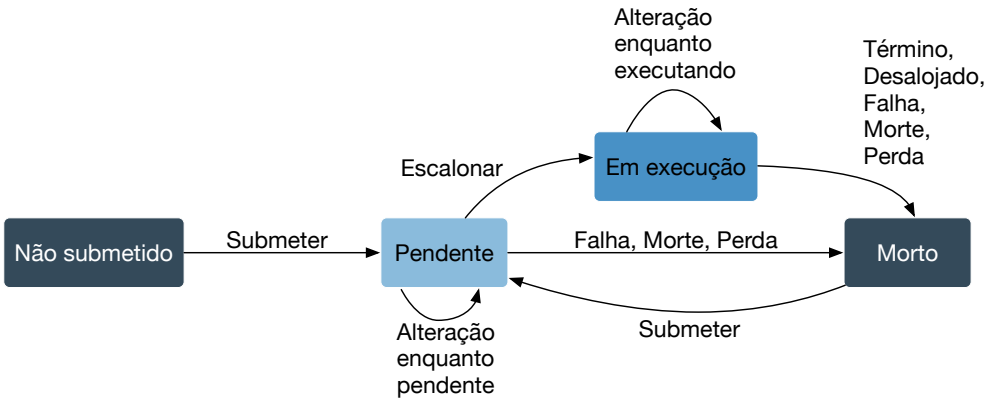


Figura 5.1: Cada evento representa uma transição de estado dos trabalhos e tarefas na nuvem do Google. Adaptado de [27].

Os eventos contêm o instante de ocorrência, com precisão de μs, além de informações sobre a tarefa a qual pertencem, dentre elas, um identificador do usuário dono e o valor normalizado dos requerimentos de CPU e memória. A normalização é feita usando como referência a maior capacidade do recurso disponível em uma máquina. Ou seja, se a maior capacidade de RAM em uma máquina da nuvem é 512 GB então 1,0 equivale a 512 GB. O mesmo vale para núcleos de CPU, onde a maior quantidade de núcleos de CPU, presentes em uma máquina da nuvem, equivale a 1,0.

Os eventos são usados para inferir as necessidades de cada usuário em cada iteração do mecanismo. Como são usadas informações de CPU e RAM, cada usuário terá duas necessidades diferentes. Eventos de *Submeter* fazem com que as necessidades do usuário correspondente sejam incrementadas. Já os eventos de *Término*, *Desalojado*, *Falha*, *Morte* e *Perda* fazem com que as necessidades sejam decrementadas. Os demais eventos não modificam as necessidades de recursos dos usuários.

As necessidades obtidas dos eventos podem variar em intervalos de μs, no entanto, o mecanismo deve ser executado em intervalos maiores, de forma que os recursos alocados possam ser aproveitados pelos usuários de alguma forma. Para as simulações, foi utilizado um intervalo de 10 min entre cada iteração do mecanismo. O valor ideal para o intervalo pode depender do comportamento dos usuários do mecanismo e do tempo de alocação de recursos no sistema.

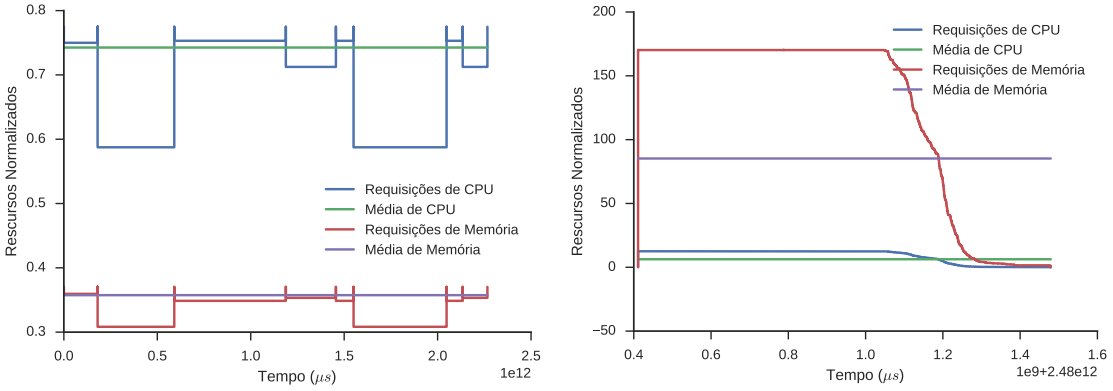


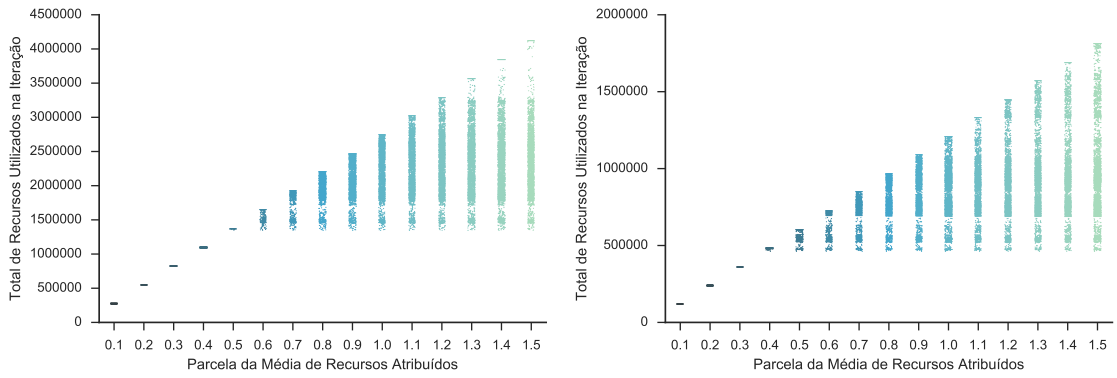
Figura 5.2: Necessidades de dois usuários arbitrários com comportamentos bem diferentes. Nos gráficos é possível ver os valores das necessidades de CPU e memória, bem como suas médias. As simulações consideram que os usuários possuem de 10% a 150% da média desses recursos.

Além disso, como dito anteriormente, a quantidade de recursos nas requisições foi normalizada em relação ao maior valor presente em uma máquina. Todavia, o mecanismo requer uma quantidade inteira de recursos. Para desnormalizar as requisições, estas foram divididas pelo menor valor normalizado encontrado e arredondadas para o menor inteiro maior ou igual (função teto). Assim, 1 passa a ser a menor quantidade de recursos que um usuário pode pedir ou fornecer.

Para definir a quantidade de recursos que cada usuário possui, usou-se como referência a média das necessidades de recursos do usuário ao longo do tempo. Todos os experimentos foram realizados considerando os usuários com quantidade de recursos variando de 10% a 150% da média de suas necessidades. A Figura 5.2 exemplifica o que isso significa para dois usuários do conjunto de dados. Com a quantidade de recursos que cada usuário possui especificada, é possível definir os tipos individuais em cada iteração do mecanismo como sendo a diferença entre a quantidade de recursos que cada usuário possui e suas respectivas necessidades.

5.2 Resultados

As análises a seguir buscam observar como o mecanismo se comporta quando os usuários possuem mais ou menos recursos. Além disso servem para determinar um



(a) Distribuição de CPU utilizada.

(b) Distribuição de memória utilizada.

Figura 5.3: Distribuição do total de recursos utilizados no sistema em todas as iterações. Observe que, para uma parcela de até 0,5 da média de CPU ou 0,4 da média de memória, o sistema está completamente sobrecarregado.

valor apropriado para δ (parâmetro de amortecimento da média móvel exponencial usado para definir a credibilidade na Equação 4.5).

A Figura 5.3 mostra a distribuição do total de recursos de CPU e memória utilizados pelos usuários em todas as iterações. Observe que, quando os usuários possuem menos recursos do que 50% da média de necessidade de CPU ou 40% da média de necessidade de memória, todos os recursos disponíveis são sempre utilizados. Já quando os usuários possuem mais recursos, o total de recursos utilizados varia, sempre limitado pela quantidade de recursos no sistema. A utilização total de recursos independe do parâmetro δ, já que esta independe da credibilidade (Equação 4.8). O que a credibilidade influencia é como esse número total de recursos será dividido entre os usuários.

A Figura 5.4 na próxima página permite observar a credibilidade dos usuários ao término de cada simulação. O gráfico em escala *symlog* (linear entre -1 e 1 e logarítmico para o resto) mostra como os valores de credibilidade podem variar bastante entre os usuários. Cada ponto do gráfico representa a credibilidade de um usuário ao fim das iterações, para diferentes valores de δ e de recursos atribuídos. Cada faixa representa credibilidades para o mesmo valor de δ mas variando a quantidade de recursos atribuídos a cada usuário, conforme a legenda.

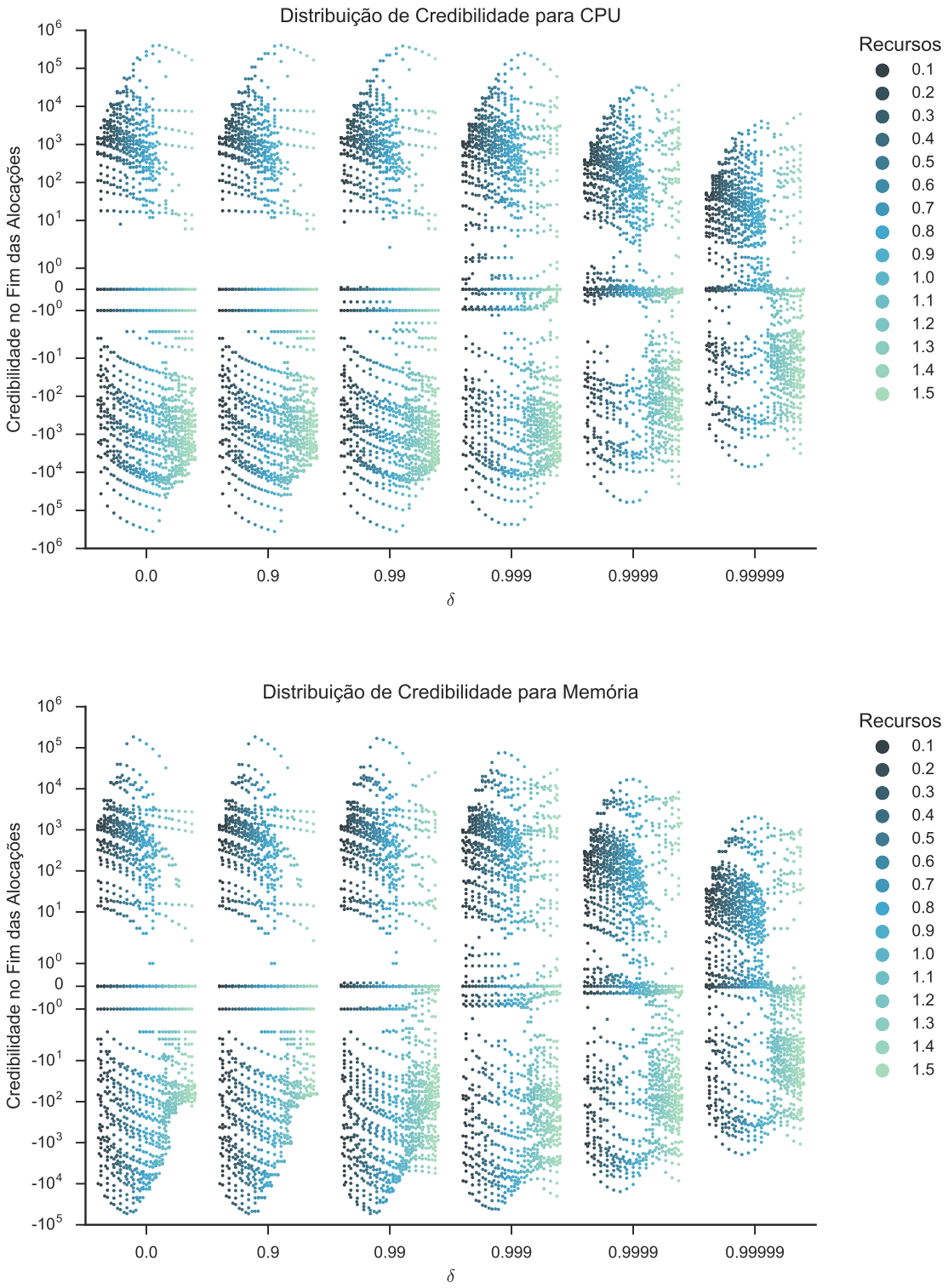
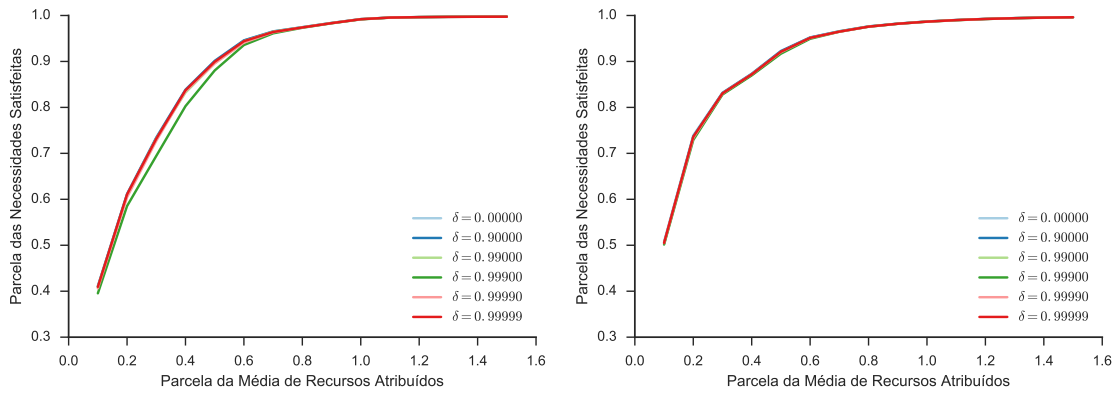


Figura 5.4: Distribuição das credibilidades de CPU e memória dos usuários no fim das alocações. Cada faixa representa um valor de δ diferente, com parcela de recursos atribuídos variando conforme a legenda.



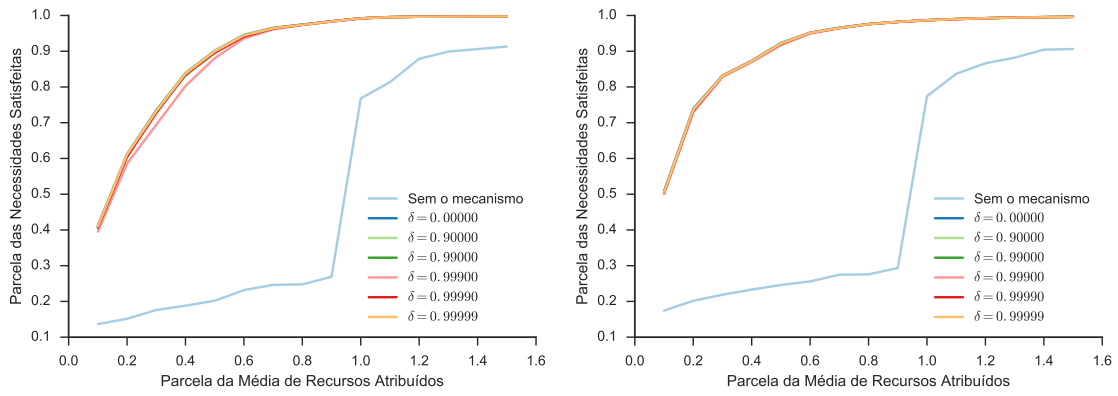
(a) Necessidades de CPU satisfeitas. (b) Necessidades de memória satisfeitas.

Figura 5.5: Parcela de necessidades por recursos atendidas para diferentes parcelas de recursos atribuídos. Não há diferença significativa de necessidades atendidas para os diferentes valores de δ.

O comportamento observado para as credibilidades de CPU e memória é parecido. A variação dos valores é praticamente igual para $\delta \in \{0, 0{,}9, 0{,}99\}$ considerando as mesmas quantidades de recursos atribuídos. Para valores maiores de δ, nota-se uma diminuição no intervalo de variação. Isso ocorre já que, quanto mais δ se aproxima de 1, mais as credibilidades demoram a convergir. Para exemplificar, considere um usuário que sempre contribui com 100 recursos. Para $\delta = 0{,}99999$, ao fim de 4100 iterações (aproximadamente o intervalo dos dados usados nas simulações) sua credibilidade seria apenas -4,02. Já para $\delta = 0{,}999$, sua credibilidade seria de $-98{,}35$.

Um outro parâmetro global de interesse é a parcela de requisições de recursos de todos os usuários que são satisfeitas ao usar o mecanismo. A Figura 5.5 mostra as parcelas das requisições de CPU e memória que foram atendidas pelo mecanismo. Isso considera apenas as requisições completamente satisfeitas, ou seja, se um usuário precisa de 5 recursos mas obtém 4 do mecanismo, sua utilidade foi positiva mas sua requisição não foi completamente satisfeita.

Como esperado, a parcela de requisições atendidas pelo mecanismo aumenta conforme a quantidade de recursos que os usuários possuem também aumenta. No entanto, mesmo para uma parcela de recursos de 0,1, o mecanismo consegue suprir 40% das requisições de CPU e 50% das requisições de memória. O efeito do



(a) Necessidades de CPU satisfeitas. (b) Necessidades de memória satisfeitas.

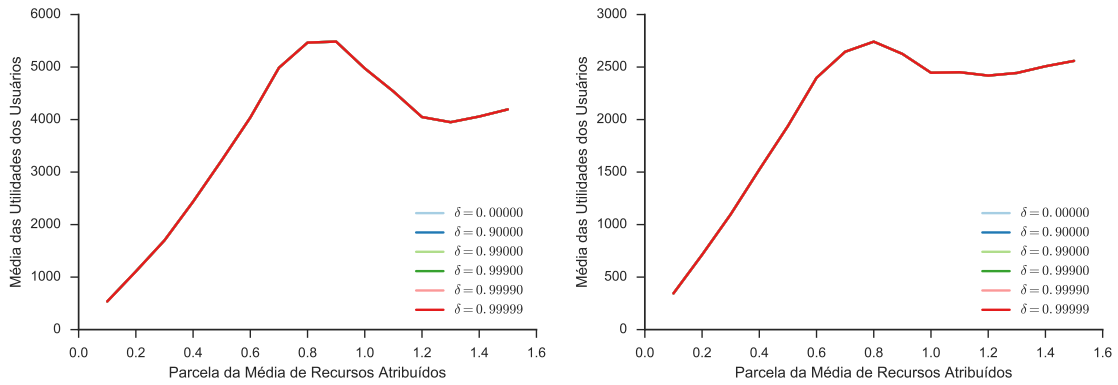
Figura 5.6: Comparação das necessidades atendidas ao usar ou não o mecanismo. A parcela de necessidades atendidas é consideravelmente maior ao usar o mecanismo, independentemente do δ utilizado.

parâmetro δ na parcela de requisições atendidas é mínimo. É também possível notar somente um pequeno desvio nas necessidades de CPU satisfeitas para $\delta = 0{,}999$.

Para observar o impacto do mecanismo na parcela de requisições satisfeitas, a Figura 5.6 inclui a parcela de requisições atendidas para quando os usuários não participam do sistema. É evidente a melhora obtida ao participar do mecanismo. Nota-se também que, sem o mecanismo, há um salto na parcela de necessidades satisfeitas para 1,0 da média de recursos atribuídos, para ambos os tipos de recursos. Isso possivelmente se deve ao fato de um número considerável de requisições dos usuários estarem próximas a média. Dessa forma, quando os usuários possuem mais recursos atribuídos do que a média, essas requisições passam a ser atendidas.

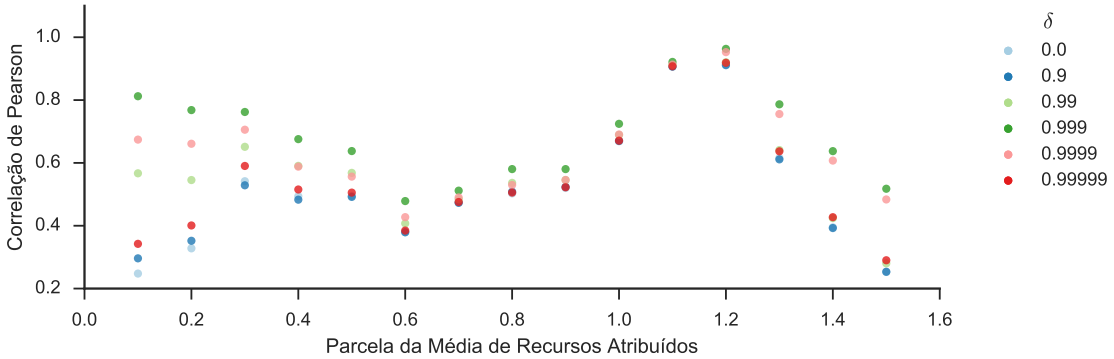
Uma grande vantagem em participar do mecanismo, mesmo quando os recursos atribuídos são 1,5 da média das necessidades, também pode ser observada. Porém, a maior vantagem global observada foi para 0,9 da média, caso em que houve um aumento de 27% para cerca de 98% das necessidades de CPU satisfeitas e de 29% para cerca de 98% das necessidades de memória satisfeitas.

Uma outra forma de medir o quanto o sistema está sendo útil aos usuários é obtendo a média das utilidades destes em todas as iterações. A Figura 5.7 na página seguinte mostra como a média das utilidades de todos os usuários em todas

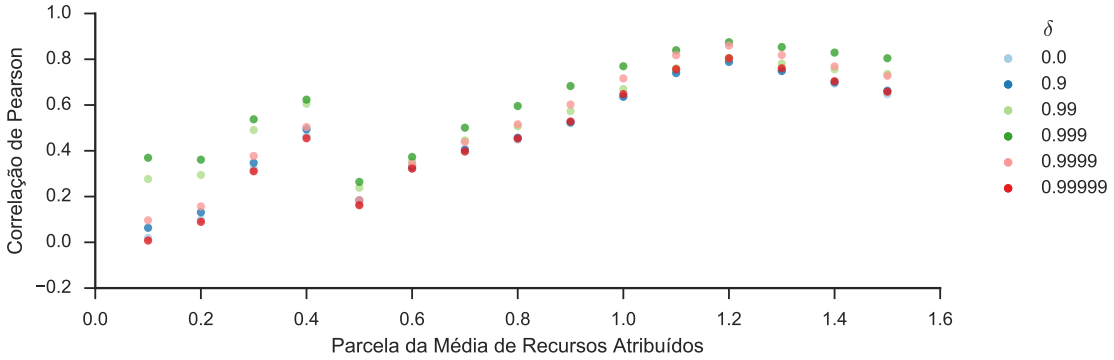


(a) Média das utilidades para CPU. (b) Média das utilidades para memória.

Figura 5.7: Média das utilidades de todos os usuários em todas as iterações para diferentes parcelas de recursos atribuídos. Pela forma como a utilidade é definida, a sua média no sistema independe do valor de δ.



(a) Correlação entre CPU contribuída e recebida.



(b) Correlação entre memória contribuída e recebida.

Figura 5.8: Correlação de Pearson entre os recursos contribuídos e recebidos no mecanismo. Observe como $\delta = 0{,}999$ resulta na melhor correlação para todas as parcelas de recursos atribuídos.

as iterações se comporta ao se variar a parcela de recursos atribuídos.

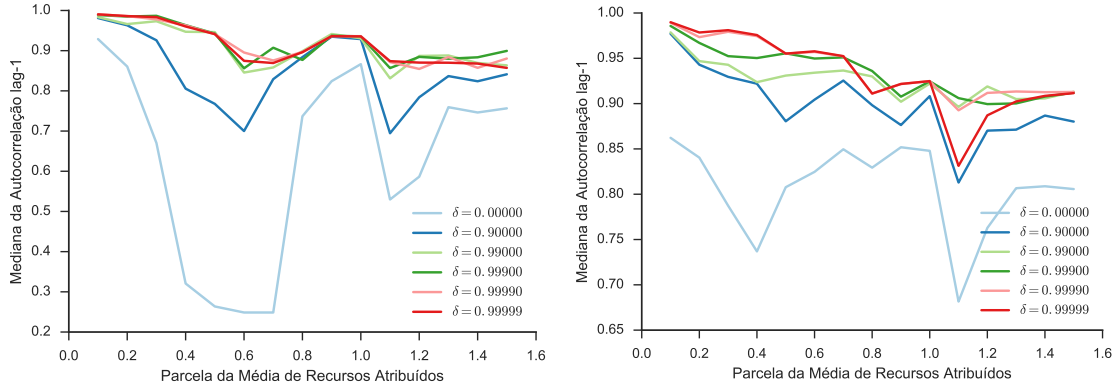
O fato de a utilidade ser menor para uma parcela de recursos maior se explica pela forma como a função utilidade foi definida, repetida aqui por conveniência:

$$u_i(o, \theta_i) = \begin{cases} \min(o_i, \theta_i), \text{ para } \theta_i \geq 0. \\ \min(0, o_i - \theta_i), \text{ para } \theta_i < 0. \end{cases} \tag{5.1}$$

A utilidade depende somente dos recursos que o usuário obtém do mecanismo e não da quantidade de recursos total que ele possui. Logo, isso não necessariamente significa que os usuários estão menos satisfeitos e sim que a utilidade que estes obtêm do mecanismo é menor. Isso acontece já que, com mais recursos, é possível suprir melhor suas próprias necessidades sem depender do mecanismo. Observe também que a média é idêntica para todos os valores de δ. Isso ocorre já que o total de recursos atribuídos em cada iteração do mecanismo é igual para qualquer valor de δ, fazendo com que a soma das utilidades também seja igual para qualquer valor de δ e consequentemente a média.

Os resultados apresentados até então analisaram o sistema de forma a constatar o desempenho global, mas sem observar o que cada usuário obtém do mecanismo. Isso também explica porque foi encontrada pouca influência do valor de δ. Como forma de avaliar se os usuários são recompensados por suas contribuições ao sistema, calculou-se o coeficiente de correlação de Pearson entre o total de recursos contribuídos e recebidos por cada usuário. O coeficiente de correlação de Pearson indica a dependência linear entre duas variáveis, quanto mais próxima de 1 for a correlação maior a probabilidade do usuário ter sido recompensado por contribuir recursos.

A Figura 5.8 na página precedente mostra o coeficiente de correlação de Pearson entre os recursos contribuídos e recebidos para as diferentes parcelas de recursos e valores de δ. A melhor correlação observada foi para $\delta = 0{,}999$, em todas as parcelas de recursos atribuídos e para ambos os tipos de recursos. Valores de δ muito próximos de 1 (como $\delta = 0{,}99999$) ou muito pequenos (como $\delta = 0$) apresentaram



(a) Estabilidade da alocação de CPU. (b) Estabilidade da alocação de Memória.

Figura 5.9: Mediana das Autocorrelações *lag-1* das alocações de CPU e memória de todos os usuários.

desempenho inferior. Esse comportamento provavelmente se deve ao fato de, para valores de δ pequenos, o sistema não guarda muitas das contribuições passadas. Já para valores de δ muito próximos de 1, o sistema demora muito a convergir, impedindo que as credibilidades dos usuários representem suas contribuições reais.

Uma outra medida de possível interesse para os usuários é o quão estáveis são as alocações. É interessante que, ao receber um recurso, este não seja retirado já na iteração seguinte. Para medir essa estabilidade de alocação, foi usada a autocorrelação *lag-1* ($\mathcal{R}_1$) da alocação de recursos, calculada para cada usuário em função das alocações $o_i(t)$ do usuário i em cada instante t:

$$\mathcal{R}_1(i) = \frac{\sum_{t=1}^{n-1}(o_i(t) - \bar{o}_i)(o_i(t+1) - \bar{o}_i)}{\sum_{t=1}^{n}(o_i(t) - \bar{o}_i)^2}, \tag{5.2}$$

onde n é o número total de iterações do mecanismo em cada simulação e $\bar{o}_i$ é a média das alocações do usuário i. Assim, quanto mais perto de 1 for $\mathcal{R}_1$, mais estáveis são as alocações.

A Figura 5.9 compara os valores das medianas de $\mathcal{R}_1$ de todos os usuários. $\delta = 0$ apresentou a alocação menos estável observada, principalmente para os recursos de CPU. Como, para $\delta = 0$, somente as alocações da última iteração são usadas como credibilidade, as decisões de alocação tendem a ser mais instáveis. As alocações de memória foram mais estáveis que as de CPU, possivelmente devido a requisições

também mais estáveis. Os valores de δ maiores que 0,9 resultaram nas alocações mais estáveis tanto para CPU como para memória.

Capítulo 6

Conclusões

Este trabalho propôs um mecanismo de alocação de recursos entre os usuários de uma nuvem colaborativa. O mecanismo proposto se baseia em teoria dos jogos tendo sido exploradas propriedades desejáveis, como Racionalidade Individual, a qual evita que os usuários queiram deixar o sistema, e Eficiência Estrita de Pareto, que garante o melhor uso dos recursos disponíveis. A alocação de recursos do mecanismo usou um algoritmo proposto também neste trabalho, o qual é usado para os dois casos análogos de conflito: demanda maior que oferta e oferta maior que demanda.

O desempenho do mecanismo foi testado com dados de requisições dos usuários da nuvem do Google. Foi possível observar os efeitos oriundos da agregação dos recursos e dos conflitos resolvidos pelo mecanismo. Nota-se a sobrecarga do sistema para quando os usuários contribuem uma parcela pequena de suas médias de recursos. A análise também mostrou os efeitos da variação do parâmetro de amortecimento de uso usado pelo mecanismo (δ). Para δ próximo de zero, o mecanismo não recompensa de forma apropriada os usuários que contribuíram com recursos. Para δ próximo de 1, o mecanismo demora a se adaptar ao comportamento real dos usuários. Os resultados mostraram que $\delta = 0{,}999$ é o melhor valor para o cenário avaliado. A parcela de requisições atendidas dos usuários também foi analisada. Neste quesito o mecanismo conseguiu desempenho melhor do que os usuários conseguiriam sozinhos, sem agregar seus recursos. No cenário em que os usuários possuem uma quantidade de recursos correspondente a 90% da média de suas necessidades, o mecanismo atendeu cerca de 98% das requisições, comparado a menos de 30% sem o uso do

mecanismo.

Como direção futura pretende-se avaliar parâmetros de justiça do mecanismo, possivelmente abrindo mão da Eficiência Estrito de Pareto. Da forma como o mecanismo foi proposto, as requisições de todos os usuários, independentemente da credibilidade são atendidas quando há recursos suficientes. Isso é essencial para que haja Eficiência Estrita de Pareto, caso houvesse maior controle sobre a justiça, é possível que usuários mal comportados fossem punidos, não recebendo recursos mesmo que disponíveis. Caso o mecanismo venha a ser estruturado dessa forma, deverá haver um compromisso entre eficiência e justiça. Outro tópico que deve ser analisado é o uso de múltiplos recursos de forma simultânea. No presente trabalho foi conduzida a análise para dois tipos de recurso tratados de forma separada, no entanto, quando há o compromisso com justiça, Ghodsi *et al.* [30] e Joe-Wong *et al.* [31] mostraram que alocações justas para recursos isolados podem deixar de ser justas quando vários tipos de recursos são combinados. Além disso, como forma de avaliar a atuação da proposta em cenários reais, o mecanismo deve ser integrado com plataformas de nuvem como o OpenStack[1] ou CloudStack[2].

Destaca-se também que, embora o mecanismo tenha sido idealizado para o cenário de nuvens colaborativas, ele é possivelmente aplicável a outros cenários. Um exemplo de aplicação alternativa é o compartilhamento energético em redes elétricas inteligentes [32]. Note contudo que o sistema exige que os cenários envolvam colaboração e compartilhamento de recursos, e que possa haver uma entidade centralizada para gerir as credibilidades. Ademais, o intervalo das iterações não pode comprometer a usabilidade dos recursos sendo compartilhados.

[1] `https://www.openstack.org/`

[2] `https://cloudstack.apache.org/`

Referências Bibliográficas

[1] COSTA, L. H. M. K., DE AMORIM, M. D., CAMPISTA, M. E. M., *et al.*, "Grandes Massas de Dados na Nuvem: Desafios e Técnicas para Inovação", *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*, p. 58, 2012.

[2] ARMBRUST, M., FOX, A., GRIFFITH, R., *et al.*, "A view of cloud computing", *Communications of the ACM*, v. 53, n. 4, pp. 50, 4 2010.

[3] SOTOMAYOR, B., MONTERO, R. S., LLORENTE, I. M., *et al.*, "Virtual Infrastructure Management in Private and Hybrid Clouds", *IEEE Internet Computing*, v. 13, n. 5, pp. 14–22, 9 2009.

[4] WEINS, K., "Cloud Computing Trends: 2016 State of the Cloud Survey", `http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey`, 2016, Acessado: 2016-09-19.

[5] VAQUERO, L. M., RODERO-MERINO, L., CACERES, J., *et al.*, "A break in the clouds", *ACM SIGCOMM Computer Communication Review*, v. 39, n. 1, pp. 50, 12 2008.

[6] COUTO, R. S., SCIAMMARELLA, T., BARRETO, H. F. S. S. M., *et al.*, "GT-PID: Uma Nuvem IaaS Universitária Geograficamente Distribuída", *Quinta Conferencia de Directores de Tecnología de Información - TICAL*, p. 19, 2015.

[7] MYERSON, R. B., *Game theory*. Harvard University Press, 2013.

[8] JACKSON, M. O., LEYTON-BROWN, K., SHOHAM, Y., "Game Theory", `https://www.coursera.org/learn/game-theory-1`, 2013.

[9] JACKSON, M. O., LEYTON-BROWN, K., SHOHAM, Y., "Game Theory II: Advanced Applications", `https://www.coursera.org/learn/game-theory-2`, 2013.

[10] SHOHAM, Y., LEYTON-BROWN, K., *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, 2008.

[11] AXELROD, R. M., *The evolution of cooperation*. Basic books, 2006.

[12] FRIEDMAN, J. W., "A Non-cooperative Equilibrium for Supergames", *The Review of Economic Studies*, v. 38, n. 1, pp. 1, jan 1971.

[13] JACKSON, M. O., "A crash course in implementation theory", *Social Choice and Welfare*, v. 18, n. 4, pp. 655–708, 10 2001.

[14] BURAGOHAIN, C., AGRAWAL, D., SURI, S., "A game theoretic framework for incentives in P2P systems", *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, v. 0121562, pp. 48–56, 2003.

[15] VELLOSO, P., LAUFER, R., O. CUNHA, D. D., *et al.*, "Trust management in mobile ad hoc networks using a scalable maturity-based model", *IEEE Transactions on Network and Service Management*, v. 7, n. 3, pp. 172–185, 9 2010.

[16] ANDRADE, N., CIRNE, W., BRASILEIRO, F., *et al.*, "OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing". In: *Job Scheduling Strategies for Parallel Processing*, v. 2862, pp. 61–86, 2003.

[17] CIRNE, W., BRASILEIRO, F., ANDRADE, N., *et al.*, "Labs of the World, Unite!!!", *Journal of Grid Computing*, v. 4, n. 3, pp. 225–246, 9 2006.

[18] DE LUCENA FALCAO, E., BRASILEIRO, F., BRITO, A., *et al.*, "Controlando a Contenção de Recursos para Promover Justiça em uma Federação Peer-to-Peer de Nuvens Privadas", *XIII Workshop em Clouds e Aplicações (WCGA2015)*, , 2015.

[19] MENASCHE, D. S., MASSOULIE, L., TOWSLEY, D., "Reciprocity and Barter in Peer-to-Peer Systems". In: *2010 Proceedings IEEE INFOCOM*, n. 1, pp. 1–9, 3 2010.

[20] LU, Z., WEN, X., SUN, Y., "A game theory based resource sharing scheme in cloud computing environment". In: *2012 World Congress on Information and Communication Technologies*, pp. 1097–1102, 10 2012.

[21] SAMAAN, N., "A Novel Economic Sharing Model in a Federation of Selfish Cloud Providers", *IEEE Transactions on Parallel and Distributed Systems*, v. 25, n. 1, pp. 12–21, 1 2014.

[22] NIYATO, D., VASILAKOS, A. V., KUN, Z., "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach", *Proceedings - 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2011*, pp. 215–224, 2011.

[23] SHOHAM, Y., "Conditional Utility, Utility Independence, and Utility Networks", *Proceedings of the Thirteenth Conference on Uncertaninty in Artificial Intelligence*, pp. 429–436, 1997.

[24] MYERSON, R. B., "Mechanism design". In: *Allocation, Information and Markets*, Springer, pp. 191–206, 1989.

[25] JACKSON, M. O., "Mechanism Theory", *Direct*, pp. 1–46, 2003.

[26] MARLER, R. T., ARORA, J. S., "Survey of multi-objective optimization methods for engineering", *Structural and Multidisciplinary Optimization*, v. 26, n. 6, pp. 369–395, 2004.

[27] REISS, C., WILKES, J., HELLERSTEIN, J. L., *Google cluster-usage traces: format + schema*, Technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2012.03.20. Posted at `http://code.google.com/p/googleclusterdata/wiki/TraceVersion2`.

[28] WILKES, J., "More Google cluster data", Google research blog, Nov. 2011, Posted at `http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html`.

[29] REISS, C., TUMANOV, A., GANGER, G. R., *et al.*, "Heterogeneity and dynamicity of clouds at scale : Google Trace Analysis", *Proceedings of the Third ACM Symposium on Cloud Computing - SoCC '12*, pp. 1–13, 2012.

[30] GHODSI, A., SEKAR, V., ZAHARIA, M., *et al.*, "Multi-resource fair queueing for packet processing", *ACM SIGCOMM Computer Communication Review*, v. 42, n. 4, pp. 1, 2012.

[31] JOE-WONG, C., SEN, S., LAN, T., *et al.*, "Multiresource Allocation: Fairness-Efficiency Tradeoffs in a Unifying Framework", *IEEE/ACM Transactions on Networking*, v. 21, n. 6, pp. 1785–1798, 12 2013.

[32] ZHU, T., HUANG, Z., SHARMA, A., *et al.*, "Sharing renewable energy in smart microgrids". In: *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, pp. 219–228, ACM, 2013.