# Distribuição de Tarefas em Sistemas de Workflow por meio da Seleção Induzida de Recursos

**Rogério Sousa e Silva**

Dissertação apresentada como requisito parcial

para obtenção do grau de Mestre pelo

Programa de Pós–graduação em Ciência da Computação da

Universidade Federal de Uberlândia

Programa de Pós–graduação em Ciência da Computação
Faculdade de Computação
Universidade Federal de Uberlândia
Minas Gerais – Brasil

Setembro de 2007

**Rogério Sousa e Silva**

# DISTRIBUIÇÃO DE TAREFAS EM SISTEMAS DE WORKFLOW POR MEIO DA SELEÇÃO INDUZIDA DE RECURSOS

Dissertação apresentada ao programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Autran Macêdo

**UBERLÂNDIA – MINAS GERAIS – BRASIL**
**Universidade Federal de Uberlândia - UFU**
**Setembro de 2007**

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU

S586d Silva, Rogério Sousa, 1971-
Distribuição de tarefas em sistemas de workflow por meio da seleção induzida de recursos / Rogério Sousa e Silva. – 2011.
83 f. : il.

Orientador: Autran Macedo.

Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós–Graduação em Ciência da Computação.
Inclui Bibliografia.

1. Computação – Teses. 2. Engenharia de software – Teses. I. Macêdo, Autran. II. Universidade Federal de Uberlândia. Programa de Pós–Graduação em Ciência da Computação. III. Título.

CDU:681.3

**Rogério Sousa e Silva**

# DISTRIBUIÇÃO DE TAREFAS EM SISTEMAS DE WORKFLOW POR MEIO DA SELEÇÃO INDUZIDA DE RECURSOS

Dissertação apresentada ao programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciência da Computação

Área de concentração: Engenharia de software.

Banca Examinadora:

Uberlândia, 12 de setembro de 2007.

---

Prof. Dr. Autran Macêdo – UFU (orientador)

---

Prof. Dr. Ilmério Reis da Silva - UFU (co-orientador)

---

Prof. Dr. Sergio Vale Aguiar Campos – UFMG

---

Prof. Dr. Stéphane Julia - UFU

*Dedicado à*

Às duas mulheres da minha vida.

Greyce e Isabela, amo vocês.

# Distribuição de Tarefas em Sistemas de Workflow por meio da Seleção Induzida de Recursos

**Rogério Sousa e Silva**

## Resumo

A entrega de tarefas para que sejam executadas pelos recursos de um sistema de *workflow* é chamada de distribuição de tarefas. A distribuição de tarefas é uma atividade importante para os sistemas de *workflow*, pois é necessário assegurar que uma determinada tarefa seja executada pelo recurso apropriado no tempo devido. Há várias abordagens para a distribuição de tarefas em sistemas de *workflow*. Este trabalho inova ao utilizar uma técnica oriunda da Análise de Ligações (*Link Analysis*) aplicada à distribuição de tarefas. A *Link Analysis* é utilizada para classificar o resultado de uma consulta na internet. A classificação é realizada considerando a relevância das páginas.

O presente trabalho propõe a aplicação da *Link Analysis* no contexto da distribuição de tarefas em sistemas de *workflow*. É proposto um novo algoritmo para a distribuição de tarefas (wf-hits) que é baseado no algoritmo de *Link Analysis*. O algoritmo wf-hits é comparado com trabalhos correlatos em termos quantitativos e qualitativos. Os experimentos realizados mostraram que a utilização do wf-hits na distribuição de tarefas aos recursos em sistemas de workflow representa ganhos na ordem de 25% em termos quantitativos mantendo os mesmos patamares de qualidade dos trabalhos relacionados.

Palavras-chave: *workflow*, análise de ligações, distribuição de tarefas, escalonamento de tarefas, *hits*.

# Tasks Distribution in Workflow Systems Based on Resources Induced Selection

**Rogério Sousa e Silva**

**Abstract**

The assingment of tasks to resources of a workflow system is called task distribution. The task distribution is an important activity for workflow systems, because it is necessary to ensure that a task is performed by the appropriate resource in due time. There are several approaches to task distribution in workflow systems. This work innovates by using a Link Analysis technique applied to the task distribution. The Link Analysis is used to rank the result of a web query. The rank is performed by considering the relevance of the pages.

This work presents the application of Link Analysis in the context of workflow task distribution. We have proposed a new task distribution algorithm (wf-hits) based on Link Analysis algorithm. We have compared wf-hits against other related ones. This comparison have considered quantitative and qualitative aspects. The experiments have shown that the use of wf-hits has improved workflow systems 25% in quantitative terms meanwhile the qualitative terms has maintained the same level of similar related works.

Keywords: workflow, link analysis, task distribution, task scheduling, hits.

# Agradecimentos

Muitas pessoas contribuíram de forma significativa para que este trabalho pudesse ser concluído. A todos os que de alguma forma me apoiaram deixo aqui meu sincero obrigado. Algumas dessas pessoas dedicaram um pouco de suas vidas para contribuir de alguma forma para que eu pudesse chegar até aqui. A estes quero deixar registrado minhas felicitações.

Greyce, sem você eu não conseguiria, você é o alicerce de nossa família. Sua força, garra e dedicação, seu apoio incondicional, os puxões de orelha, foram fundamentais para essa conquista. Só você para abdicar de tudo, prorrogar seus sonhos por mim. Sou eternamente grato a você. Graças a Deus que tenho você comigo.

Isabela, tudo isso é por você. Papai te ama muito.

Mãe e Pai, tudo o que disser será pouco para exprimir minha gratidão, vocês dedicaram a vida para eu chegar aqui.

Irmãos, o apoio de vocês, a confiança que sempre depositaram em mim alimentava minha garra e persistência. Fábio, Ricardo e Roberto amo vocês. Muito obrigado.

Vó, sempre que precisei de seu apoio a Senhora esteve pronta e fez de tudo por mim. Obrigado por tudo.

Loide, obrigado pela força nos momentos difíceis, seu total apoio ao nos receber de braços abertos jamais será esquecido.

Autran, muito mais que orientar este trabalho, seu apoio e compreensão foram importantes nesta etapa. Muito obrigado meu amigo.

Jean e Edileny, meus grandes amigos, as estadias e o apoio nos momentos difíceis jamais serão esquecidas. Obrigado a vocês pela dedicação, força e suporte que me deram.

Amigos do mestrado, Flávio, Renê, Rodolfo, Gustavo, Ivan, Alexandre, Grings, Ítalo, Joslaine, Heverton (Ton), Fernanda, Juliana, Vinícius, Márcio, Sérgio, valeu

galera, obrigado pelo apoio nas noites que passamos em claro, pela colaboração, pela diversão. Foi muito bom este tempo com vocês.

Aos companheiros do boliche, da sinuca, do futebol, do espetinho, da cantina e do “dina-bomba”, das pizzas baratas, das churrascarias em dia de promoção. Estas escapadas também foram fundamentais prá suportarmos a carga do mestrado.

Obrigado ao Programa de Pós–Graduação em Ciência da Computação da Universidade Federal de Uberlândia, aos professores e funcionários, pela oportunidade.

Obrigado a FESURV e UEG pelas concessões e aos Coordenadores/Diretores (Fabiana Girotto, Márcio Vilela, Paulo Francisco e Juliana) pelo apoio, pelos ajustes de horário, pelas liberações.

Obrigado Jesus, em todos os momentos que estive só, que fraquejei, que tive medo, que sentí saudades, que sentí dor, que dormí ao volante, que saí da estrada, o Senhor esteve ao meu lado me protegendo e dando forças prá continuar.

"*Minha vida é andar por esse país pra ver se um dia descanso feliz...*"

(Luíz Gonzaga)

# Sumário

# Lista de Abreviaturas

| Abreviatura | Significado |
|---|---|
| Abs_alloc | *Absolute allocation* |
| EDD | *Earliest due date* |
| FIFO | *First-in, first-out* |
| HITS | *Hypertext induced topic selection* |
| Sel-push | *Selective-push* |
| Sel-push-ft | *Selective-push-flowtime* |
| SGWf | Sistema de gerenciamento de workflow |
| SGWfs | Sistemas de gerenciamento de workflow |
| SIRO | *Service in random order* |
| SPT | *Shortest processing time* |
| SRI | *Sistemas de recuperação da informação* |
| SWTE | *Simple workflow test environment* |
| TI | *Tecnologia da informação* |
| UT | Unidades de tempo |
| VAB | Valor de alocação absoluta |
| XML | *Extensible markup language* |
| XPDL | *XML process definition language* |
| Wf-hits | *Workflow hipertext induced topic selection* |
| WfMC | *Workflow management coalition* |
| WHITS | *Weighted hypertext induced topic selection* |
| WPDL | *Workflow process definition language* |

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Distribuição de tarefas é uma atividade importante em sistemas de *workflow*. Essa importância está relacionada à necessidade de se assegurar que uma dada tarefa seja executada pelo recurso correto no tempo devido. A distribuição de tarefas tem influência direta no desempenho e na qualidade do serviço prestado por um sistema de *workflow* [23]. A maneira em que as tarefas são alocadas aos recursos é muito importante para a eficiência e eficácia do sistema como um todo [13].

A distribuição de tarefas é uma das atividades realizadas pelos sistemas de gerência de *workflow* (SGWf). Em geral, os SGWf adotam a política FIFO de distribuição associada a uma das estratégias abaixo [31]:

**Push** - o SGWf seleciona um recurso para executar uma dada tarefa;

**Pull** - o SGWf apresenta uma dada tarefa a um conjunto de recursos; um deles seleciona a tarefa para execução.

A estratégia *Push* é mais restritiva do que a *Pull*, mas há maior controle sobre execução da tarefa. Nos SGWf que adotam a estratégia *Push* os recursos são escolhidos segundo suas competências, assim espera-se que as tarefas sejam executadas com maior qualidade. SGWfs que adotam a estratégia *Pull* disponibilizam as tarefas para vários recursos. Eventualmente, um deles seleciona uma tarefa para execução. Os SGWf baseados em *Pull* alcançam menores tempos de execução das tarefas que os sistemas baseados em *Push*, pois por apresentar uma tarefa à vários recursos a tendência é que sejam diminuídas as filas de tarefas prontas para execução.

Segundo Aalst e Hee [31], o objetivo de um sistema de *workflow* é distribuir

as tarefas aos recursos o mais rápido possível. Neste contexto, alguns trabalhos [20, 29, 32], têm apresentado que a adoção de políticas de distribuição de tarefas mais sofisticadas do que *Pull* e *Push* simplesmente promovem maior eficiência no sistema de *workflow* como um todo. O presente trabalho propõe uma nova estratégia de distribuição de tarefas a partir das técnicas oriundas dos sistemas de recuperação de informações (SRI [5, 25, 26]).

Tarefas em SGWf são associadas aos recursos que são competentes para executá-las. Um recurso pode executar muitas tarefas ao passar do tempo, e as tarefas podem ser executadas por diferentes recursos. Esta relação entre recursos e tarefas tem características similares a ligação existente entre páginas da internet.

O relacionamento entre documentos (páginas) da internet ocorre a partir das ligações de entrada (*in-links*) e de saída (*out-links*) em uma coleção de documentos. A Análise de Ligações (*L*ink Analysis [2]) é uma técnica de classificação que explora a associação entre as páginas da internet. O algoritmo HITS (*H*ipertext Induced Topic Selecion [19]) que implementa a análise de ligações inspira este trabalho para distribuição de tarefas em ambientes de *workflow*. Até onde se estendeu esta pesquisa, trata-se de uma abordagem inédita.

A proposta deste trabalho inclui a definição do algoritmo *wf-hits*. O *wf-hits* utiliza os conceitos de *hub* e autoridade da Análise de Ligações para a classificação de recursos e tarefas. A classificação é realizada segundo o Valor de Alocação Absoluta (VAB) que é calculado a partir de parâmetros relacionados a recursos e tarefas. Quatro mecanismos para distribuição de tarefas foram desenvolvidos com base no algoritmo *wf-hits* (*Full-wf-hits*, *Random-wf-hits*, *Selective-wf-hits* e *Dynamic-wf-hits*) e o desempenho e qualidade na distribuição de tarefas realizadas por estes mecanismos foram analisados segundo os critérios estabelecidos nos trabalhos de Kumar *et, al.* [20] e Veloso [32]. O presente trabalho compara os resultados obtidos pelos mecanismos propostos e os confrontam com os resultados dos trabalhos de Kumar *et, al.* e Veloso.

Observou-se que os mecanismos propostos apresentaram ganhos de desempenho da ordem de 25% quando comparados aos trabalhos relacionados [20,32] melhorando sensivelmente os índices de qualidade no cenário de *workflow* simulado. Esses ganhos foram demonstrados através de gráficos comparativos de qualidade da distribuição e

de tempo de execução e também através de testes estatísticos (teste-t pareado [14]) que demonstraram a viabilidade desta proposta para aplicação em um sistema real de *workflow*.

## 1.1 Trabalhos relacionados

Duas áreas do conhecimento em computação formam a base para o desenvolvimento do presente trabalho: a distribuição de tarefas em *workflow* e a análise de ligações. A distribuição de tarefas é o objetivo deste trabalho. Todavia, para alcançar este fim, são utilizados os conceitos oriundos da área de análise de ligações. Desta forma, foram selecionados alguns trabalhos dessas duas áreas que fomentam o desenvolvimento deste. A seguir são apresentados esses trabalhos e suas relações com o proposto.

Tramontina [29] propõe uma técnica baseada em algoritmos genéticos (*Guess and Solve*) para tratar problemas de incerteza quanto (1) aos tempos de execução de cada atividade (tarefa) e (2) às rotas de execução de cada caso (processo). O foco desse trabalho é planejar uma possível sequência de tarefas antes do processo ser instanciado. O presente trabalho está focado na distribuição de tarefas de processos já instanciados.

Kumar, *et. al.* [20] visualizam a distribuição de tarefas conforme perspectivas de qualidade e desempenho, tentando criar um balanço entre elas. Nesse trabalho os autores relacionam a segurança com a qualidade das associações, considerando que as tarefas são atribuídas sempre aos recursos altamente capacitados. São apresentados mecanismos para estimar a aptidão e a taxa de ocupação dos recursos, a carga de trabalho do sistema e a qualidade de execução dos casos. Os mecanismos de distribuição propostos – baseados na estratégia *Pull* – distribuem as tarefas apresentando-as para os recursos mais qualificados de acordo com limiares de adequação. Esses mecanismos possibilitam uma melhor sintonia entre qualidade e desempenho.

O trabalho de Veloso [32] apresenta uma proposta de distribução de tarefas tendo como base a aptidão dos recursos. Neste contexto são apresentados três mecanismos baseados na estratégia *Push*. Os mecanismos de distribuição propostos apresentam melhor eficiência na distribuição de tarefas, com um balanceamento mais refinado

nos quesitos qualidade e desempenho. Esse trababalho apresentou resultados que demonstraram ganhos consideráveis relativos ao tempo de execução e à qualidade em comparação ao trabalho de Kumar, *et. al.* [20].

O trabalho de Kleinberg [19] apresenta o algoritmo(HITS - *Hipertext Induced Topic Selection*) para a classificação de páginas (documentos) da *web*. Este trabalho mostra que é possível classificar documentos a partir dos seus graus de autoridade. Isto é, no topo do *ranking* estarão os documentos com os maiores graus de autoridade. Por outro lado, também é mostrado que é possível classificar documentos considerando-se o grau de *hub*. Neste caso os documentos que estão no topo possuem ligações para os documentos com maior grau de *autoridade*. Melhorias foram adicionadas ao algoritmo HITS por Bharat e Henzinger [1], Chakrabarti [6] e Xing [38]. Esses trabalhos apresentam extensões ao trabalho de Kleinberg [18] e demonstram que a adição de pesos às ligações entre as páginas da internet provém melhorias significativas na precisão dos *rankings* gerados.

O presente trabalho utiliza os critérios de qualidade e desempenho de Kumar, *et. al.*, mas a abordagem baseada no HITS o torna completamente diferente. Além disso, o trabalho desses utilizam a estratégia *Pull* enquanto o presente trabalho utiliza a estratégia *Push*, com melhores resultados. Quanto ao trabalho de Veloso, o presente trabalho implementa uma abordagem diferente (HITS) cujos resultados são melhores do que esse. Um algoritmo baseado no HITS é apresentado neste trabalho. O algoritmo proposto gera *rankings* de recursos e de tarefas conforme seus valores de *hub* e de autoridade respectivamente. Além disso o algoritmo proposto adiciona pesos à relação entre recursos e tarefas, como nos trabalhos propostos por [1, 6, 38], porém, neste caso, os pesos são relacionados ao fator de alocação absoluta [20]. O fator de alocação absoluta é uma métrica que determina a alocação de uma tarefa a um recurso. A associação de pesos ao algoritmo proposto visa melhorar a precisão dos *rankings* gerados.

## 1.2 Organização do trabalho

Este trabalho está organizado da seguinte forma: o capítulo 2 discute os conceitos relacionados à análise de ligações e o algoritmo HITS. O capítulo 3 conceitua sistemas de *workflow* e apresenta a distribuição de tarefas em sistemas de *workflow*.

O capítulo 4 mostra o mapeamento entre os conceitos dessas duas áreas e propõe o algoritmo *wf-hits* e as estratégias baseadas nesse algoritmo. O capítulo 5 descreve o experimento computacional realizado neste estudo, o ambiente de experimentação, as estratégias adotadas e os resultados obtidos, discutindo suas implicações. Finalmente, o capítulo 6 conclui o trabalho analisando os impactos do uso desta técnica nos sistemas de *workflow* tendo em vista as melhorias de desempenho e de qualidade e apresenta os trabalhos que podem ser desenvolvidos a partir do assunto abordado nessa dissertação.

# Capítulo 2

# Análise de Ligações

A internet possui um grande acervo de documentos apresentados na forma de páginas interligadas. As ligações entre estas páginas (*hiperlinks*) proveêm uma valorosa fonte para a recuperação de informações [16]. Esta área da "Recuperação de Informações" é chamada Análise de Ligações (*link analysis*).

A Análise de Ligações tem sido aplicada para classificar documentos (páginas) da internet [3, 5, 26] conforme suas respectivas relevâncias. A relevância de uma página é calculada levando-se em conta as ligações de entrada (*in-links*) e de saída (*out-links*) de cada página. Uma página que é apontada por várias outras possui ligações de entrada. Uma página que aponta para várias outras possui ligações de saída. Esse relacionamento de apontamento (ligação) entre páginas *web* produz um esquema de propagação em dois níveis denominado **relacionamento mutuamente reforçado**. Isto é, se uma página é apontada por muitas outras ela é considerada uma **autoridade**, e se uma página aponta para muitas autoridades ela é denominada de ***hub***. A figura 2.1 ilustra esse relacionamento.

O relacionamento mutuamente reforçado da Análise de Ligações pode ser percebido em *workflow* com relação a tarefas e recursos. As tarefas "são apontadas" por recursos cujas competências os habilitam à execução das mesmas. Por sua vez, os recursos "apontam" para tarefas as quais eles têm competência para executar.

Alguns mecanismos de buscas conhecidos utilizam desta técnica para gerar um *ranking* de documentos relevantes à uma dada consulta, por exemplo, Altavista[1]

[1] © 2007 Overture Services, Inc. – http://www.altavista.com

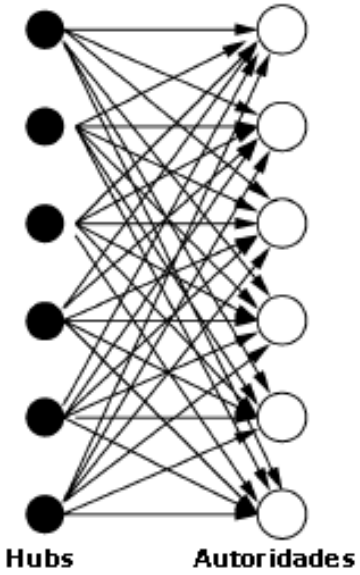



Figura 2.1: Ligações de entrada e de saída entre páginas da *web*.

[24] e Google[2] [17]. Este capítulo apresenta os conceitos relacionados a Análise de Ligações que serão referência para um melhor entendimento dos capítulos seguintes.

## 2.1 O Algoritmo HITS

Um algoritmo bastante conhecido para a análise de ligações é o *Hipertext Induced Topic Selection* – HITS, proposto por Kleinberg [19]. O HITS classifica as páginas da internet através de suas ligações em uma coleção de páginas [21]. Um enfoque simplificado é adotado pelo algoritmo HITS para mensurar a importância da página. Esta importância é aferida a partir do número de ligações de entrada (*in-links*) e de saída (*out-links*). Intuitivamente, se uma página $A$ que trata de um assunto $X$ é apontada por várias páginas, então estes apontamentos conferem a $A$ um grau de importância. Assim essas páginas consideram a página $A$ uma "autoridade" no que se refere ao assunto $X$. Por outro lado, se uma página aponta para "boas" autoridades, então pode se dizer que essa página é uma boa fonte de referência. Esta relação confere às páginas graus que pontuam o quanto uma página é autoridade em um assunto e também quão boa ela é como fonte de referência. Segundo Borodin, *et, al.* [2] estes graus são computados um em função do outro, ou seja,

[2] © 2007 Google – http://www.google.com

páginas consideradas "boas" autoridades são aquelas apontadas por "boas" fontes de referência, e "boas" fontes de referência são aquelas páginas que apontam para "boas" autoridades em um dado assunto.

Duas métricas foram propostas por Kleinberg [19] para mensurar a importância de uma página da internet: o grau de autoridade e o grau de *hub*. O grau de autoridade de uma página está relacionado com a quantidade de páginas que apontam para ela, ou seja, é o grau que mensura a autoridade da página em um dado assunto. O grau de *hub* está relacionado com a quantidade de páginas de qualidade que ela aponta, ou seja, o *hub* é o grau que indica o quão boa é a página como fonte de referência.

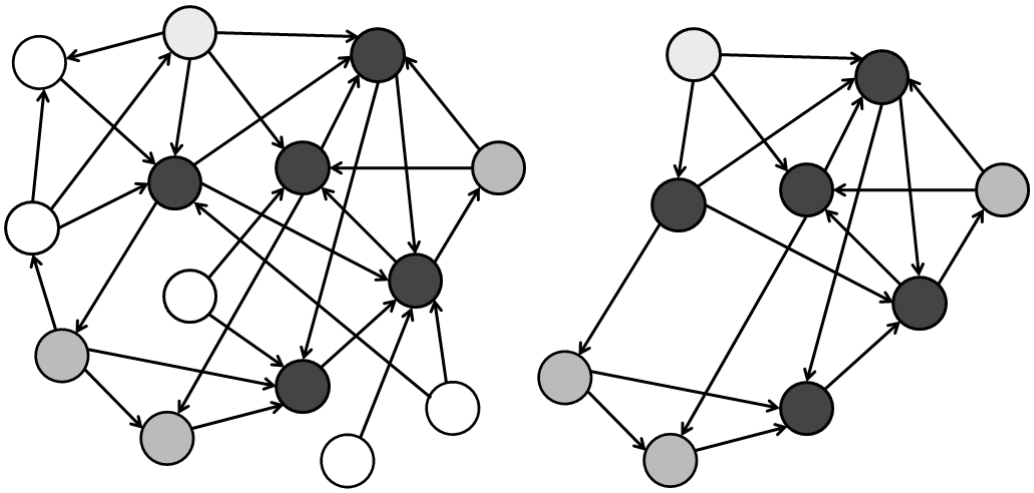
*Hubs* e autoridades apresentam então um relacionamento mutuamente reforçado, ou seja, bons *hubs* são páginas que apontam para boas autoridades, e boas autoridades são páginas que são apontadas por bons *hubs*. O algoritmo HITS calcula os graus de *hub* e de autoridade de cada página de uma coleção de páginas da internet. Estes cálculos levam em consideração uma análise da estrutura de ligações entre as páginas desta coleção [25]. Os graus são obtidos a partir de uma somatória dos pesos associados às ligações entre estas páginas. Este processo é repetido até que se alcance uma convergência para os graus de *hub* e autoridade. Em [19], o autor demonstra que o algoritmo converge e mostra experimentos onde a convergência é alcançada em aproximadamente 20 iterações.

O produto resultante da aplicação do algoritmo HITS a uma coleção de páginas da internet são dois *rankings*. No topo de um dos *rankings* estão as páginas com maiores graus de autoridade e no topo do outro *ranking* as páginas com maiores graus de *hub*. Se o critério adotado for autoridade, as páginas do topo são as mais relevantes para a consulta baseada no grau de autoridade desta no assunto da consulta. Caso o critério adotado seja o grau de *hub* da página, as páginas que aparecem no topo do *ranking* possuem ligações para aquelas com maiores graus de autoridade.

O algoritmo HITS pode ser aplicado a uma coleção de páginas da internet cujas ligações são representadas na forma de um grafo. A seção seguinte mostra a construção deste grafo.

## 2.2 Grafo de páginas

Podemos representar uma coleção de páginas da internet $S$ como um grafo direcionado $G = (N, AR)$ (figura 2.2(a)), onde $N$ é o conjunto dos vértices que correspondem às páginas, e $AR$ é o conjunto dos arcos que correspondem às ligações entre estas páginas. Se $p, q \in N$, então o arco direcionado $(p, q) \in AR$ indica a presença de uma ligação da página representada pelo vértice $p$ apontando para a página representada pelo vértice $q$. Os graus de entrada e de saída representam o número de arcos que chegam e saem de um vértice do grafo, ou seja, são respectivamente o número de ligações que apontam para um vértice $p$ (*in-link*) e o número de ligações que o vértice $p$ aponta (*out-links*).



(a) Grafo de páginas da internet ($G$) (b) Subgrafo de páginas da internet ($G_\sigma$)

Figura 2.2: Coleção de páginas relevantes da internet.

Um conjunto $S$ de páginas da internet pode ser obtido a partir de um mecanismo de busca tradicional, por exemplo, o modelo vetorial aplicado a uma consulta especificada pelo texto $\tau$, porém, dependendo da consulta, o modelo vetorial poderá retornar um conjunto muito grande de páginas (figura 2.2(a)). Para que se possa computar os graus de *hub* e autoridade destes documentos com um menor custo computacional, torna-se necessário focar uma subcoleção de páginas da internet $S_\sigma$ com as seguintes propriedades [19, 39]:

(i) $S_\sigma$ deverá ser relativamente pequena;

(ii) $S_\sigma$ deverá ser rica em páginas relevantes;

(iii) $S_\sigma$ deverá conter a maioria das autoridades da coleção $S$.

Desta forma, segundo Kleinberg [19], um pré-processamento da consulta deverá ser aplicado ao grafo $G$ com o intuito de construir uma coleção de páginas em concordância com as propriedades previamente estabelecidas. O algoritmo 1 gera o subgrafo $G_\sigma$ (figura 2.2(b)) que representa o subconjunto de páginas da coleção.

Sendo $S$ uma coleção de páginas da internet e $S_\sigma$ um subconjunto das páginas da coleção $S$, tal que, $S_\sigma \subseteq S$, podemos gerar um grafo $G_\sigma = (N_\sigma, AR_\sigma)$ para denotar o subgrafo que contém o subconjunto das páginas da coleção. Neste caso, $N_\sigma$ seria o conjunto de nós do subgrafo (páginas de $S_\sigma$) e $AR_\sigma$ o conjunto dos arcos envolvendo os nós de $N_\sigma$ (*links* de $S_\sigma$).

**Algoritmo 1** Algoritmo para seleção de páginas $S_\sigma$

```
1: Sejam: σ a consulta realizada
2:        ε um mecanismo de busca
3:        t, d números naturais
4: Inicie R_σ com os primeiros t resultados de σ em ε
5: Inicie S_σ com R_σ
6: ∀ página p ∈ R_σ, faça
7:        Seja Γ+(p) o conjunto de páginas que p aponta
8:        Seja Γ−(p) o conjunto de páginas que apontam para p
9:        Adicione todas páginas de Γ+(p) para S_σ
10:       se |Γ−(p)| < d , então
11:             Adicione todas páginas de Γ−(p) para S_σ
12:       senão
13:             Adicione um subconjunto de d páginas de Γ−(p) para S_σ
14:       fim se
15: Retorne S_σ
```

No algoritmo 1, inicialmente são atribuidas ao conjunto temporário $R_\sigma$ os primeiros $t$ resultados da consulta $\sigma$ no mecanismo de busca $\varepsilon$ (linha 4), os resultados do topo do *ranking* são os mais relevantes para a consulta. Em seguida (linha 5) a subcoleção $S_\sigma$ é iniciada com todas as páginas de $R_\sigma$. A cada iteração do laço o algoritmo seleciona uma página $p$ de $R_\sigma$ (linhas de 6 à 14) e adiciona todas as páginas que $p$ aponta em $S_\sigma$ (linha 9), se o número de páginas que apontam para $p$ for menor

que o valor $d$ pré-estabelecido (linha 10 e 11), então todas as páginas que apontam para $p$ são adicionadas à $S_\sigma$, caso contrário, um subconjunto contendo $d$ páginas são adicionadas à $S_\sigma$ (linha 13). O algoritmo encerra retornando o subconjunto $S_\sigma$ de páginas da coleção (linha 15).

A próxima seção mostra o cálculo dos graus de *hub* e de autoridade para as páginas do conjunto $S_\sigma$.

## 2.3 Computando *hubs* e autoridades

O cálculo dos graus de *hub* e autoridade é realizado a partir do grafo dirigido $G_\sigma = (N_\sigma, AR_\sigma)$. $G_\sigma$ é o grafo que representa um conjunto base de páginas e suas ligações, onde, $N_\sigma$ corresponde ao conjunto de vértices do grafo (páginas) e $AR_\sigma$ o conjunto de arcos (ligações entre as páginas). O algoritmo HITS computa os graus de *hub* e de autoridade a partir do grafo $G_\sigma$.

O algoritmo HITS utiliza o relacionamento mutuamente reforçado entre as páginas para manter e atualizar os graus de *hub* e autoridade. Para cada página, o grau de autoridade é obtido pela soma dos graus de *hub* das páginas que apontam para ela, e de maneira análoga, o grau de hub é obtido pela soma dos graus de autoridade das páginas que ela aponta. Ele opera sobre dois vetores: $H[n]$ que corresponde ao grau de *hub* da página $n \in N$; e $A[n]$ que corresponde ao grau de autoridade da página $n \in N$. O algoritmo 2 ilustra o algoritmo HITS proposto originalmente.

**Algoritmo 2** Algoritmo HITS

1: $x \leftarrow \mid N \mid$
2: $H[1..x] \leftarrow 1$
3: $A[1..x] \leftarrow 1$
4: **enquanto** $H$ e $A$ `não convergirem`
5: $\quad \forall n \in N$
6: $\quad\quad A[n] = \sum_{(m,n) \in AR} H[m]$
7: $\quad \forall n \in N$
8: $\quad\quad H[n] = \sum_{(n,m) \in AR} A[m]$
9: $\quad$ `Normalize` $A$ e $H$
10: **fim_enquanto**

No algoritmo HITS, inicialmente $x$ recebe o número de elementos de $N$ (linha 1) e cada posição dos vetores $H$ e $A$ é iniciada com 1 (linhas 2–3). Em seguida, em cada iteração do laço (linhas 4–8), o grau de autoridade (linha 5) e de *hub* (linha 6) são calculados. $A[n]$ é igual à soma dos graus de *hub* de todas as páginas que apontam para $n$. $H[n]$ é igual à soma dos graus de autoridade de todos as páginas para as quais $n$ aponta. Cada iteração termina com uma normalização, tal que $\sum_{i=1}^{x} H[i] = 1$ e $\sum_{i=1}^{x} A[i] = 1$. O laço termina quando $H$ e $A$ convergirem, isto é quando os valores desses vetores não tiverem mudança significativa entre a iteração corrente e a imediatamente anterior. Kleinberg [19] prova essa convergência e experimentalmente mostra que a mesma ocorre em torno da vigésima iteração.

## 2.4 Algoritmo HITS com pesos

O algoritmo HITS não qualifica as ligações de entrada ou saída. Todas as ligações possuem o mesmo valor para o cálculo dos graus de *hub* e autoridade. Os graus de *hub* e de autoridade são obtidos a partir da quantidade de ligações de entrada e saída. Assim, não são consideradas a relevância das páginas interligadas.

Algumas variações do algoritmo HITS tem sido desenvolvidas com o intuito de melhorar a precisão dos *rankings* gerados. Bharat e Henzinger [1] apresentaram um algoritmo no qual pesos são inseridos no cálculo dos graus de *hub* e autoridade. O algoritmo de Bharat e Henzinger realiza modificações no algoritmo HITS substituindo as equações das linhas 6 e 8 pelas equações 2.1 e 2.2.

$$A[n] = \sum_{(m,n)\in AR} H[m] \times aut\_wt(m,n) \tag{2.1}$$

$$H[n] = \sum_{(n,m)\in AR} A[m] \times hub\_wt(n,m) \tag{2.2}$$

Na equação 2.1, $aut\_wt(n,m)$ é o peso de autoridade da página $m$ que é obtido por $\frac{1}{k}$, onde $k$ é o número de páginas que apontam para $m$. Na equação 2.2, $hub\_wt(n,m)$ é o peso de *hub* da página $m$ que é obtido por $\frac{1}{l}$, onde $l$ é o número de páginas que $m$ aponta.

A aplicação de algoritmos mais eficientes para estimar os pesos das páginas de uma coleção melhoram a precisão dos *rankings* gerados [1, 21]. Li *et, al.* [21] demonstram em seus experimentos que a precisão do *ranking* pode ser melhorada consideravelmente pela adição de pesos ao cálculo dos graus de *hub* e autoridade. No trabalho de Li *et, al.* [21] o algoritmo HITS com pesos é comparado a versões com pesos obtidos a partir dos cálculos de similaridade baseados nos métodos: VSM (*Vector Space Model*) – Modelo Vetorial, Similaridade de Okapi e CDR (*Cover Density Ranking*) entre outros. Os resultados dessas comparações indicaram que o algoritmo HITS com pesos não apresentou diferenças significativas em relação aos demais baseados na análise de conteúdos.

# Capítulo 3

# Distribuição de tarefas em sistemas de *workflow*

Este capítulo trata da distribuição de tarefas em sistemas de *workflow*, porém, serão apresentados inicialmente alguns conceitos importantes relacionados com *workflow* que serão úteis para melhor compreensão do conteúdo.

## 3.1 Conceitos de *workflow*

Segundo a WfMC (*Workflow Management Coalition*) [36,37] – uma instituição não-governamental responsável pela normatização dos conceitos e terminologias relacionadas a *workflow* – ***workflow*** é automação total ou parcial de um processo de negócio na qual documentos, informações e tarefas são passadas de um participante para outro para que sejam executadas de acordo com um conjunto de regras e procedimentos.

A ativação, gerência e execução de um *workflow* é realizada por um **sistema de gerenciamento de *workflow*** (**SGWf**) . O SGWf é capaz de interpretar uma definição de processos, interagir com os participantes de *workflow* e quando necessário, invocar as aplicações e ferramentas de tecnologia da informação (TI), além de possibilitar a execução do processo de negócio.

Um **processo de negócio** é um conjunto de uma ou mais atividades relacionadas que coletivamente realizam um objetivo de negócio [37], geralmente dentro do contexto de uma organização, definido a partir de papéis funcionais e relaciona-

mentos [22]. A representação de um processo de negócio em uma forma que apoie a operação automatizada é uma **definição de processo**. A ferramenta de definição de processos permite definir o modelo funcional do processo de negócio (modelagem do processo de negócio).

A definição de processo é composta por um conjunto de atividades e suas relações além da indicação dos recursos aptos a executá-las.

**Recurso** é um nome genérico para uma pessoa ou máquina ou para um grupo de pessoas ou máquinas que podem realizar uma tarefa.

**Atividades** estão relacionadas com as unidades de trabalho de um processo de negócio. A cada instante a atividade recebe uma denominação diferente (tarefa, item de trabalho ou atividade). Uma **tarefa** é uma unidade lógica de trabalho. Ela requer, para que seja executada, um ou mais recursos (humanos/máquinas).

O recurso possui uma **lista de trabalho** (*worklist*), que é a relação das tarefas disponiveis para serem realizadas por ele. Em alguns casos a *worklist* pode ser compartilhada por um grupo de recursos. Em caso de compartilhamento, o recurso é responsável por escolher o item de trabalho que irá executar.

Ao ser disponibilizada na lista de trabalho (*worklist*) de um ou mais recursos a tarefa é chamada de **item de trabalho** (*workitem*), ou seja, o item de trabalho é uma tarefa que está disponível na *worklist* de um participante [22, 32]. O **participante** de *workflow* é um recurso que realiza um trabalho representado por um item de trabalho. A partir da seleção do item de trabalho pelo participante, este passa a ser considerado uma atividade, ou seja, a **atividade** é o item de trabalho em execução [32, 37].

A **execução do processo** é o tempo em que o processo torna-se operacional, ou seja, é o momento em que as instâncias do processo são criadas, executadas e gerenciadas. A **instância do processo** é um caso da definição de processo que foi ativada para a execução em um dado momento. Várias instancias do processo podem ser criadas a partir de uma definição de processo. O mecanismo responsável pela instanciação e execução de um processo é o motor *workflow* (*workflow engine*). O **motor *workflow*** é um serviço que provê o amibente de tempo de execução da instância de processo.

A figura 3.1 mostra os relacionamento entre os conceitos associados aos sistemas

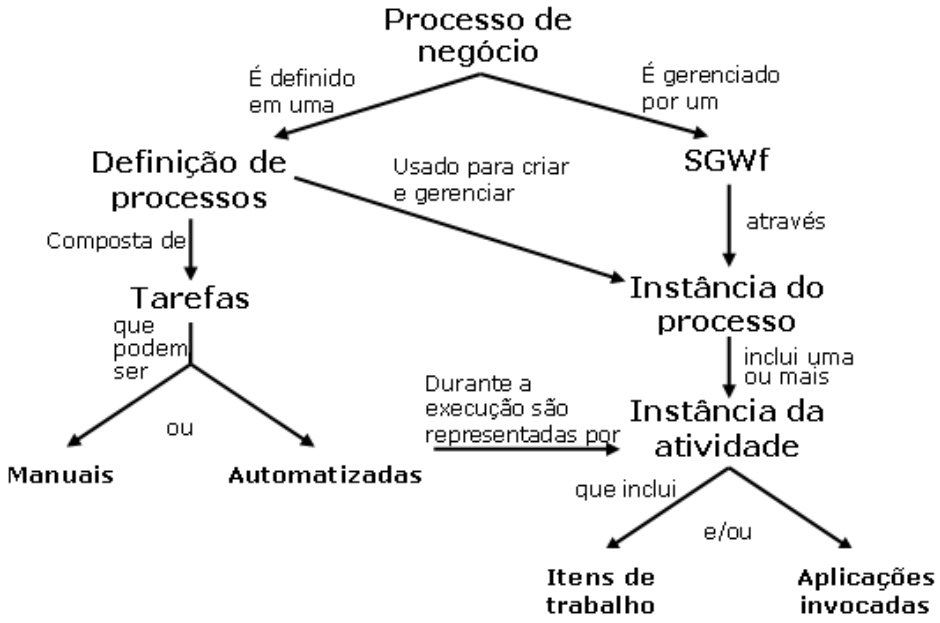



Figura 3.1: Relacionamento entre os conceitos de workflow.

de *workflow*.

## 3.2 Distribuição de tarefas

Um processo de negócio indica quais tarefas devem ser executadas, qual o recurso deverá executá-la e também a ordem de execução destas tarefas. Entretanto, a indicação de um recurso específico para execução de uma tarefa não é uma boa estratégia, pois desta forma o "*workflow*" fica totalmente dependente do modelo de recursos previamente definido.

A maneira que os itens de trabalho são alocados aos recursos é muito importante para a eficiência e eficácia do *workflow* [31]. Alocar as tarefas aos recursos mais competentes para executá-las pode garantir que as mesmas sejam executadas mais rapidamente e com melhores resultados. Além disso, evitar que recursos fiquem demasiadamente ocupados ou ociosos pode diminuir o tempo total de execução dos processos.

De modo a garantir que cada atividade seja executada pelo recurso apropriado as tarefas são modeladas na definição do processo com especificações dos papéis cujos recursos são aptos executá-las. Contudo, as associações entre tarefas e recursos somente são efetivadas durante a execução da instância de *workflow*. Isto acontece

devido às características dinâmicas dos recursos no procedimento de distribuição, ou seja, a tarefa deve ser associada à um recurso, porém, recursos podem estar "ocupados", recursos competentes podem executar a tarefa em intervalos de tempo distintos, e ainda, recursos podem cometer violações às restrições do processo de negócio. Estas características podem influenciar o tempo de execução dos casos dos processos de negócio.

## 3.3 Modificações dinâmicas

Uma característica desejável de um SGWf é a possibilidade de tratar as exceções que possam ocorrer durante a execução de uma instância de *workflow*, por exemplo, a indisponibilidade de uma impressora ou a ausência de um participante. Tal característica é importante para que os casos possam se adaptar às mudanças do ambiente do *workflow*. Estas mudanças são chamadas de modificações dinâmicas [23, 28].

Segundo Momotko e Subieta [23], as modificações dinâmicas são classificadas em dois tipos: modificações de instância e modificações de definição. As modificações de instância lidam com as alterações realizadas durante o tempo de execução dos casos de *workflow*, como por exemplo, a distribuição de tarefas à recursos. As modificações de definição ocorrem na especificação dos processos de negócio, por exemplo, uma reestruturação organizacional ou um aumento no número de tarefas do processo de negócio.

Segundo os trabalhos de Aalst [8, 9] os sistemas de gerenciamento de workflow têm problemas ao lidar com mudanças na definição dos processos em tempo de execução. Porém, o trabalho de Ellis, *et. al.* [12] apresenta um conjunto de definições matemáticas para representar as modificações dinâmicas em sistemas de *workflow* e apresenta uma solução para o problema. Momotko [23] apresenta uma proposta para alterações dos participantes dos procesos de *workflow* em tempo de execução. As modificações dinâmicas focadas na distribuição de tarefas constitui o objeto de estudo desse trabalho.

## 3.4 Princípios de distribuição

A associação de recursos às tarefas não é trivial [31]. Distribuir tarefas à recursos de forma desorganizada pode influenciar negativamente o tempo de conclusão das instâncias de *workflow*. Segundo Momotko e Subieta [23], o principal objetivo de um SGWf é garantir que uma dada tarefa será executada pelo recurso mais indicado no tempo devido. Desta forma, segundo Aalst e Hee [31], para se obter intervalos de tempo menores na execução dos processos de negócio, três decisões devem ser tomadas: (1) qual a ordem em que as tarefas serão transformadas em itens de trabalho? (2) qual o recurso deverá executar uma certa atividade? (3) qual a ordem em que os itens de trabalho serão transformados em atividades? Segundo Veloso [32], estas decisões influenciam diretamente na taxa de ocupação dos recursos, no aumento do tempo de execução e na qualidade da execução dos casos de *workflow*.

Segundo Tramontina e Wainer [30], diferentes heurísticas podem ser aplicadas para ordenar as tarefas. Elas foram propostas para solucionar problemas de escalonamento dos *job-shops* e são chamadas de *dispatching rules* [29,31]. Alguns conceitos relacionados com *workflow* e com problemas de escalonamento compartilham as mesmas idéias, o que possibilita um mapeamento entre os conceitos destas duas áreas. A escolha de uma tarefa para ser alocada a um recurso entre vários apresenta muitas similaridades com o roteamento de um produto através das máquinas em um departamento da produção [31]. Algumas dessas heurísticas são apresentadas a seguir:

- *First-In, First-Out (FIFO)*: as atividades são executadas na ordem em que foram instanciadas. Essa é uma heurística simples e robusta e é a mais utilizada na prática pelos sistemas de gerenciamento de *workflow*.

- *Shortest Processing Time (SPT)*: as atividades são executadas de acordo com o tempo de processamento, aquelas que possuirem os menores tempos de processamento serão selecionadas primeiro.

- *Earliest Due-Date (EDD)*: uma atividade é iniciada em um certo tempo (*date*) e deve preferencialmente ser terminada no tempo previsto(*due date*). Essa heurística determina a ordem de execução das atividades conforme sua *due*

*date*, ou seja, as tarefas que devem ser terminadas primeiro tem prioridade sobre as demais.

- *Service In Random Order (SIRO)*: nessa heurística, as atividades são selecionadas aleatóriamente. As atividades são apresentadas aos seus potenciais executores que escolhem quais serão executadas. Essa é uma política comum em SGWfs.

Segundo Veloso [32], o procedimento de alocação de tarefas é realizado em três etapas (figura 3.2): ordenação global, distribuição de tarefas e ordenação local. A ordenação global é realizada antes que o mecanismo de distribuição seja executado. Ela ocorre no momento que as tarefas estão prontas para se tornarem itens de trabalho. A distribuição de tarefas se encarrega de disponibilizar os itens de trabalho nas listas de trabalho dos recursos. A ordenação local, geralmente realizada pelos recursos, é o momento no qual os itens de trabalho são escolhidos para se tornarem atividades.

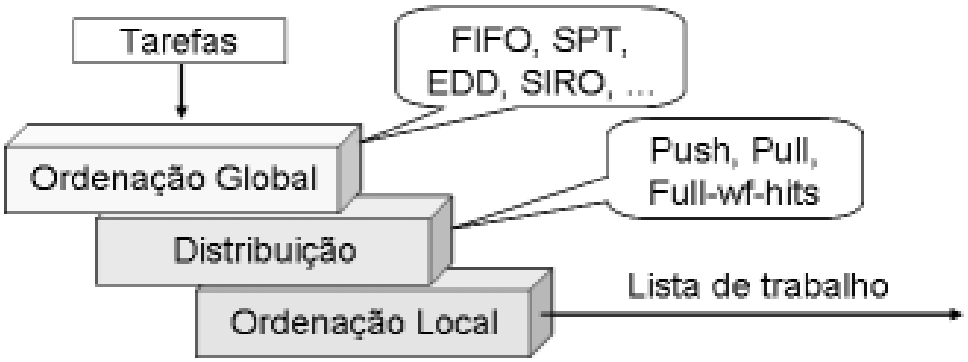



Figura 3.2: Etapas do processo de alocação de tarefas.

O SGWf é responsável por selecionar e distribuir as tarefas aos recursos. A seleção dos recursos está diretamente relacionada com a ordem que as tarefas ficam prontas para execução. A escolha dos recursos e a distribuição das tarefas deve acontecer de tal maneira que não ocorra sobrecarga aos recursos e não prejudique o tempo médio de execução das instâncias de *workflow*.

Alguns trabalhos [7,40] mostram que a atribuição de tarefas a recursos é realizada de acordo com as classes de papéis. Identificar qual recurso pertence a um determinado papel é considerado uma resolução de papel (*role resolution*). Na resolução de

papel o SGWf deve identificar os membros de um papel antes de notificá-los. Segundo Muehlen, essas decisões acerca das resoluções de papéis afetam diretamente a produtividade e a eficiência dos trabalhadores em uma organização [41]. Desse modo, faz-se necessário o desenvolvimento de mecanismos eficazes que administrem essas decisões.

## 3.5 Mecanismos de distribuição de tarefas

A distribuição de tarefas em sistemas de gerenciamento *workflow* em geral é realizada segundo uma política FIFO associada a uma das estratégias abaixo [20, 27, 29, 31]:

- *Push*: A associação entre itens de trabalho e recursos é realizada de forma automática. Dentro de certas condições, o SGWf escolhe o recurso que deverá realizar o item de trabalho. O recurso é incapaz de fazer escolhas. Assim que o recurso termina de executar uma atividade, outro item de trabalho é encaminhado a ele. Dessa forma, o motor atribui (*pushes*) itens de trabalho aos recursos (figura 3.3).

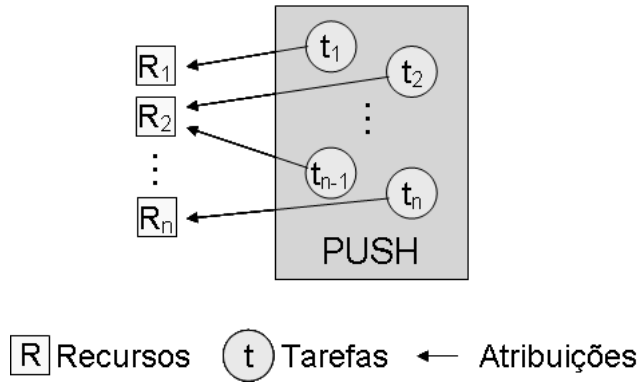



Figura 3.3: Mecanismo *Push*

- *Pull*: Os próprios recursos escolhem os itens de trabalho que querem executar. Cada recurso verifica o item que quer executar e o retira da fila compartilhada. Essa fila pode ser visível para todos os recursos ou somente para alguns, classificados em papéis. No caso da fila estar visível aos papéis, somente os recursos pertencentes a esses papéis podem acessá-la e retirar (*pull out*) itens (figura 3.4).

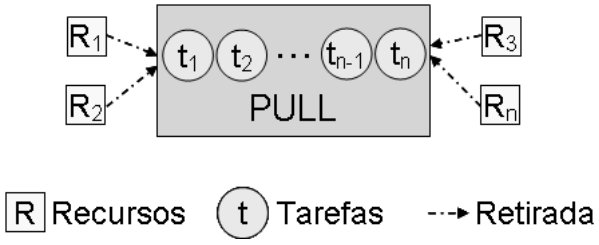



Figura 3.4: Mecanismo *Pull*

A estratégia *Push* é mais restritiva do que a *Pull*, mas há maior expectativa de que o produto final atenda seus requisitos de qualidade, pois os itens de trabalho são atribuídos somente aos recursos mais competentes para sua realização, assim, o produto final terá entrega dentro do prazo e com menos re-trabalho.

Na estratégia *Pull* a medida que o item de trabalho torna-se disponível o recurso tem o compromisso de retirá-lo da lista de trabalho. A retirada de itens pelos recursos torna a distribuição mais dinâmica e em consequência diminui o tempo de entrega do produto final.

Alguns trabalhos [20,29,32,33], no entanto, têm motivado a adoção de políticas de distribuição mais sofisticadas para promoverem maior eficiência no SGWf como um todo. O presente trabalho define extensões às estratégias de distribuição propostas por Kumar *et. al.* [20] e Veloso [32] no propósito de atingir um melhor equilíbrio entre qualidade (segurança) e tempo de execução dos casos (desempenho).

### 3.5.1 Qualidade e tempo de execução de processos

A "qualidade"da execução de uma instância de *workflow* está diretamente relacionada ao grau de aptidão dos recursos alocados para cada item de trabalho [20]. A métrica qualidade, segundo Kumar *et. al.*, indica o quão adequado foi o recurso para a execução do item de trabalho. Ao priorizar a entrega do item de trabalho ao recurso mais apto espera-se que este o realize com maior qualidade, e em consequência aumente a qualidade da execução da instância do *workflow*.

De acordo com o trabalho de Kumar et, al. [20], a aptidão de um recurso $R$ para execução de tarefa $T$ está diretamente relacionada com o grau de adequação de $R$ com relação a $T$. O grau de adequação está associado ao nível de competência dos recursos para a realização dos itens de trabalho do *workflow*. No contexto deste

trabalho, recursos com alto grau de adequação à uma determinada tarefa possuem maior aptidão para a executá-la. Por conseguinte, é de se esperar que esses recursos realizem essa tarefa de modo mais eficiente do que aqueles menos adequados. A eficiência está relacionada ao índice de re-trabalho e ao tempo de execução das tarefas. Quanto menores esses índices, maior a eficiência [32].

A aplicação de métricas de alocação e de qualidade promovem ganhos significativos nos quesitos de desempenho e de qualidade na distribuição de tarefas dos processos de *workflow* [20, 32]. Dessa forma, os SGWfs devem ser capazes de selecionar os recursos mais aptos para realizarem as tarefas de um processo qualquer que se deseja automatizar. A seção 3.6 apresenta os conceitos relacionados a aptidão dos recursos para a execução de tarefas e as métricas de alocação definidas por Kumar *et. al.* [20].

## 3.6 Aptidão dos recursos

Algumas características dos recursos e das tarefas são importantes para melhorar a segurança e desempenho dos SGWFs na distribuição de tarefas [20]. Apoiados nestas características, Kumar *et. al.* definiram algumas métricas de alocação. A seção seguinte apresenta estas métricas.

### 3.6.1 Métricas de alocação

Kumar *et. al.* definiram métricas de alocação para criar dinâmicamente um balanço entre qualidade e tempo de execução dos casos de *workflow*. Essas métricas são utilizadas para determinar os recursos mais adequados para a execução de tarefas. Para definição dessas métricas foram adotados alguns parâmetros oriundos das características dos recursos e das tarefas, são elas: Adequação, urgência, conformidade e disponibilidade.

#### Adequação

Adequação é a qualificação inerente de um recurso para executar uma determinada tarefa. As informações de adequação são armazenadas em uma tabela. A função $adequação(t, rs) \in [0, 1]$ retorna o grau de adequação do recurso $rs$ para a execução

da tarefa $t$. Quanto menor o valor de *adequação*, menos aptidão $rs$ possui para executar $t$. *Adequação*: $TXRS \rightarrow [0,1]$, onde $T$ é o conjunto de tarefas e $RS$ é o conjunto de recursos.

#### Urgência

Cada item de trabalho tem uma dependência de tempo para sua conclusão. Esse parâmetro é expresso pela função $urgência(t) \in [0,1]$. Quanto maior o valor, maior é a urgência de $t$. Intuitivamente, uma combinação de fatores afeta a urgência de uma tarefa. Esses fatores podem ser, por exemplo, a proximidade com a data de expiração da tarefa, a quantidade de dinheiro envolvido, o tamanho de uma compra ou uma necessidade imediata de um cliente. As tarefas terão níveis de urgência variados, e valores altos irão priorizar a sua alocação. *Urgência* é uma medida subjetiva determinada pelo gerente de *workflow*.

#### Conformidade

*Conformidade* define o quanto um recurso respeita as restrições definidas para o *workflow*. Para isso, utiliza-se de dois outros parâmetros: *penalidade* e *violação*.

Dado um conjunto de restrições $C$, para cada restrição $c \in C$ é atribuida uma penalidade ($penalidade(c) \in [0,1]$). Essa penalidade simboliza uma "multa" pela violação da restrição. As penalidades podem ser flexíveis ($penalidade(c) < 1$) ou inflexíveis ($penalidade(c) = 1$).

*Violação* ($violacao(c,t,rs) \in \{true, false\}$), indica se a restrição $c$ foi violada (ou não) pelo recurso $rs$ na execução da tarefa $t$. Assim sendo, a *conformidade* pode ser mensurada através da equação 3.1.

$$conformidade(t,rs) = \prod_{c \in C, violacao(c,t,rs)=true}(1 - penalidade(c)) \tag{3.1}$$

Como exemplo, considere que, em um conjunto de restrições $C$, duas delas $c_1$ e $c_2$ sejam flexíveis, com penalidades de 0,3 e 0,4 respectivamente e que estas sejam violadas. A conformidade neste caso é obtida a partir do cálculo apresentado em

3.2, ou seja, *conformidade* = 0,42.

$$\begin{aligned} conformidade(t, rs) &= (1 - penalidade(c_1)) \times (1 - penalidade(c_2)) \\ &= (1 - 0,3) \times (1 - 0,4) \\ &= 0,42 \end{aligned} \tag{3.2}$$

Similarmente, se ambas as restrições tiverem *penalidade* de 0,4, então a conformidade será de 0,36. O fator *conformidade* é outro parâmetro incluído na métrica de alocação e é uma medida do alcance de aceitação com as restrições. Um valor 0 de conformidade significa que uma restrição inflexível está sendo violada. Assim, valores de conformidade próximos de 1 (um) implicam menores violações, enquanto valores pequenos refletem maiores violações (ou baixa conformidade com as restrições).

#### Disponibilidade

*Disponibilidade* indica o quanto um recurso está disponível para a realização de uma tarefa. Esse parâmetro é expresso pela função $disponibilidade(rs) \in [0, 1]$. Valores abaixo de 1 indicam que o recurso está alocado a alguma tarefa. Valor igual a zero indica que o recurso está indisponível.

### 3.6.2 Fator de alocação absoluta

Os parâmetros apresentados anteriormente são combinados em uma métrica que determina a alocação de uma tarefa a um recurso. Esta métrica, chamada de *fator de alocação absoluta* ou, simplesmente, *absAlloc*, é definida como o produto das quatro métricas de aptidão.

$$\begin{aligned} absAlloc(t, rs) = \; & adequacao(t, rs) \times urgencia(t) \times \\ & conformidade(t, rs) \times disponibilidade(rs) \end{aligned} \tag{3.3}$$

O fator de alocação absoluta (equação 3.3) está associado a uma única tarefa e a somente um recurso. O *absAlloc* recebe valores que variam entre 0 e 1, e é uma medida absoluta de adequação de um recurso para execução de uma tarefa em um instante de tempo. Se o valor de *absAlloc* de um recurso é maior, então este recurso é o mais adequado (ou apropriado) para executar a tarefa no instante observado. É importante notar a diferença entre $adequacao(t, rs)$ e $absAlloc(t, rs)$. A primeira

é independente de contexto e não leva em consideração quaisquer restrições. A segunda depende da urgência de uma tarefa, de possíveis violações de restrições e ainda da disponibilidade do recurso.

Segundo Veloso [32], a métrica *absAlloc* pode ser adaptada caso o sistema de *workflow* não suporte alguns parâmetros. Por exemplo, se na modelagem dos processos não existirem restrições especificadas, basta considerar o parâmetro *conformidade* igual a 1. O valor 1 para conformidade significa que o recurso não violou quaisquer restrições, o que é verdadeiro, já que não existem ou não foram especificadas restrições na definição do processo. Da mesma forma, é também possível que todas as tarefas mantenham sempre o mesmo valor de urgência, como uma constante, assim todas as tarefas teriam a mesma prioridade.

**Exemplo 1**: Suponha a execução de uma instância de *workflow* qualquer no qual uma tarefa $A$ esteja pronta para ser executada e cuja urgência de alocação é 0,8. Considere também a inexistência de restrições. Para essa instância, três recursos estão disponíveis, a saber, $rs1$, $rs2$ e $rs3$. Os valores respectivos de adequação e disponibilidade estão expressos na tabela 3.1.

Tabela 3.1: Parâmetros de adequação para a tarefa $A$

| **Recursos** | **Disponibilidade** | **Adequação** |
|---|---|---|
| $rs1$ | 0,7 | 1,0 |
| $rs2$ | 1,0 | 0,8 |
| $rs3$ | 0,9 | 0,9 |

Utilizando a equação 3.3, o sistema calcula o valor de alocação absoluta para cada recurso disponível, identificando o recurso mais apropriado para executar a tarefa em questão. A tabela 3.2 apresenta os graus de alocação absoluta (*absAlloc*) dos recursos para a execução da tarefa $A$. Observando os valores de alocação absoluta, percebe-se que o recurso $rs3$ é o mais adequado para a execução da tarefa $A$.

### 3.6.3 Qualidade

A qualidade do trabalho realizado em sistemas *workflow* é uma métrica que informa se os recursos alocados para a instância de processo recém-executada foram os mais adequados ou não. Essa métrica é a razão da qualidade do trabalho realizado (quali-

Tabela 3.2: Valores de alocação absoluta para a tarefa $A$

| **Recursos** | $absAlloc$ |
|---|---|
| $rs1$ | 0,56 |
| $rs2$ | 0,64 |
| $rs3$ | 0,65 |

dade atual) e a máxima qualidade possível depois da atribuição de todas as tarefas que são parte da instância de *workflow*. Nesse sentido, definimos a qualidade atual como sendo:

$$Q_{atual}(w) = \sum_{t \in w} adequacao(t, exec(t)) \times conformidade(t, exec(t)) \tag{3.4}$$

A equação 3.4 avalia a qualidade atual da realização das tarefas $t$ em um *workflow* $w$. A função $exec$ mapeia uma tarefa em um recurso que atualmente executou $t$. Essa equação define a qualidade atual como o produto entre adequação e conformidade. Os valores resultantes dessa métrica estão entre 0 e 1. Os valores mais próximos a 1 indicam uma boa qualidade do trabalho. Qualidade com valor zero nunca irá ocorrer. Para ser zero, o recurso não é adequado ou não está em conformidade com as restrições, e se isso porventura acontecesse, o recurso não seria alocado para realizar a tarefa. Portanto, a qualidade nunca assumirá o valor zero.

A qualidade máxima do *workflow*, ou qualidade ideal, é definida como:

$$Q_{ideal}(w) = \sum_{t \in w} \max_{rs \in RS} [adequacao(t, rs) \times conformidade(t, rs)] \tag{3.5}$$

A métrica da equação 3.5 considera o valor máximo do produto entre adequação e conformidade de um recurso $rs \in RS$. São avaliados os recursos que participam do *workflow* $w$ e que podem realizar a tarefa $t$.

A qualidade atual e a ideal são utilizadas na definição da qualidade total. Essa qualidade total é a razão entre as qualidades atual e ideal:

$$Q_{total}(w) = \frac{Q_{atual}(w)}{Q_{ideal}(w)}$$

A qualidade total é tida como um valor no intervalo entre 0 e 1. Valores próximos a zero (nunca em zero) indicam baixa qualidade, ou seja, os recursos selecionados não

foram os melhores. Valores próximos a 1 mostram que os recursos mais indicados foram escolhidos e realizaram o trabalho.

**Exemplo 2**: Suponha que, no exemplo 1, a tarefa $A$ seja alocada para o recurso $rs3$. Logo, aplicando as equações 3.4 e 3.5, pode-se obter a qualidade resultante dessa alocação. Por exemplo:

$$Q_{atual}(w) = 0,9 \times 1,0 = 0,9$$

$$Q_{ideal}(w) = \max_{recursos}[(1,0 \times 1,0), (0,8 \times 1,0), (0,9 \times 1,0)] = 1,0$$

$$Q_{total}(w) = \frac{0,9}{1,0} = 0,9$$

Verifica-se, então, que a qualidade da alocação realizada possui qualidade igual a 0,9, ou seja, o grau de recursos adequados que executaram tarefas no sistema foi próximo ao ideal.

## 3.7 Mecanismos de distribuição baseados na aptidão dos recursos

A aptidão dos recursos é uma importante característica a ser considerada pelos mecanismos de distribuição, pois, com ela, é possível inferir a capacidade e a eficiência dos recursos para execução de tarefas específicas. Alguns autores [20, 32] definiram mecanismos para a distribuição de tarefas apoiados na aptidão dos recursos. Kumar *et al.* [20] apresentaram uma proposta de distribuição baseada no mecanismo *pull*. Veloso em [32] apresentou uma proposta baseada em *push*. As sessões subsequentes apresentam os mecanismos de distribuição propostos por Kumar *et al.* [20] e Veloso [32].

### 3.7.1 Mecanismos de distribuição baseados em *pull*

Mecanismos de distribuição baseados em *pull* apresentam as tarefas para os recursos que são competentes para executá-las. Em sua proposta Kumar *et al.* [20], apresentaram um algoritmo de distribuição de tarefas baseado em *pull* que considera a aptidão dos recursos para a distribuição das tarefas. Neste sentido, Kumar *et*

*al.* realizaram experimentos para mostrar a efetividade do *absAlloc* com diferentes níveis de exigência, ou seja, as tarefas são entregues somente aos recursos que alcançarem *absAlloc* maiores que os limiares preestabelecidos. Três limiares foram adotados nos experimentos de Kumar *et al.*, são eles: *Typical*, *Receptive* e *selective*:

- *Receptive*: uma tarefa $t$ é visível ao recurso $rs$ se, e somente se, $absAlloc(t, rs)$ exceder um valor de limiar menor que o típico, e.g., 0,4.
- *Typical*: uma tarefa $t$ é visível ao recurso $rs$ se, e somente se, $absAlloc(t, rs)$ exceder um valor de 0,5.
- *Selective*: uma tarefa $t$ é visível ao recurso $rs$ se, e somente se, $absAlloc(t, rs)$ exceder uma limiar mais alto que o típico, e.g., 0,6.

A figura 3.5 demonstra o funcionamento do mecanismo definido por Kumar *et. al.*.

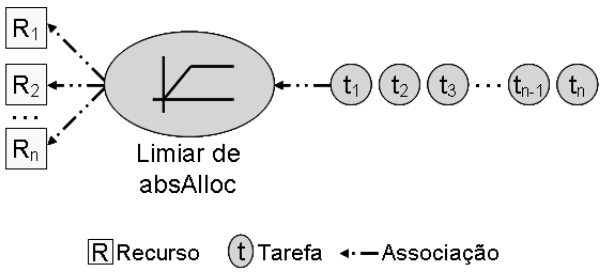



Figura 3.5: Mecanismos *Pull-driven* baseados em aptidão

### 3.7.2 Mecanismos de distribuição baseados em *push*

Veloso [32] apresenta em seu trabalho uma proposta de extensão aos mecanismos de distribuição definidos por Kumar *et. al.*. Embasando-se nos valores de alocação absoluta, essa extensão se diferencia das demais pela forma com que as tarefas são entregues aos recursos. Antes que os recursos recebam as tarefas, verificam-se os seus valores de disponibilidade. Ao receberem tarefas, os recursos ficam menos disponíveis no contexto do sistema. Assim, a medida que a carga de trabalho de um recurso aumenta, diminui a entrega de tarefas para ele. As tarefas que não são alocadas são delegadas para outros recursos adequados e que tenham disponibilidade para executá-las. Baseada em *push*, a extensão é implementada como o mecanismo

*Sel-push* e suas duas modificações, *Sel-push-10* e o *Sel-push-flowtime* [32]. Os mecanismos propostos por Veloso alcançaram um melhor balanceamento entre tempo de execução e qualidade na execução dos casos em comparação ao proposto por Kumar *et al.* [20].

### O Mecanismo *Sel-push*

O mecanismo *Sel-push* utiliza uma heurística em que cada recurso procura ocupar-se de tarefas enquanto estiver disponível. Veja na figura 3.6 o esquema de funcionamento do mecanismo *Sel-push*.

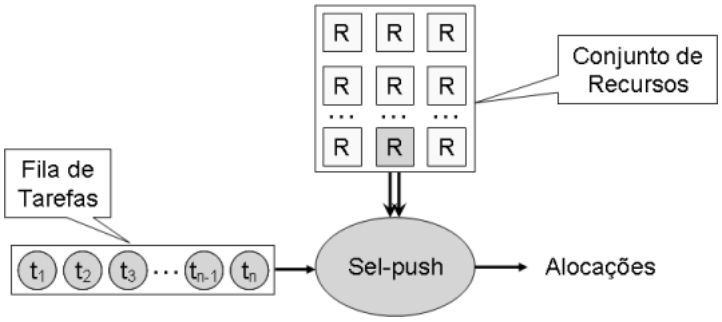



Figura 3.6: Esquema de funcionamento Sel-push

O mecanismo *Sel-push* realiza a seleção dos recursos segundo um limiar dinâmico de *absAlloc*, isto é, um dado recurso inicialmente encontra-se dentro de uma faixa de limiar seletivo ($< 0,6$) e, a medida que recebe tarefas, sofre decréscimos em seu valor de *absAlloc*, passando para o limiar típico ($> 0,5$) e em seguida para o receptivo ($> 0,4$). O *sel-push* verifica de antemão se os recursos mais aptos para a tarefa são seletivos, caso não existam, então procura-os em limiares mais baixos.

### O Mecanismo *Sel-push-10*

Semelhante ao *sel-push* em funcionamento, o mecanismo *sel-push-10* procura atribuir tarefas à recursos adequados. No entanto, além de selecionar o recurso com maior *absAlloc*, este mecanismo também considera a vizinhança do recurso. A vizinhança de um recurso são todos os outros recursos que possuem valores de alocação absoluta próximos em um raio de 10% do valor de *absAlloc* do recurso mais adequado. O recurso selecionado para execução da tarefa é o que tem maior disponibilidade entre os recursos da vizinhança. Veja figura 3.7.

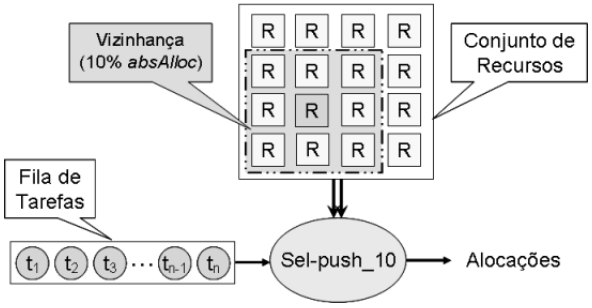



Figura 3.7: Esquema de funcionamento Sel-push-10

Segundo Veloso [32], o objetivo desta versão do *Sel-push* é diminuir uma possível sobrecarga do recurso mais adequado. Isso pode ocorrer quando, mesmo tendo recebido algumas tarefas, o recurso mais adequado ainda possua um valor de *absAlloc* maior que os demais.

### O Mecanismo *Sel-push-flowtime*

Outro mecanismo proposto por Veloso [32] também baseado no *Sel-push*, é o mecanismo denominado *Sel-push-flowtime*. Esse mecanismo agrega ao *Sel-push* a característica de considerar, além da disponibilidade e a aptidão dos recursos, o *flowtime* das tarefas.

Define-se o *flowtime* de uma tarefa, como o tempo entre o seu início e término em uma instância de workflow [4]. Assim, o objetivo é realizar as entregas das novas tarefas de acordo com os *flowtimes* daquelas já alocadas. Quando o mecanismo detecta uma tarefa pronta para ser executada, procura pelo recurso com maior *absAlloc*. No entanto, o recurso escolhido será aquele com menor *flowtime* para a sua última alocação. Este mecanismo também adota o limiar dinâmico de *absAlloc* para a alocação.

O esquema desse algoritmo é mostrado na figura 3.8, onde cada recurso possui um atributo que mede o *flowtime* de cada tarefa (e.g., 5ut ou 5 unidades de tempo).

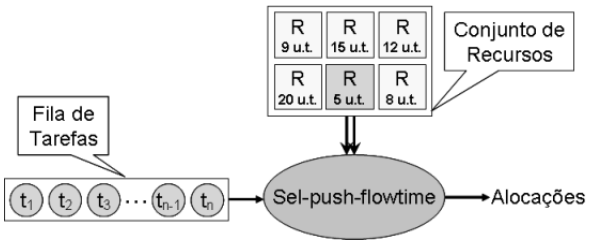



Figura 3.8: Esquema de funcionamento Sel-push-flowtime

# Capítulo 4

# Aplicação do HITS em *workflow*

A associação entre recursos e tarefas em *workflow* podem ser estudadas como ligações entre páginas da web, desde que sejam feitas algumas adaptações. As ligações (*in-links* e *out-links*) na internet ocorrem entre objetos do mesmo tipo: documentos (páginas) da internet. As ligações em *workflow* ocorrem entre objetos de tipos diferentes: recursos e tarefas. Assim sendo, para aplicação do HITS em *workflow* foi adotado o seguinte critério para as ligações: as tarefas possuem *in-links* e os recursos possuem *out-links*. Deste modo, os recursos apontam para as tarefas às quais esses possuem competência para executar e em consequencia as tarefas são apontadas pelos recursos competentes para executá-las.

O HITS associa um grau de *hub* a uma página $H$ de acordo com a quantidade de páginas apontadas por $H$ e da qualidade dessas páginas, e associa um grau de autoridade a uma página $A$ de acordo com a quantidade de páginas que apontam para $A$ e o grau de importancia dessas páginas. Analogamente em *workflow*, um grau de *hub* será atribuído a um recurso $R$ de acordo com a quantidade e o grau de autoridade das tarefas para as quais $R$ aponta (i.e. possui competência para executar). Já o grau de autoridade da tarefa $T$ será calculado por meio da quantidade e do grau de *hub* dos recursos que apontam para a tarefa $T$ (i.e. tem competência para executá-la). A associação entre recursos e tarefas é apresentada na figura 4.1.

O resultado obtido pelo algoritmo HITS é apresentado através de dois *rankings*, um classificado pelos graus de autoridade das páginas da coleção e outro classificado pelos graus de *hub* destas páginas. As páginas do topo destes *rankings* são aquelas que obtiveram maiores índices respectivamente.

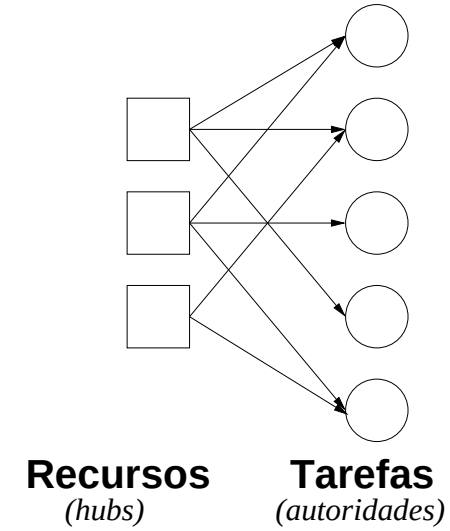



Figura 4.1: Grafo de associação de recursos à tarefas.

Em *workflow*, os *rankings* apresentam a classificação dos recursos e das tarefas em ordem crescente dos seus graus de *hub* e autoridade. Um recurso com alto grau de *hub* é um recurso versátil, pois este é capaz de executar várias tarefas com alto grau de autoridade. Já uma tarefa com alto grau de autoridade é uma tarefa cuja execução poderá ser realizada por vários recursos ou por recursos bastante versáteis. Neste sentido podemos associar esses conceitos como apresentado na tabela 4.1:

Tabela 4.1: Associação entre conceitos de Workflow e HITS.

| **HITS** | **Workflow** |
|---|---|
| *Hub* | Versatilidade do recurso para executar tarefas. |
| Autoridade | Versatilidade da tarefa em ser executadas por recursos. |
| *in-link* | Tarefa é apontada por recursos competentes para executá-la. |
| *out-link* | Recurso aponta para as tarefas as quais tem competência para executar. |

A intenção em realizar a classificação do *ranking* em ordem crescente é porque se pretende deixar a alocação dos "bons" *hubs* por último. "Bons" *hubs* são os recursos versáteis, isto é aqueles que têm competência para executar diferentes tarefas. Deixando-os por último, aumenta-se a chance de se encontrar um recurso apropriado para uma tarefa.

## 4.1 Algoritmo wf-hits

Seguindo a linha dos algoritmos propostos por [2,6,10,11,15,16,18,19,35], o wf-hits é um algoritmo para a classificação de recursos e tarefas que utiliza os conceitos de *hub* e autoridade.

O algoritmo wf-hits opera sobre um grafo dirigido $G = (N, AR)$, que representa a ligação entre os recursos e as tarefas. $N$ corresponde ao conjunto de vértices do grafo, composto pelo conjunto de recursos $R$ e de tarefas $T$, portanto, $N = R \cup T$. $AR$ corresponde ao conjunto de arcos $(r, t)$, tal que $r \in R$ e $t \in T$. O algoritmo opera sobre dois vetores $H$ e $A$ que representam, respectivamente, os graus de *hub* e de autoridade.

O wf-hits insere o conceito de "valor de alocação absoluta" (VAB) [20] que é calculado a partir de alguns parâmetros relacionados a recursos e tarefas:

**disponibilidade** - indica a ocupação de um recurso;

**adequação** $(r, t)$ - indica o quão adequado é o recurso $r$ para executar a tarefa $t$;

**urgência** $(t)$ - indica a urgência da tarefa $t$;

**conformidade** $(r, t)$ - indica a violação de regras quando o recurso $r$ executa $t$.

O VAB é utilizado para dar peso aos arcos $(r, t)$, conforme proposta similar de uma versão do HITS com pesos [38]. O algoritmo wf-hits utiliza a função $fvab : R \times T \rightarrow [0, 1]$, tal que $fvab(r, t)$ retorna o VAB de $r$ em relação a $t$. O algoritmo wf-hits proposto é ilustrado no quadro Algoritmo 3.

No algoritmo wf-hits (Algoritmo 3), inicialmente $x$ recebe o número de elementos de $R$ (linha 1) e $y$ recebe o número de elementos de $T$ (linha 2). A cada elemento dos vetores $H$ e $A$ é atribuído inicialmente o valor 1 (linhas 3–4). Em seguida, em cada iteração do laço (linhas 5–11), o grau de autoridade (linha 7) e de *hub* (linha 9) são calculados. $A[t]$ é igual à soma do produto entre $fvab(r, t)$ e $H[r]$, para todo arco $(r, t)$. $H[r]$ é igual à soma do produto entre $fvab(r, t)$ e $A[t]$, para todo arco $(r, t)$. O laço termina quando $H$ e $A$ convergirem, isto é quando os valores desses vetores não sofrerem mudança significativa[1] entre a iteração corrente e a imediatamente

[1]Neste trabalho consideramos que diferenças maiores que um patamar de $0,00000000001$ são consideradas diferenças significativas

**Algoritmo 3** Algoritmo wf-hits

1: $x \leftarrow \mid R \mid$

2: $y \leftarrow \mid T \mid$

3: $H[1..x] \leftarrow 1$

4: $A[1..y] \leftarrow 1$

5: **enquanto** $H$ e $A$ `não convergirem`

6: $\forall t \in T$

7: $$A[t] = \sum_{\forall r \mid (r,t) \in AR} fvab(r,t).H[r]$$

8: $\forall r \in R$

9: $$H[r] = \sum_{\forall t \mid (r,t) \in AR} fvab(r,t).A[t]$$

10: `Normalize` $A$ e $H$

11: **fim_enquanto**

12: `Classifique` $H$ `e` $A$ `em ordem crescente`

anterior. Ao final da execução, os vetores de *hub* e autoridade são classificados em ordem crescente (linha 12).

## 4.2 Mecanismos de distribuição de tarefas baseados no algoritmo wf-hits

Quatro mecanismos de distribuição de tarefas são propostos neste trabalho, todos tem como princípio a classificação dos recursos e tarefas através do algoritmo *wf-hits* que gera duas filas, a fila de recursos(*hubs*) e a fila de tarefas(*autoridades*). Essas filas estão ordenadas em ordem crescente dos valores de *hub* e *autoridade* e aqui são chamadas de *rankings*. Os primeiros elementos destes *rankings* são respectivamente os recursos mais especializados e as tarefas mais restritivas.

Consideramos que as tarefas mais restritivas e os recursos mais especializados trariam maior dificuldade para associação, então estas associações devem ser realizadas logo de início. À medida que os recursos vão sendo alocados, restam no *ranking* recursos mais versáteis, ou seja, recursos aptos a executar uma maior quantidade de tarefas distintas. Deste mesmo ponto de vista, à medida que as tarefas vão sendo executadas, restam tarefas menos restritivas, ou seja, tarefas que podem ser executadas por uma quantidade maior de recursos.

A distribuição de tarefas é realizada através de um algoritmo de distribuição. Este algoritmo obtém um conjunto de tarefas do sistema de *workflow*, associa os recursos do sistema às tarefas pertinentes e aplica a classificação por meio do algoritmo *wf-hits* apresentado na seção anterior. Após a classificação, é selecionada uma tarefa e alocado um recurso para realização da tarefa. A seleção de tarefas e alocação de recursos prossegue iterativamente até que todas as tarefas sejam alocadas.

### 4.2.1 Variantes do algoritmo wf-hits

Variamos o algoritmo de montagem do grafo de entrada e a forma de seleção das tarefas para alocação, resultando em quatro estratégias que são descritas a seguir.

#### Full-wf-hits

A distribuição de tarefas através da estratégia *Full-wf-hits* é realizada conforme apresentado na figura 4.2 e descrita a seguir.

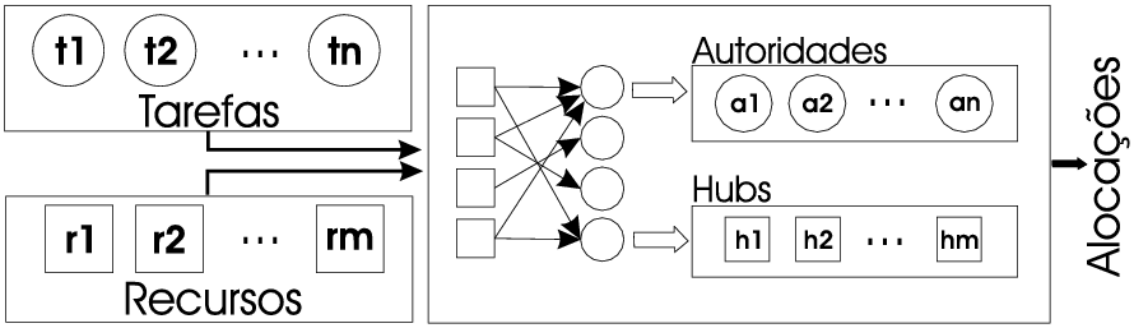



Figura 4.2: Estratégia *Full-wf-hits* de distribuição de tarefas.

Todas as tarefas coletadas pelo sistema e os recursos associados a pelo menos uma dessas tarefas formam o grafo de entrada para o algoritmo *wf-hits*. Este algoritmo gera então o *ranking* de tarefas, em ordem crescente de seu grau de autoridade e o *ranking* de recursos, em ordem crescente de seu grau de *hub*.

A tarefa $a_i$, no topo do *ranking* de autoridades é selecionada e removida. Então, o *ranking* de recursos é percorrido à partir do topo até que seja encontrado um recurso que esteja associados à tarefa $a_i$. Sendo $r_j$ este recurso, a tarefa $a_i$ é alocada ao recurso $r_j$. A distribuição prossegue iterativamente com a seleção de novo par (tarefa, recurso) até que todas as tarefas sejam alocadas.

**Random-wf-hits**

Nesta estratégia, o grafo de entrada para o algoritmo *wf-hits* é montado da mesma maneira que na estratégia *Full-wf-hits*. Entretanto, após a classificação, a seleção de uma tarefa ignora a ordem no *ranking* de autoridades e uma tarefa $t_i$ é escolhida aleatoriamente. As demais etapas são iguais as da estratégia *Full-wf-hits*.

**Selective-wf-hits**

Nesta estratégia, o grafo de entrada para o algoritmo *wf-hits* contêm apenas recursos com fator de alocação absoluta maior que o limiar de $0,6$, $absAlloc(tarefa, recurso) > 0,6$ (veja figura 4.3). Recursos com essa característica são chamados *recursos seletivos* em [20]. As demais etapas são iguais à estratégia *Full-wf-hits*.

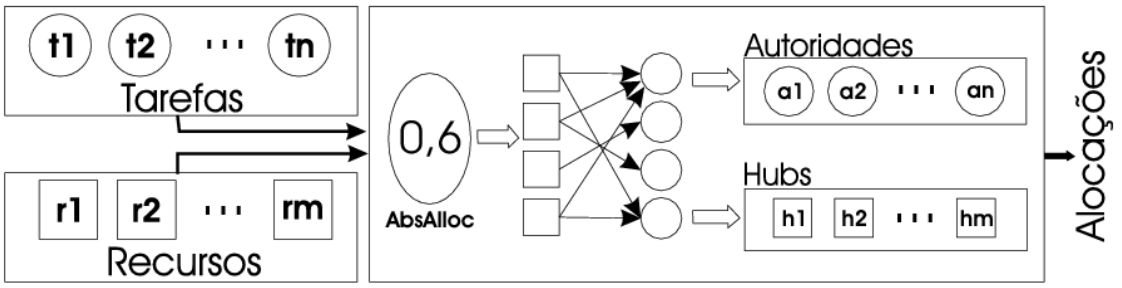



Figura 4.3: Estratégia *Selective-wf-hits* de distribuição de tarefas.

A estratégia *Selective-wf-hits* somente seleciona para a distribuição os recursos que alcançarem $absAlloc(tarefa, recurso) > 0,6$. Apesar do critério ser bastante seletivo, não ocorre inanição, ou seja, os parâmetros para o cálculo do valor de alocação absoluta do recurso são atualizados a cada iteração do algoritmo (a disponibilidade do recurso é modificada, a urgência da tarefa aumenta com o passar do tempo, ...), essas mudanças influenciam no cálculo do valor de alocação absoluta dos recursos garantindo sempre que existam recursos com $absAlloc(tarefa, recurso) > 0,6$.

**Dynamic-wf-hits**

Nesta estratégia, procura-se inicialmente, montar o grafo de entrada para o algoritmo *wf-hits* com recursos com fator de alocação absoluta maior que o limiar de $0,6$ ($absAlloc(tarefa, recurso) > 0,6$). Entretanto, caso não exista recurso com fator

de alocação absoluta que se enquadre neste limiar, o limiar é decrementado em $-0,1$, repetindo-se o decremento até que seja encontrado pelo menos um recurso com fator de associação à tarefa superior ao novo limiar(veja figura 4.4). As demais etapas são iguais à estratégia *Full-wf-hits*.

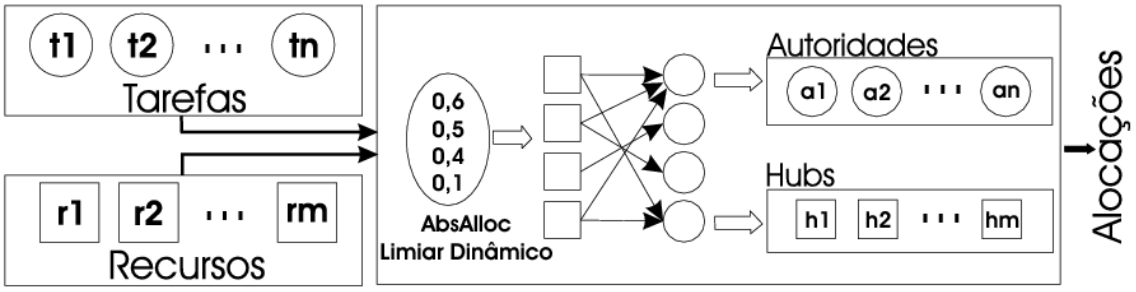



Figura 4.4: Estratégia *Dynamic-wf-hits* de distribuição de tarefas.

# Capítulo 5

# Experimentos

Nesta seção são apresentados o cenário da proposta, a estratégia para realização dos experimentos, e o ambiente de simulação adotado.

## 5.1 Cenário

Um cenário de *workflow* foi definido com a intenção de prover meios para a comparação deste trabalho com os trabalhos de [20,29,32]. Para garantir a conformidade dos experimentos de [20,32], reproduziu-se o mesmo cenário utilizado. As definições e os parâmetros adotados também seguem a linha destes trabalhos. Este cenário é apresentado na figura 5.1.


Figura 5.1: Processo de negócio utilizado nos experimentos.

O processo de negócio adotado neste cenário possui 3 tarefas em sequência (Tarefa A, Tarefa B e Tarefa C). Uma tarefa é uma unidade de trabalho que pode ser realizada de forma manual, semi-automática ou automática. As tarefas estão associadas respectivemente aos papéis (Papel 1, Papel 2 e Papel 3). Um papel é um modelo abstrato que reflete os aspectos do sistema, e que agrupa recursos afins. Um recurso é um participante concreto de um sistema de *workflow*, ele é incumbido de executar as tarefas. A cada papel estão associados um conjunto de recursos (cinco

neste experimento, vide tabela 5.2). Desta forma, um recurso estará apto a executar as tarefas que estão associadas aos papéis os quais ele pertence.

O tempo da tarefa é o tempo definido para que a tarefa seja executada. Já o tempo do recurso é o tempo que o recurso demanda para executar uma determinada tarefa. As tarefas foram definidas com um tempo variando em torno de um padrão de execução de $10ut$(unidades de tempo). Porém para que o ambiente de simulação se aproximasse mais da realidade, foram gerados tempos distintos para os recursos.

Os experimentos realizados consistiram em gerar automaticamente várias instâncias do processo de negócio apresentado. Cada instância de execução do processo é gerada levando em consideração diferentes cargas de trabalho. A carga de trabalho (*workload*) é considerado um aspecto crucial no desempenho de um sistema de *workflow* [29]. Esta é definida como o número de tarefas que estão prontas para serem executadas no sistema. Foram definidos alguns parâmetros que afetam diretamente a carga de trabalho das instâncias para evitar que os experimentos se tornassem homogêneos (Veja tabela 5.1). A diversificação dos tempos das tarefas e dos tempos dos recursos tornam o ambiente de experimentação mais heterogêneo.

Tabela 5.1: Parâmetros de Simulação

| | baixa | média | alta |
|---|---|---|---|
| Carga média de sistema | 0,2 | 0,6 | 1,0 |
| Variabilidade de tarefas | 0,2 | 0,8 | 1,4 |
| Variabilidade de recursos | 0,2 | 0,8 | 1,4 |

O parâmetro "carga média de sistema" mede a proporção com que as tarefas ficam prontas para serem executadas. Ele determina o fluxo de novas tarefas que chegam ao sistema. Por exemplo, uma carga média de sistema com patamar $0,2$ indica que somente 20% da capacidade do sistema será utilizada. A diversificação da carga de trabalho das instâncias de um processo *workflow* é obtido a partir de uma distribuição de *Poisson* com $\lambda = (l.n)/tm$, onde $l$ é a carga média de sistema, $n$ é o numero de recursos e $tm$ é o tempo médio das tarefas.

O parâmetro variabilidade de tarefas determina como será a geração da amostra populacional diversificada de tarefas. As tarefas geradas terão um tempo de processamento variando em torno do valor padrão($10ut$). Para a tarefa $t_j$, a média

de tempo de processamento $\overline{p_j}$ é gerada com base em uma distribuição normal com média $tm$ (tempo médio das tarefas) e desvio padrão *varTarefa*. Assim, $\overline{p_j} \approx N(tm, (tm \times varTarefa)^2)$. Por exemplo, considerando uma variabilidade de tarefas $0,2$, teremos um tempo de processamento variando no intervalo $[8ut, 12ut]$.

A variabilidade dos recursos indica o percentual de variação dos tempos que os recursos consomem para executar as respectivas tarefas. O cálculo do tempo de processamento $P_{ij}$ de uma tarefa $t_i$ por um recurso $r_j$, ocorre de maneira similar à variação de tarefas. $P_{ij}$ é calculado através de uma distribuição normal onde $p_{ij} \approx N(\overline{p_j}, (\overline{p_j} \times varRecurso)^2)$. Por exemplo, caso uma tarefa tenha seu tempo de execução definido como $10ut$ e a variabilidade de recursos seja $0,2$, significa que o recurso executará aquela tarefa com um tempo que irá variar no intervalo $[8ut, 12ut]$.

Conforme os parâmetros apresentados na tabela 5.1, o sistema tem a capacidade de gerar 540 condições experimentais, ou seja, para cada uma das 27 combinações possíveis da tabela são geradas aletoriamente 20 instâncias de problemas. Cada estratégia de distribuição será avaliada sob estas condições.

Outros parâmetros para a realização da simulação são a *disponibilidade* de cada recurso, a *urgência* de uma tarefa, a *conformidade* de um recurso para com uma tarefa, e a *adequação* de um recurso para executar uma determinada tarefa. Em todos os experimentos, os valores de *disponibilidade*, *urgência*, e *conformidade* estão de acordo com o apresentado por Kumar *et, al.* [20], onde são fixados em 1,0, 0,8 e 1,0, respectivamente. Os valores de *adequação* são apresentados na tabela 5.2. Esta tabela mostra também o mapeamento *papel* x *tarefa*, associando os recursos pertencentes aos papéis às suas respectivas tarefas. Um pequeno contingente de recursos (cinco recursos associados a cada papel) foi adotado para os experimentos. É comum em uma organização que um número pequeno de recursos seja atribuído à cada papel, sendo estes suficientes para a simulação de um cenário de *workflow*.

A *adequação* é representada na tabela pela intersecção entre um recurso e uma tarefa (colunas **Tarefa A**, **Tarefa B**, **Tarefa C**) e indica o quão adequado um recurso é para executar uma determinada tarefa. Os valores de *adequação* podem estar em um intervalo $[0,1]$ e foram atribuídos aleatoriamente com valores do conjunto $\{0,4;\ 0,5;\ 0,6;\ 0,7;\ 0,8;\ 0,9;\ 1,0\}$.

Tabela 5.2: Mapeamento *papel* x *recurso* e adequação *recurso* x *tarefa*

| | | **Adequação** | | |
|---|---|---|---|---|
| **Papéis** | **Recursos** | **Tarefa A** | **Tarefa B** | **Tarefa C** |
| papel1 | $r_1$ | 1,0 | 0 | 0 |
| | $r_2$ | 0,9 | 0 | 0 |
| | $r_3$ | 0,8 | 0 | 0 |
| | $r_4$ | 0,8 | 0 | 0 |
| | $r_5$ | 0,8 | 0 | 0 |
| papel2 | $r_6$ | 0 | 0,8 | 0 |
| | $r_7$ | 0 | 0,8 | 0 |
| | $r_8$ | 0 | 0,6 | 0 |
| | $r_9$ | 0 | 1,0 | 0 |
| | $r_{10}$ | 0 | 0,7 | 0 |
| papel3 | $r_{11}$ | 0 | 0 | 0,4 |
| | $r_{12}$ | 0 | 0 | 1,0 |
| | $r_{13}$ | 0 | 0 | 0,9 |
| | $r_{14}$ | 0 | 0 | 0,8 |
| | $r_{15}$ | 0 | 0 | 0,5 |

## 5.2 Ambiente de Experimentação

As estratégias de implementação do wf-hits foram simuladas pela ferramenta Lambari-SWTE (*Simple Workflow Test Environment*) [34]. O Lambari é um simulador de *workflow*, implementado em Java, que dá apoio à execução de vários processos simultâneos.

A figura 5.2 ilustra o funcionamento do simulador Lambari. Inicialmente, o Lambari analisa o arquivo XPDL[1] (veja exemplo no apêndice A) que contém dados relativos ao processo (fluxo de tarefas), aos recursos e as tarefas. O Lambari é capaz de simular todas as estratégias ou uma estratégia específica. Ao final da simulação, o Lambari apresenta o grafo de sequenciamento de tarefas, a listagem correspondente a esse grafo, e gráficos comparativos entre as estratégias simuladas.

[1]XML Process Definition Language [37]

Originalmente o escalonador do Lambari contemplava as estratégias definidas nos trabalhos de Kumar, *et. al.* e Veloso [20,32], porém, para atender as simulações desse experimento, foram desenvolvidas e acopladas ao escalonador do Lambari as quatro estratégias propostas nesse trabalho. Dessa forma, os resultados obtidos a partir dos experimentos apresentados aqui puderam ser comparados com os resultados das demais estratégias de distribuição.

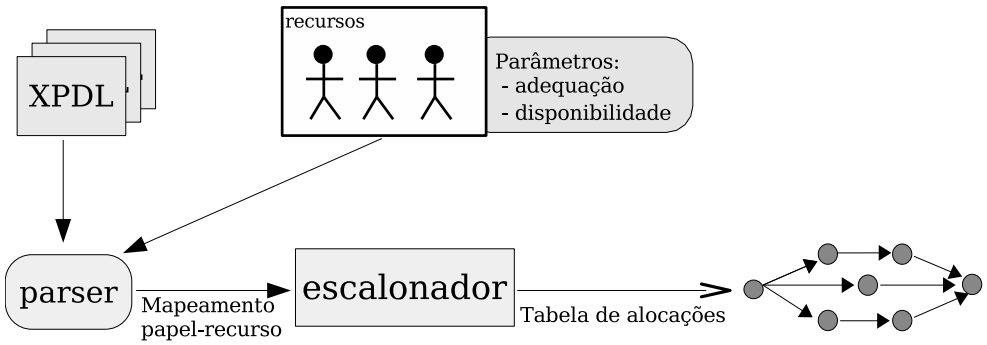



Figura 5.2: Funcionamento do Lambari-SWTE

A figura 5.3 mostra o grafo de sequenciamento de tarefas oferecido pelo Lambari. O grafo em questão corresponde à simulação de quatro instâncias do processo da figura 5.1. Os vértices do grafo correspondem a tarefas. Os vértices **start** e **end** são tarefas fictícias que representam o início e o fim de todas as instâncias. Os arcos horizontais ligam tarefas de uma mesma instância. Assim, a sequência de instâncias de tarefa *Start, A.1, B.1, C.1, End* corresponde à execução da instância 1 do processo citado. Os arcos verticais indicam que as tarefas ligadas por estas são executadas por um mesmo recurso (que não é apresentado explicitamente no grafo). Deste modo, as instâncias de tarefa *A.2 e A.4* são executadas pelo mesmo recurso, assim como as instâncias *B.1 e B.4*, e *C.3 e C.4*. Finalmente, os arcos estão rotulados com o tempo de execução da tarefa. A tarefa *Start*, por exemplo, demanda zero unidade de tempo, por sua vez, a tarefa *A.2* demanda 12 (doze) unidades de tempo para sua execução.

## 5.3 Estratégias para a distribuição de tarefas

Sistemas de *workflow* realizam a distribuição de tarefas através dos mecanismos *Pull* e *Push* [29]. Estas estratégias apresentam resultados divergentes em relação a qua-

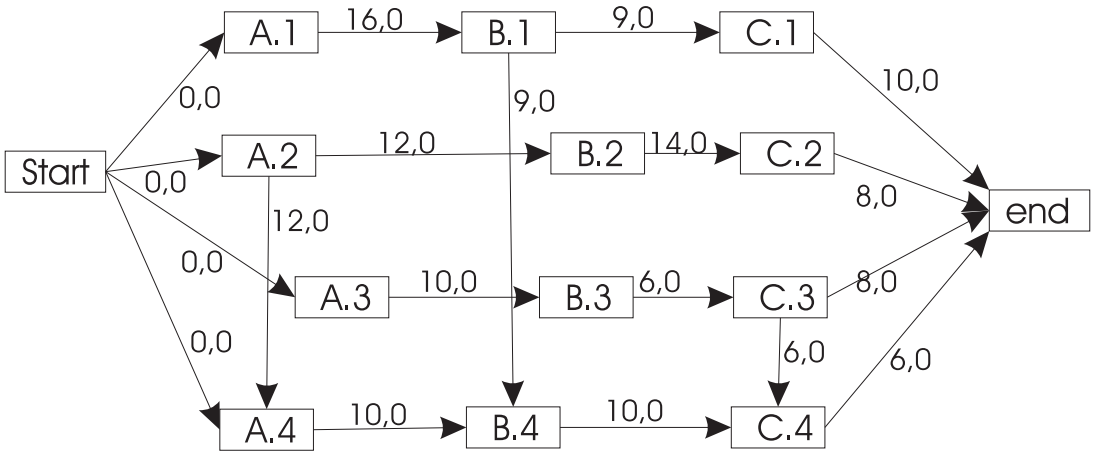



Figura 5.3: Grafo de sequenciamento.

lidade e tempo de execução do *workflow*. Enquanto a estratégia *Push* alcança altos níveis de qualidade na execução do *workflow*, a *Pull* obtém os menores intervalos de tempo de execução. Por exemplo, um sistema de *workflow* que adote o mecanismo *Push* de distribuição, devido a característica de entregar a tarefa somente ao recurso mais competente para executá-la, alcança altos índices de qualidade em prejuízo do tempo de execução, em contrapartida, o mecanismo *Pull* apresenta as tarefas ao conjunto dos recursos, e um destes recursos decide por executá-la, desta forma a distribuição é realizada em intervalos de tempo menores, porém com baixos níveis de qualidade.

A adoção de somente um dos dois mecanismos para a distribuição não é uma boa estratégia [20, 29, 32]. O sistema de *workflow* que adota o mecanismo *Push*, pode enviar uma tarefa para um recurso que não está disponível, assim a tarefa pode permanecer no sistema até que o determinado recurso esteja novamente disponível, acarretando em filas de espera. Por outro lado, um sistema que adote somente o mecanismo *Pull* pode ter as tarefas executadas por recursos que não são os mais aptos para tal, degradando assim a qualidade de execução das tarefas.

Outros mecanismos de distribuição foram concebidos no intuito de tentar criar um balanço entre qualidade e desempenho. Estes mecanismos foram definidos tendo como base as estratégias *Push* e *Pull*.

São considerados para o nosso experimento 12 estratégias de distribuição de tarefas. Entre as estratégias adotadas estão inseridas as estratégias *Push* e *Pull*, as estratégias *Receptive, Typical* e *Selective* [20], as estratégias *Sel-push, Sel-push-10* e

*Sel-push-ft* [32] e quatro novas estratégias, *Full-wf-hits, Random-wf-hits, Selective-wf-hits e Dynamic-wf-hits*, propostas neste trabalho.

Seja $R$ o conjunto dos recusos, tal que, $R = \{r_1, r_2, ..., r_n\}$, e $T$ o conjunto das tarefas, tal que, $T = \{t_1, t_2, ..., t_m\}$, os mecanismos definidos em conformidade com as estratégias *Push*, *Pull*, *Receptive*, *Typical*, *Selective*, *Sel-push*, *Sel-push-10* e *Sel-push-ft* realizam a distribuição da seguinte forma:

- *Push*: Um recurso $r$ é escolhido pelo sistema para executar uma tarefa $t$;
- *Pull*: Uma tarefa $t$ é visível a todos os recursos $r \in R' \subseteq R$ tal que $R'$ é o subconjunto de recursos qualificados a executar $t$;
- *Receptive*: Uma tarefa $t$ é visível somente aos recursos $r$ que possuirem fator de alocação absoluta $fvab(t, r)$ acima de um limiar $0,5$. Este limiar indica que os recursos são receptivos;
- *Typical*: Uma tarefa $t$ é visível somente aos recursos $r$ que possuirem fator de alocação absoluta $fvab(t, r)$ acima de um limiar $0,4$. Este limiar indica que os recursos são típicos;
- *Selective*: Uma tarefa $t$ é visível somente aos recursos $r$ que possuirem fator de alocação absoluta $fvab(t, r)$ acima de um limiar $0,6$. Este limiar indica que os recursos são seletivos;
- *Sel-push*: Uma tarefa $t$ é entregue ao recurso $r$ mais apto, se $r$ estiver disponível. Caso contrário, um recurso disponível com aptidão mais próxima de $r$ é selecionado.
- *Sel-push-10*: Uma tarefa $t$ é entregue ao recurso mais disponível entre o conjunto de recursos em uma vizinhança de 10% do valor $fvab(t, r)$, onde $r$ é o recurso mais apto.
- *Sel-push-ft*: Uma tarefa $t$ é entregue a um recurso $r$ de forma semelhante à estratégia *Sel-push*, entretanto considera a *disponibilidade* e o tempo restante para concluir a tarefa atual.

## 5.4 Resultados

Foram realizados experimentos para as doze estratégias apresentadas neste trabalho. Os resultados são apresentados na forma de gráficos de qualidade da distribuição e de tempo de execução das simulações realizadas e através do teste-t pareado (*pairwise t-test*) [14]. O teste-t pareado é uma ferramenta estatística para análise de dados relacionados. Segundo Graziano e Raulin, o teste-t pareado é aplicado quando é necessário verificar a existência de diferenças significativas entre dois grupos [14]. Para ilustrar se houveram diferenças significativas entre as estratégias simuladas segundo o teste-t pareado, serão utilizadas tabelas de sinais.

A tabela de sinais apresenta o confrontamento entre as estratégias experimentadas e mostra se houveram diferenças através dos sinais "+", "−" e "=". O sinal "+" indica que a estratégia relacionada na linha da tabela foi "melhor"[2] que a estratégia relacionada na coluna correspondente da tabela. O sinal "−" indica que a estratégia da linha obteve "piores"[3] resultados em relação a estratégia da coluna correspondente. O sinal "=" indica que não houveram diferenças significativas entre as duas estratégias comparadas. O sinal "n.a." indica que a comparação não se aplica para o caso.

Será apresentada uma tabela para cada análise realizada. A tabela apresentará os resultados na forma "sinal/sinal" (por exemplo +/−), onde o primeiro sinal representa o resultado da análise de tempo de execução e o segundo sinal representa o resultado da análise de qualidade.

Quatro análises foram realizadas no intuito de comparar as estratégias propostas com as da literatura. A primeira análise compara as estratégias baseadas no algoritmo *wf-hits* entre si. Após realizada a primeira análise, constatou-se que a estratégia *Selective-wf-hits* era a que apresentava maior compromisso entre desempenho e qualidade na distribuição entre as estratégias propostas, desta forma, ela fora

[2]No contexto deste trabalho o termo "melhor" significa: em caso de análise do tempo de execução "melhor desempenho", ou seja, menores tempos de execução, já em caso de análise de qualidade, "maiores índices de qualidade"

[3]No contexto deste trabalho o termo "pior" significa: em caso de análise de tempo de execução, "pior desempenho", ou seja maiores tempos de execução, já em caso de análise da qualidade, "piores índices de qualidade"

selecionada para comparação com as demais estratégias da literatura. A segunda compara as estratégias *Push, Pull* e *Selective-wf-hits*, o objetivo desta comparação, é a de situar a estratégia proposta selecionada entre os limiares de desempenho e qualidade. A terceira, compara as estratégias *Selective, Sel-Push* e *Selective-wf-hits*, com o intuito de confrontar a *Selective-wf-hits* com as estratégias destacadas como principais por Kumar *et, al.* [20] e Veloso [32]. Finalmente, é apresentada uma comparação global incluindo todas as estratégias experimentadas.

Os gráficos foram traçados a partir da média obtida das 20 simulações realizadas em cada estratégia. Dois tipos de gráficos são apresentados: tempo de execução e qualidade. Os gráficos de tempo comparam carga de trabalho do *workflow* versus tempo de execução. Os gráficos de qualidade comparam carga de trabalho de *workflow* versus índice de qualidade. Os índices de carga de trabalho apresentados no gráfico correspondem às combinações daquelas apresentadas na tabela 5.1.

### 5.4.1 Análise das estratégias baseadas no algoritmo wf-hits

Quatro estratégias foram experimentadas neste trabalho (*Full-wf-hits*, *Random-wf-hits*, *Dynamic-wf-hits* e *Selective-wf-hits*). Esta análise dos resultados obtidos tem o objetivo de selecionar a estratégia de destaque entre estas. Os gráficos das figuras 5.4 e 5.5 apresentam as comparações de tempo de execução e de qualidade e a tabela 5.3 sumariza os resultados do teste estatístico para as estratégias propostas neste trabalho.

Conforme o gráfico da figura 5.4 observa-se que a estratégia *Full-wf-hits* obteve os menores tempos de execução em relação às demais baseadas no algoritmo *wf-hits*, porém, esta estratégia não apresenta bons índices de qualidade (veja no gráfico da figura 5.5). Ainda no gráfico da figura 5.4, observa-se também que a estratégia *Selective-wf-hits* obteve o segundo melhor desempenho.

A comparação dos índices de qualidade pode ser observada no gráfico da figura 5.5, nele é possível observar que a estratégia *Selective-wf-hits* obteve os maiores índices de qualidade. A tabela 5.3 relaciona as quatro estratégias. Nela podemos observar que segundo o teste-t pareado a estratégia *Selective-wf-hits* obteve melhor desempenho quando comparada com as estratégias *Random-wf-hits* $(18,72\%)$ e *Dynamic-wf-hits* $(24,03\%)$ e desempenho inferior a estratégia *Full-wf-*

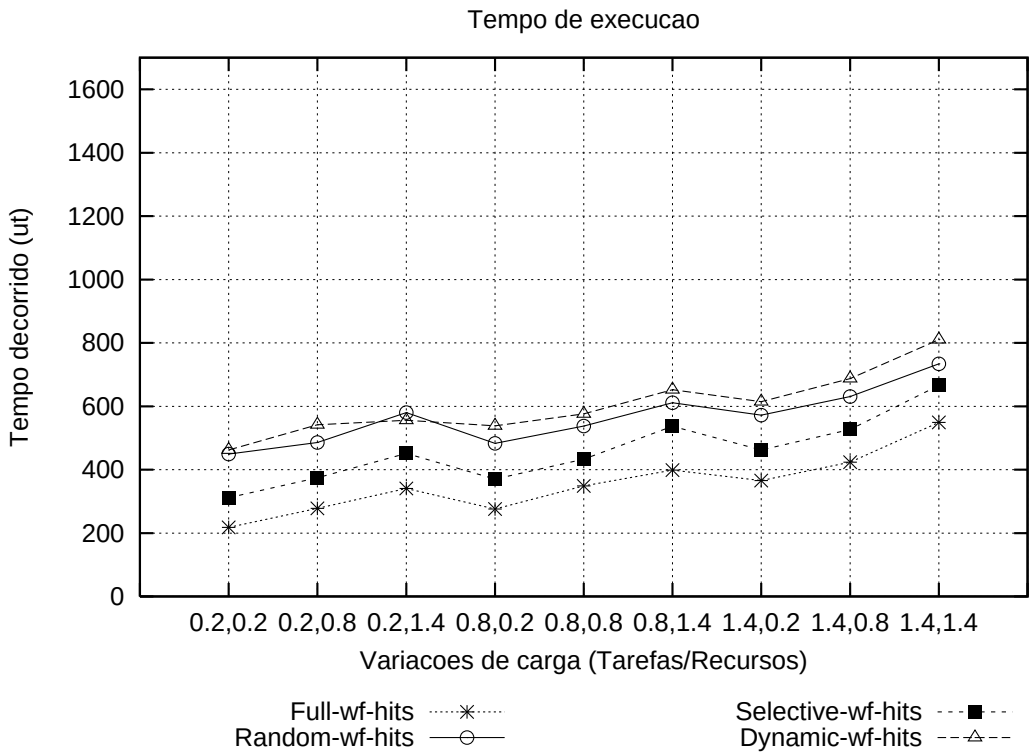



Figura 5.4: Gráfico "Tempo de execução" para as estratégias baseadas no algoritmo HITS.

*hits* ($-29,17\%$) (sinais do lado esquerdo de cada coluna). No âmbito da qualidade (sinais do lado direito de cada coluna), a estratégia *Selective-wf-hits* apresentou melhores resultados que as demais baseadas no algoritmo *wf-hits* (média de $11,6\%$ superior).

Tabela 5.3: Teste estatístico para as estratégias baseadas no algoritmo wf-hits

| | Full-wf-hits | Random-wf-hits | Selective-wf-hits | Dynamic-wf-hits |
|---|---|---|---|---|
| Full-wf-hits | n.a. | +/− | +/− | +/+ |
| Random-wf-hits | −/+ | n.a. | −/− | +/+ |
| Selective-wf-hits | −/+ | +/+ | n.a. | +/+ |
| Dynamic-wf-hits | −/− | −/− | −/− | n.a. |

A estratégia *Selective-wf-hits* é a que apresenta o melhor compromisso entre tempo de execução e qualidade dentre as propostas neste trabalho. Assim sendo, esta estratégia foi selecionada como a representante que será comparada com as estratégias representantes dos demais trabalhos da literatura.

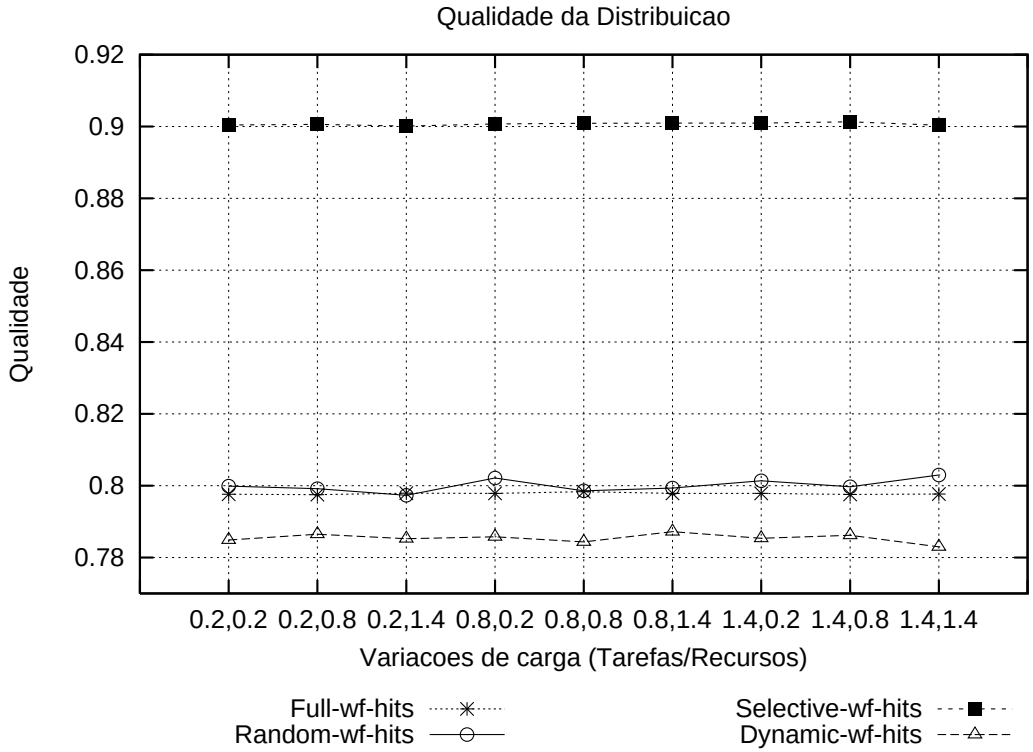



Figura 5.5: Gráfico "Qualidade da distribuição" para as estratégias baseadas no algoritmo HITS.

### 5.4.2 Comparação dos resultados das estratégias Pull, Push e Selective-wf-hits

As estratégias *Push* e *Pull* representam, respectivamente, limiares de tempo e qualidade. A literatura considera que a estratégia *Push* representa um limiar ideal de qualidade, enquanto a estratégia *Pull* como um limiar ideal de tempo [20]. A comparação aqui apresentada tem o objetivo de situar a estratégia *Selective-wf-hits* entre esses limiares. Os gráficos das figuras 5.6 e 5.7 apresentam essas comparações.

Observa-se no gráfico da figura 5.6 que a estratégia *Selective-wf-hits* apresenta um desempenho acentuadamente superior em termos de tempo de execução quando comparada à estratégia *Push* e apresenta um desempenho próximo ao da estratégia *Pull*. Em termos de qualidade (gráfico da figura 5.7) pode-se perceber que a estratégia *Selective-wf-hits* se coloca entre os limiares estabelecidos pelas estratégias *Push* e *Pull*.

Uma análise no gráfico da figura 5.11 reforça essa conclusão e mostra que a estratégia *Selective-wf-hits* obteve os melhores índices de qualidade das demais es-

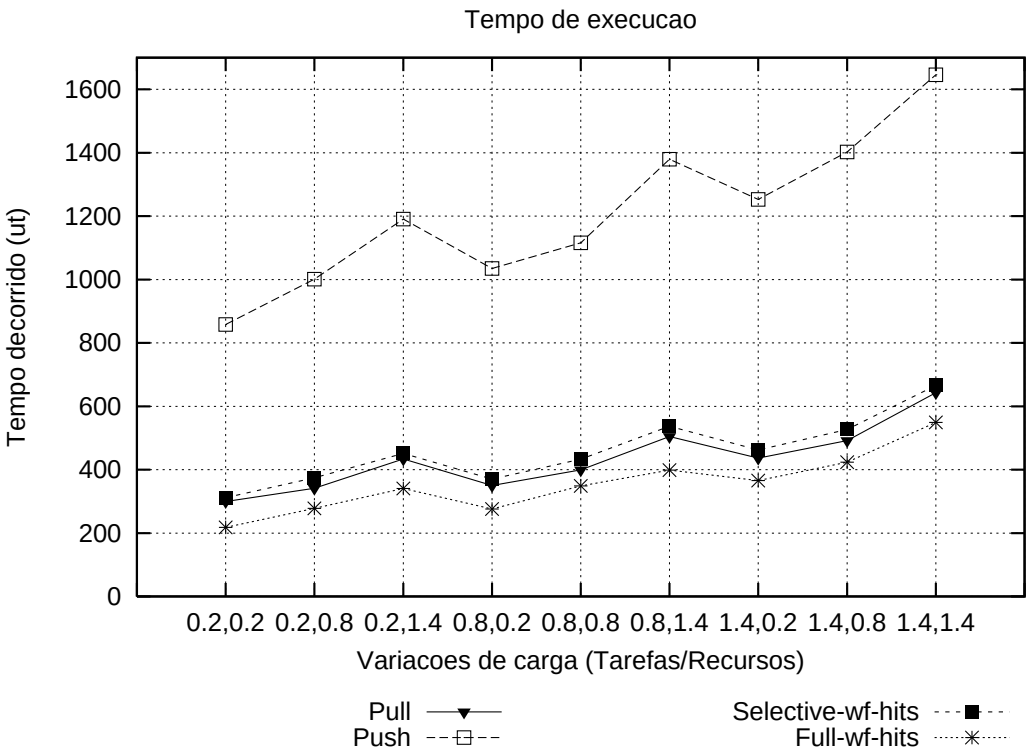



Figura 5.6: Gráfico "Tempo de execução" para as estratégias *Push, Pull e Selective-wf-hits.*

tratégias anteriormente propostas e que se posiciona muito próximo do limiar estabelecido pela estratégia *Push.*

O teste-t pareado indica diferenças significativas entre a estratégia *Selective-wf-hits* e as estratégias *Push* e *Pull* quando comparados os resultados de tempo de execução e qualidade da distribuição. A tabela 5.4 apresenta as diferenças entre estas estratégias.

Tabela 5.4: Teste estatístico para as estratégias Pull, Push e Selective-wf-hits

| | Pull | Push | Selective-wf-hits |
|---|---|---|---|
| Pull | n.a. | +/− | +/− |
| Push | −/+ | n.a. | −/+ |
| Selective-wf-hits | −/+ | +/− | n.a. |

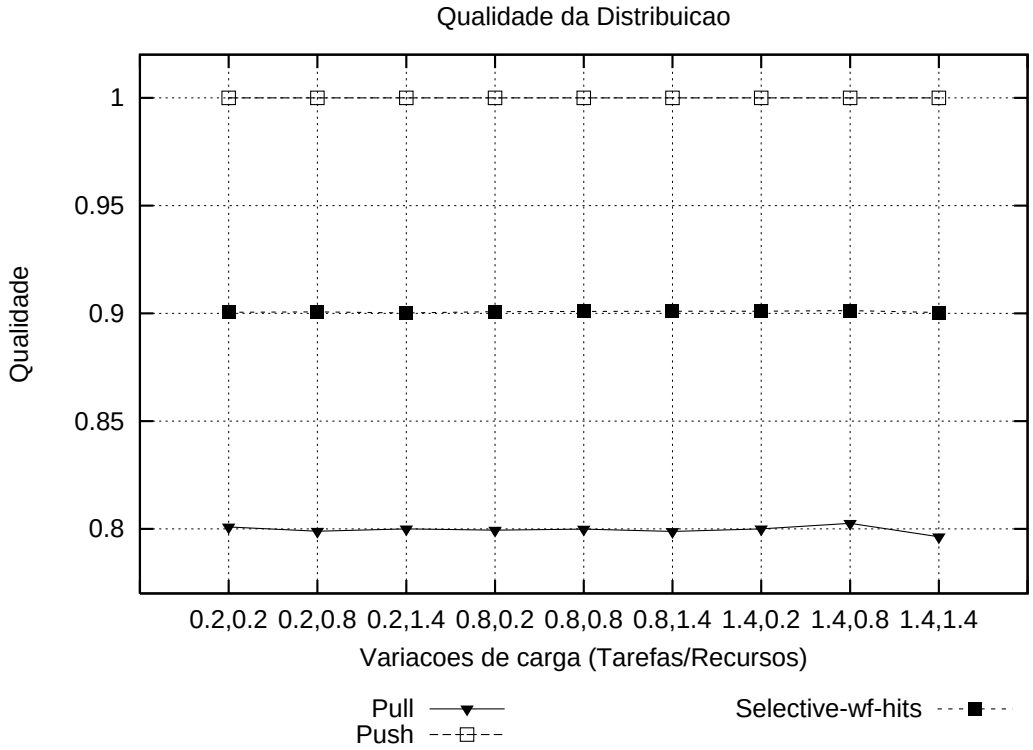



Figura 5.7: Gráfico "Qualidade da distribuição" para as estratégias *Push, Pull e Selective-wf-hits*.

### 5.4.3 Comparação dos resultados das estratégias Selective, Sel-Push e Selective-wf-hits

Uma outra comparação é realizada entre a *Selective-wf-hits* e as estratégias *Selective*, e *Sel-Push*, neste caso, o objetivo é de posicionar a estratégia *Selective-wf-hits* entre as melhores estratégias de distribuição apresentadas em trabalhos similares.

Os gráficos 5.8 e 5.9 comparam as estratégias *Selective*, *Sel-Push* e *Selective-wf-hits*. O gráfico da figura 5.8 mostra uma vantagem da estratégia *Selective-wf-hits* sobre a estratégia *Sel-Push* [32], no que se refere a tempo de execução. Os resultados do teste-t pareado indicam que as diferenças são estatísticamente muito significativas. A diferença entre as médias dos resultados obtidos pelas duas estratégias são da ordem de $8,38\%$, ou seja, a estratégia *Selective-wf-hits* obteve melhor desempenho em relação a estratégia *Sel-push*. Em comparação com a estratégia *Selective* de Kumar *et, al.* [20], o teste-t pareado indica que as diferenças são extremamente significativas. A estratégia *Selective-wf-hits* apresenta desempenho superior com tempo de execução 25% menor em relação à estratégia de Kumar *et, al.*.

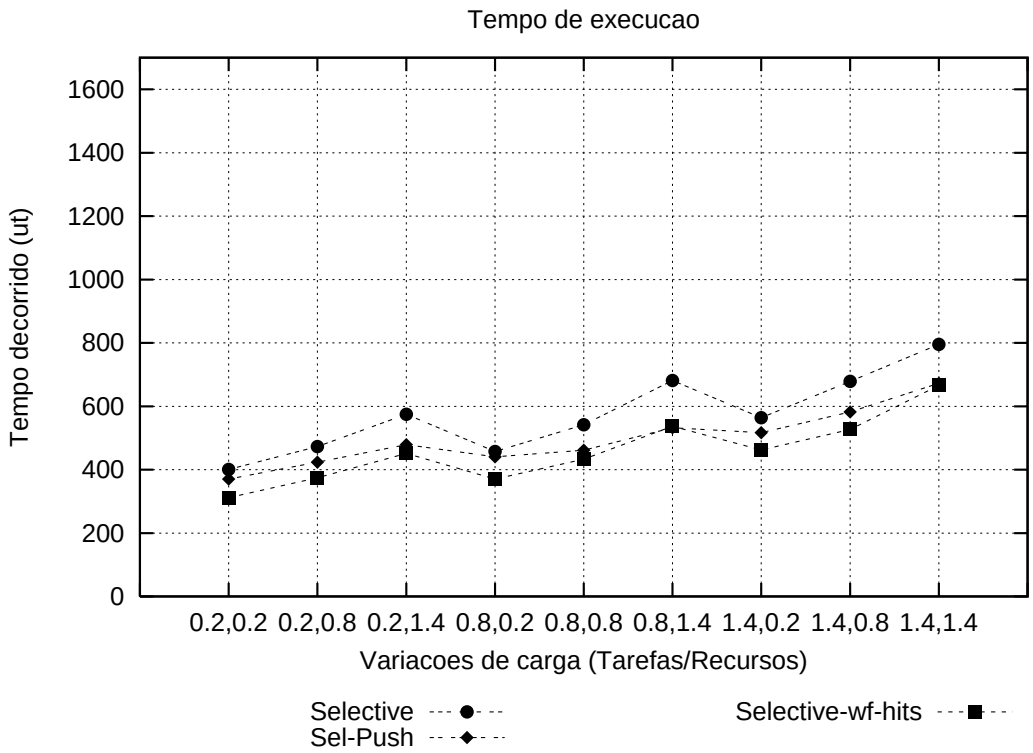



Figura 5.8: Gráfico "Tempo de execução" para as estratégias *Selective, Sel-Push e Selective-wf-hits*.

A qualidade da execução (gráfico da figura 5.9) da estratégia *Selective-wf-hits* alcançou patamares ligeiramente superiores aos resultados dos mecanismos *Selective* e *Sel-Push*. Segundo o teste-t pareado, a comparação com a estratégia *Sel-Push* apresenta diferença significativa. A diferença das médias das duas estratégias é da ordem de aproximadamente 1%. A comparação com a estratégia *Selective* não apresenta diferenças significativas segundo o teste-t pareado. Estas duas comparações mostram que a estratégia *Selective-wf-hits* apresenta melhores resultados de tempo de execução em relação às estratégias de [32] e [20], e de qualidade em relação à estratégia de [32]. A comparação com a estratégia de de Kumar *et, al.* [20] indica que não existe diferença significativa entre elas, porém, com a grande diferença (25%) no quesito tempo de execução, entende-se que a *Selective-wf-hits* mantém um maior compromisso entre desempenho e qualidade na execução das instâncias de *workflow*.

A tabela 5.5 mostra as diferenças entre as estratégias seletivas. Através da tabela é possível corroborar as informações dos gráficos 5.8 e 5.9. Observe que a estratégia *Selective-wf-hits* apresenta o sinal "+" no lado esquerdo das colunas relativas as duas

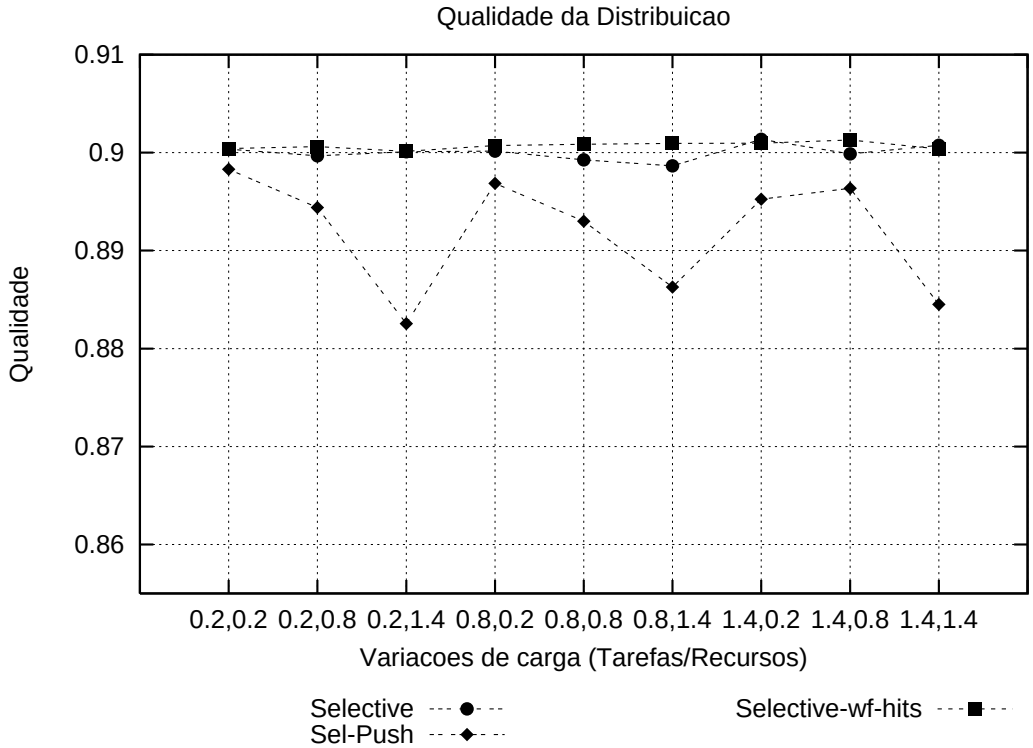



Figura 5.9: Gráfico "Qualidade da distribuição" para as estratégias *Selective, Sel-Push e Selective-wf-hits*.

outras estratégias, o que indica que esta foi melhor que as demais no quesito tempo de execução. Em relação a comparação do quesito qualidade, na coluna correspondente a estratégia *Selective* (lado direito) aparece o sinal "=" o que indica que segundo o teste-t pareado, não houve diferenças significativas entre estas estratégias. Na comparação com a estratégia *Sel-push*, a estratégia proposta obteve resultados que indicam índices de qualidade ligeiramente superiores.

### 5.4.4 Análise global do tempo de processamento e da qualidade da distribuição

Os resultados obtidos a partir de todas as estratégias simuladas foram concentrados em gráficos de tempo de execução e qualidade com o intuito de situar a estratégia *Selective-wf-hits* junto as demais estratégias simuladas (as estratégias *Random-wf-hits* e *Dynamic-wf-hits* foram omitidas propositadamente para facilitar a leitura dos respectivos gráficos). As figuras 5.10 e 5.11 demonstram os resultados obtidos por estas simulações.

Tabela 5.5: Teste estatístico para as estratégias Selective, Sel-Push e Selective-wf-hits

| | Selective | Sel-Push | Selective-wf-hits |
|---|---|---|---|
| Selective | n.a. | $-/+$ | $-/=$ |
| Sel-Push | $+/-$ | n.a. | $-/-$ |
| Selective-wf-hits | $+/=$ | $+/+$ | n.a. |

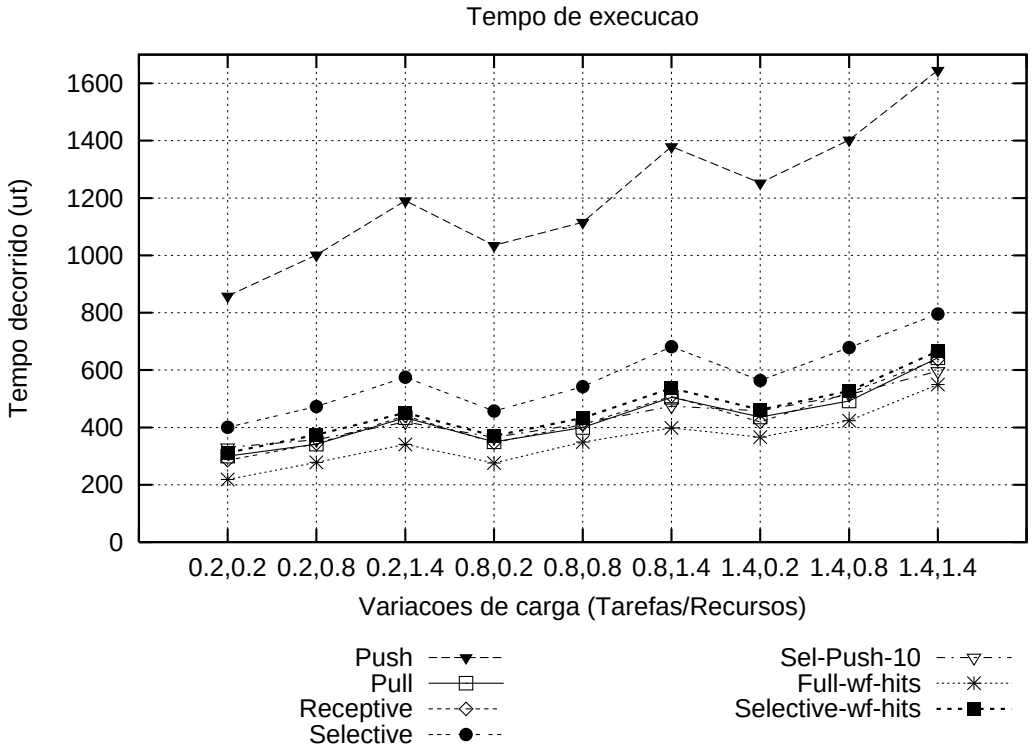



Figura 5.10: Gráfico "Tempo de execução".

No gráfico da figura 5.10 observa-se que as estratégias *Full-wf-hits*, *Pull*, *Receptive* e *Selective-wf-hits*, nesta ordem, são as que apresentam menores índices de tempo para as simulações realizadas. Um destaque deve ser feito à estratégia *Full-wf-hits*, o gráfico de tempo de execução apresentado mostra que esta estratégia obteve o melhor desempenho neste quesito, sendo até melhor que a estratégia *Pull* citada como limite inferior em [20]. As estratégias *Pull* e *Receptive* que apresentaram desempenho melhores que a *Selective-wf-hits* não apresentam boa qualidade na distribuição de tarefas.

Segundo o gráfico de qualidade da figura 5.11, as estratégias *Push*, *Selective-wf-hits*, *Selective* e *Sel-Push-10*, nesta ordem, obtiveram os melhores patamares de

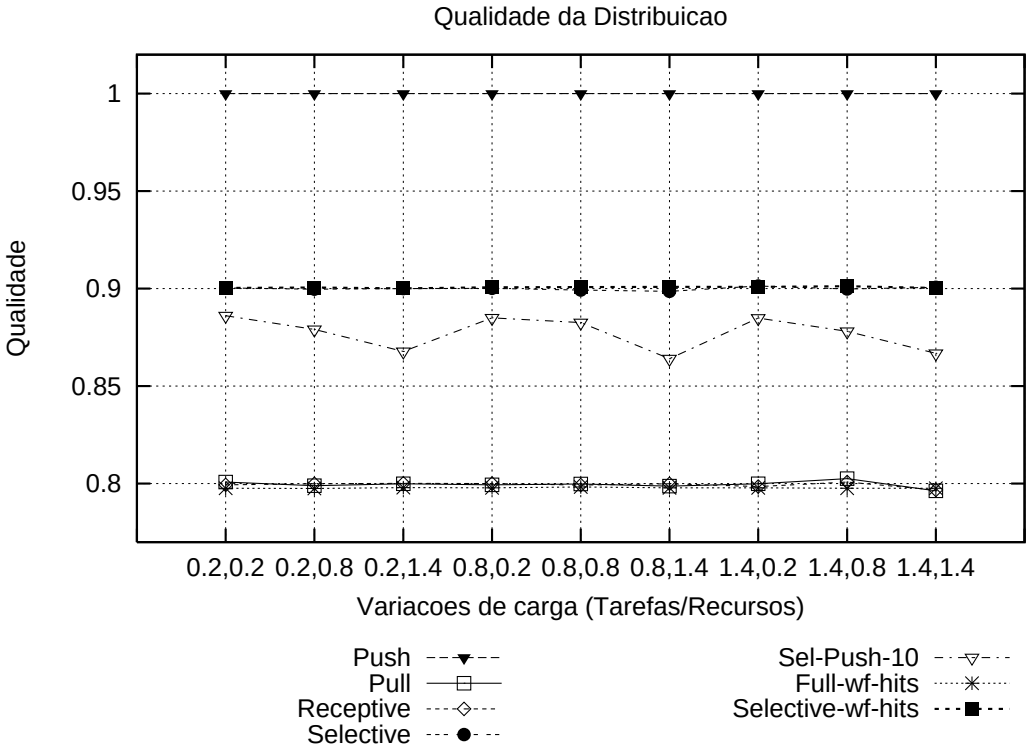



Figura 5.11: Gráfico "Qualidade da distribuição".

qualidade do conjunto das estratégias testadas. A estratégia *Push* apresenta valores que representam a máxima qualidade do processo de distribuição (valores $1,0$) para todas as cargas de trabalho impostas ao simulador. Nestes termos, a estratégia *Selective-wf-hits* apresenta resultados melhores que os das demais estratégias da literatura.

Os resultados obtidos das comparações de tempo de execução e de qualidade mostram que a estratégia *Selective-wf-hits*, proposta neste trabalho, apresenta melhores resultados quando se trata desses dois quesitos. Ela apresenta um maior balanceamento dos resultados de qualidade e tempo de execução da ditribuição de tarefas em sistemas de workflow.

# Capítulo 6

# Conclusão e trabalhos futuros

A distribuição de tarefas em sistemas de workflow é uma atividade importante para a eficiência e eficácia dos sistemas de *workflow*. A adoção de políticas mais elaboradas para a distribuição de tarefas pode diminuir o tempo de execução e melhorar a qualidade dos casos de *workflow*. Políticas de distribuição de tarefas baseadas na aptidão dos recursos produzem melhores resultados que as tradicionais políticas baseadas em *Pull* e *Push*.

Esse trabalho abordou uma técnica oriunda da área de Análise de Ligações em Sistemas de Recuperação de Informações (SRI) para solucionar um problema da distribuição de tarefas em sistemas de workflow. Nesse contexto foi apresentada uma proposta para distribuição de tarefas que utiliza o algoritmo ***wf-hits***. Esse algoritmo é uma adaptação do algoritmo HITS para distribuição de tarefas em sistemas de *workflow* com a adição de pesos. Os pesos associados ao *wf-hits* são referentes a aptidão do recurso para executar uma dada tarefa.

Os experimentos realizados para corroborar esta proposta demonstraram que a distribuição de tarefas por meio da seleção induzida de recursos superou as técnicas investigadas neste trabalho. Esses experimentos foram realizados a partir de simulações da distribuição de tarefas em sistemas de *workflow* através do ambiente de simulação Lambari-SWTE [34]. Doze estratégias foram selecionadas para simulação e os resultados obtidos foram confrontados entre si.

No contexto das estratégias baseadas no algoritmo wf-hits, a estratégia *Selective-wf-hits* foi a que obteve o melhor compromisso entre qualidade e desempenho no processo de distribuição. Os resultados obtidos por essa estratégia foi confrontado

com os das principais estratégias destacadas nos trabalhos de Kumar *et, al.* [20] (*Selective*) e Veloso [32] (*Sel-Push*). Os resultados apresentados demonstram ganhos nos quesitos, qualidade e desempenho. A distribuição de tarefas através da estratégia *Selective-wf-hits* quando comparada com a estratégia *Sel-Push* destacada como a principal no trabalho de Veloso [32] obteve desempenho $8,38\%$ superior. Na comparação com a estratégia *Selective* de Kumar, *et, al* [20], os ganhos foram da ordem de 25%. Os índices de qualidade obtidos foram ligeiramente melhores (em média 1%) que o trabalho de Veloso [32] e não apresentaram diferenças significativas segundo o teste-t pareado. Esses resultados confirmam o melhor compromisso entre desempenho e qualidade da distribuição de tarefas para as estratégias *Selective-wf-hits*.

Outro ponto positivo deste trabalho é a estratégia de distribuição *Full-wf-hits* que obteve os menores tempos de execução quando comparada com todas as outras estratégias encontradas na literatura. Segundo o teste-t pareado a estratégia *Full-wf-hits* obteve resultados significativamente melhores que a melhor estratégia considerada pela literatura, a estratégia Pull. A *Full-wf-hits* foi 18% mais rápida em média que a *Pull* em relação ao tempo de execução.

## 6.1 Trabalhos futuros

Alguns pontos que podem estender este trabalho foram identificados ao longo do tempo. Estes pontos seguem como sugestões de trabalhos futuros.

- O algoritmo *wf-hits* classifica os recursos e as tarefas conforme os graus de autoridade e *hub*. São adotados como pesos para o cálculo dos *rankings* os valores de alocação absoluta que são computados a partir dos valores de adequação, urgência, conformidade e disponibilidade. No contexto deste trabalho, os valores de disponibilidade do recurso e urgência da tarefa são prefixados. Seria desejável que estes valores fossem ajustados em tempo de execução de acordo com informações oriundas da própria distribuição. Por exemplo, à medida que um recurso fosse se ocupando de atividades, que sua disponibilidade fosse reajustada, e ainda, que o valor de urgência da tarefa sofresse acréscimos com o passar do tempo.

- Variações da estratégia *Random-wf-hits* podem ser definidas. Na estratégia *Random-wf-hits* a ordem das tarefas é desprezada e elas são selecionadas aleatóriamente. Não foram realizados neste trabalho experimentos relacionados a seleção aleatória dos recursos. Sugere-se então o desenvolvimento de uma estratégia onde a seleção dos recursos é realizada aleatóriamente (desprezando a ordem definida pelo *wf-hits*), porém obedecendo a ordem definida para as tarefas. Uma versão onde realiza-se a seleção aleatória de ambos (recursos e tarefas) também pode ser implementada, porém neste caso, perde-se a característica "hits" desta versão.

- O algoritmo *wf-hits* é utilizado no contexto deste trabalho exclusivamente para a distribuição de tarefas. Outras classificações acontecem no procedimento de alocação de recursos às tarefas, são elas a ordenação global e a ordenação local (veja detalhes na sessão 3.4). Heurísticas como FIFO, SPT, SIRO entre outras são adotadas para os procedimnentos de ordenação. Sugere-se então, adaptar o algoritmo *wf-hits* para que este possa ser aplicado às etapas de ordenação global e local.

A distribuição de tarefas em sistemas de *workflow* ainda é uma área em franco desenvolvimento. As sugestões acima abrem espaço para que novas pesquisas possam ser realizadas e que a aplicação dos conceitos oriundos dos Sistemas de Recuperação de Informação especificamente na área de Análise de Ligações possam ser estendidos para a aplicação em outras áreas da Ciência da Computação.

## 6.2 Publicações

Algumas publicações foram realizadas a partir dos resultados obtidos com este trabalho de mestrado. A listagem dos artigos publicados é apresentada a seguir:

1. **Distribuição de Tarefas em Sistemas de Workflow por Meio da Seleção Induzida de Recursos**
   Simpósio Brasileiro de Sistemas Colaborativos - SBSC2006
   Novembro de 2006

2. **Uso de Simulações para a Análise de Alocações e Tempo de Processamento de Tarefas em Sistemas de Workflow**
   Semana da Computação da Fesurv – SECOMF05
   Novembro de 2005

# Referências Bibliográficas


[1] Bharat, K., and Henzinger, M. R. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1998), ACM Press, pp. 104–111.

[2] Borodin, A., Roberts, G. O., Rosenthal, J. S., and Tsaparas, P. Link analysis ranking: algorithms, theory, and experiments. *ACM Trans. Inter. Tech. 5*, 1 (2005), 231–297.

[3] Brin, S., and Page, L. The anatomy of a large-scale hypertextual Web search engine. In *Proc. of the 7th International World Wide Web Conference (WWW7)* (Brisbane, Australia, 1998), pp. 107–117.

[4] Brucker, P. *Scheduling Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[5] Calado, P., Ribeiro-Neto, B., Ziviani, N., Moura, E., and Silva, I. Local versus global link information in the web. *ACM Transactions on Information Systems 21*, 1 (january 2003), 42–63.

[6] Chakrabarti, S., Dom, B. E., Kumar, S. R., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, D., and Kleinberg, J. Mining the web's link structure. *Computer 32*, 8 (1999), 60–67.

[7] Cheng, E. C. An object-oriented organizational model to support dynamic role-based access control in electronic commerce. *Decision Support Systems 29*, 4 (2000), 357–369.

[8] DER AALST, W. V. Generic workflow models: How to handle dynamic change and capture management information? *coopis 00* (1999), 115.

[9] DER AALST, W. V., AND JABLONSKI, S. Dealing with workflow change: Identification of issues and solutions. *Computer Systems Science and Engineering 15* (2000), 267–267.

[10] DING, C., HE, X., HUSBANDS, P., ZHA, H., AND SIMON, H. D. Pagerank, hits and a unified framework for link analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2002), ACM Press, pp. 353–354.

[11] DRORI, O. Algorithm for documents ranking: idea and simulation results. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering* (New York, NY, USA, 2002), ACM Press, pp. 99–102.

[12] ELLIS, C., KEDDARA, K., AND ROZENBERG, G. Dynamic change within workflow systems. In *Proceedings of conference on Organizational computing systems* (New York, NY, USA, 1995), COCS '95, ACM, pp. 10–21.

[13] GOVERNATORI, G., ROTOLO, A., AND SADIQ, S. A model of dynamic resource allocation in workflow systems. In *XV Australasian Database Conference* (Dunedin, Nova Zelândia, 2004), vol. 27.

[14] GRAZIANO, A. M., AND RAULIN, M. L. *Research Methods: a process of inquiry*, 4th ed. Allyn and Bacon, Inc., 1999.

[15] HENZINGER, M. Hyperlink analysis on the world wide web. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia* (New York, NY, USA, 2005), ACM Press, pp. 1–3.

[16] HENZINGER, M. R. Hyperlink analysis for the web. *IEEE Internet Computing 5*, 1 (2001), 45–50.

[17] INC., G. Google search engine. http://www.google.com, 2007.

[18] KLEINBERG, J. M. Authoritative sources in a hyperlinked environment. In *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete*

*algorithms* (Philadelphia, PA, USA, 1998), Society for Industrial and Applied Mathematics, pp. 668–677.

[19] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *J. ACM-SIAM (Symposium on Discrete Algorithms) 46*, 5 (1999), 604–632.

[20] Kumar, A., van der Aalst, W. M., and Verbeek, E. M. Dynamic work distribution in workflow management systems: How to balance quality and performance? *Journal of Management Information Systems 18*, 3 (2001), 157–193.

[21] Li, L., Shang, Y., and Zhang, W. Improvement of hits-based algorithms on web documents. In *WWW '02: Proceedings of the 11th international conference on World Wide Web* (New York, NY, USA, 2002), ACM Press, pp. 527–535.

[22] mendes de Araujo, R., and da Silva Borges, M. R. Sistemas de workflow. XX Jornada de Atualização em Informática, 2001. Fortaleza, Ceará, Brasil.

[23] Momotko, M., and Subieta, K. Dynamic changes in workflow participant assignment. In *ADBIS Research Communications* (Bratislava, Slovakia, setembro 2002), vol. 2, Slovak University of Technology, Bratislava, pp. 175–184.

[24] Overture Services, I. Altavista search engine. http://www.altavista.com, 2007.

[25] Silva, I., de Souza, J. N., Moura, R. F. L., and Ribeiro-Neto, B. A. Informação de links no modelo vetorial usando uma estrutura funcional. In *SBBD* (2003), pp. 170–184.

[26] Silva, I., Ribeiro-Neto, B., Calado, P., Moura, E., and Ziviani, N. Link-based and content-based evidential information in a belief network model. In *Proc of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval* (Athens, Greece, Julho 2000). Best student paper.

[27] Silva, R. S., Macêdo, A., Silva, I. R., and de Sousa, J. N. Distribuição de tarefas em sistemas de workflow por meio de seleção induzida de recursos.

In *Anais do III Simpósio Brasileiro de Sistemas Colaborativos-SBSC2006* (Natal/RN, November 2006), S. S. B. de Computação, Ed., vol. 3, pp. 107–117.

[28] Stohr, E. A., and Zhao, J. L. Workflow automation: Overview and research issues. *Inform. Systems Frontiers 3* (2001), 281–296.

[29] Tramontina, G. B. Análise de problemas de escalonamento de processos em workflow. Master's thesis, Instituto de Computação, Universidade de Campinas (UNICAMP), Campinas, SP, Brazil, 2004.

[30] Tramontina, G. B., Wainer, J., and Ellis, C. Applying scheduling techniques to minimize the number of late jobs in workflow systems. In *Proceedings of the 2004 ACM Symposium on Applied Computing* (New York, NY, USA, 2004), A. Press, Ed., pp. 1396–1403.

[31] van der Aalst, W. M. P., and van Hee, K. M. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[32] Veloso, R. R. Distribuição de tarefas em sistemas de workflow baseada na aptidão dos recursos. Master's thesis, Faculdade de Computação, Universidade Federal de Uberlândia (UFU), Uberlândia, MG, Brazil, 2006.

[33] Veloso, R. R., and Macêdo, A. Distribuição de tarefas em sistemas de workflow baseado na aptidão dos recursos. In *Anais do Workshop Brasileiro de Tecnologias de Colaboração (WCSCW)* (Juiz de Fora, Minas Gerais, Brasil, novembro 2005), vol. 2. Mídia digital.

[34] Veloso, R. R., and Macedo, A. Lambari: Simple workflow test environment. Tech. rep., Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil, 2005.

[35] Wang, K., and Su, M.-Y. T. Item selection by "hub-authority" profit ranking. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2002), ACM Press, pp. 652–657.

[36] WfMC. Interface 1: Process definition interchange process model. Tech. Rep. WfMC TC-1016-P, Workflow Management Coalition, Hampshire, Reino Unido, Fevereiro 1999.

[37] WfMC. Terminology and glossary. Tech. Rep. WfMC-TC-1011, Workflow Management Coalition, Hampshire, Reino Unido, Fevereiro 1999.

[38] Xing, W., and Ghorbani, A. Weighted pagerank algorithm. In *CNSR '04: Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 305–314.

[39] Xue, G.-R., Zeng, H.-J., Chen, Z., Ma, W.-Y., Zhang, H.-J., and Lu, C.-J. Implicit link analysis for small web search. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (New York, NY, USA, 2003), ACM Press, pp. 56–63.

[40] Zeng, D. D., and Zhao, J. L. Effective role resolution in workflow management. *INFORMS journal on computing 17*, 3 (2005), 374–387.

[41] zur Muehlen, M. Resource modeling in workflow applications. In *Proceedings of the 1999 Workflow Management Conference* (novembro 1999), pp. 137–153.

# Apêndice A

# Arquivo XPDL utilizado pelo Simulador Lambari

O simulador e motor de *workflow* Lambari SWTE utiliza um modelo de XPDL (*XML Process Definition Language*) simplificado. Este modelo contém as definições das tarefas (apresentadas na cláusula *<tasks> ... </tasks>*), das pré-condições (cláusulas *precond = " "*), das pós-condições (cláusulas *poscond = " "*), do tempo de execução das tarefas (cláusulas *time = " "*), dos papéis associados (cláusulas *role = " "*) e das transições do processo de negócio (cláusulas *<transitions> ... </transitions>*).

## A.1 Exemplo de XPDL


Figura A.1: Processo de negócio utilizado nos experimentos.

Nesta seção é apresentado um exemplo de arquivo XPDL. O XPDL apresentado é o mesmo utilizado nos experimentos deste trabalho. Este arquivo descreve o processo de negócio da figura A.1. Este processo de negócio é composto por três tarefas: *Tarefa A*, *Tarefa B* e *Tarefa C* que estão associadas a três papéis: *Papel 1*, *Papel 2* e *Papel 3*. A ordem das tarefas é especificada pela cláusula *transitions*. De

acordo com o processo de negócio apresentado a *Tarefa A* é a tarefa inicial, seguida pela *Tarefa B* e a *Tarefa C* é a que finaliza o fluxo. Veja a definição do processo de negócio em XPDL no texto listado a seguir:

```
 <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lxpdl SYSTEM "lxpdl.dtd"> <lxpdl>
    <tasks>
        <task name="Tarefa A" precond="NULL" poscond="" time="10">
            <role name="Papel 1"/>
        </task>
        <task name="Tarefa B" precond="Tarefa A" poscond="" time="10">
            <role name="Papel 2"/>
        </task>
        <task name="Tarefa C" precond="Tarefa B" poscond="" time="10">
            <role name="Papel 3"/>
        </task>
        <start name="1"/>
        <end name="3"/>
    </tasks>
    <transitions>
        <transition source="Tarefa A" destination="Tarefa B"/>
        <transition source="Tarefa B" destination="Tarefa C"/>
    </transitions>
</lxpdl>
```

## A.2 Gramática DTD para XPDL

Apresenta-se nesta seção a gramática DTD que define a sintaxe do XPDL utilizado pelo simulador. O código dessa gramática é listado a seguir.

```
<?xml version='1.0' encoding='UTF-8'?>
<!--
    TODO define vocabulary indentification
    PUBLIC ID: -//Facom, Federal University of Uberlandia //DTD //EN
```

```
    SYSTEM ID: lxpdl.dtd
-->
<!ELEMENT transition EMPTY>
<!ATTLIST transition
    destination CDATA #IMPLIED
    source CDATA #IMPLIED
  >
<!ELEMENT transitions (transition)+>
<!ELEMENT role EMPTY>
<!ATTLIST role
    name CDATA #IMPLIED
  >
<!ELEMENT task (role)>
<!ATTLIST task
    time CDATA #IMPLIED
    poscond CDATA #IMPLIED
    precond CDATA #IMPLIED
    name CDATA #IMPLIED
  >
<!ELEMENT start EMPTY>
<!ATTLIST start
    name CDATA #IMPLIED
  >
<!ELEMENT end EMPTY>
<!ATTLIST end
    name CDATA #IMPLIED
  >
<!ELEMENT tasks (end|start|task)+>
<!ELEMENT lxpdl (transitions|tasks)+>
```