HENRIQUE DUARTE LIMA

MAGNETIC SWARMLINDA: DISTRIBUIÇÃO DE TUPLAS AFETADA POR CAMPOS MAGNÉTICOS

Dissertação de Mestrado apresentado ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Curitiba
2016

HENRIQUE DUARTE LIMA

MAGNETIC SWARMLINDA: DISTRIBUIÇÃO DE TUPLAS AFETADA POR CAMPOS MAGNÉTICOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Ph.D. Luiz Augusto de Paula Lima Junior

Curitiba
2016

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

ATA DE SESSÃO PÙBLICA

DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 09/2016

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA – PPGIa
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ - PUCPR

Em sessão pública realizada às 14h00 de 19 de Setembro de 2016, no Auditório Guglielmo Marconi, ocorreu a defesa da dissertação de mestrado intitulada **"Magnetic Swarmlinda: Distribuição de Tuplas Afetada por Campos Magnéticos"** apresentada pelo aluno **Henrique Duarte Lima**, como requisito parcial para a obtenção do título de **Mestre em Informática**, na área de concentração **Ciência da Computação,** perante a banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Augusto de Paula Lima Junior (Orientador)- PUCPR

Prof. Dr. Alcides Calsavara – PUCPR

Prof. Dr. Henri Eberspacher - PUCPR

Prof. Dr. Elias Procópio Duarte Junior – UFPR

Após a apresentação da dissertação pelo aluno e correspondente arguição, a banca examinadora emitiu o seguinte parecer sobre a tese:

Membros	Parecer
Prof. Dr. Luiz Augusto de P.Lima Junior	(X) Aprovada ( ) Reprovada
Prof. Dr. Alcides Calsavara	(X) Aprovada ( ) Reprovada
Prof. Dr. Henri Eberspacher	(X) Aprovada ( ) Reprovada
Prof. Dr. Elias Procópio Duarte Junior	(X) Aprovada ( ) Reprovada

Portanto, conforme as normas regimentais do PPGIa e da PUCPR, a dissertação foi considerada:

(X) **APROVADA**
(aprovação condicionada ao atendimento integral das correções e melhorias recomendadas pela banca examinadora, conforme anexo, dentro do prazo regimental)

( ) **REPROVADA**

E, para constar, lavrou-se a presente ata que vai assinada por todos os membros da banca examinadora. Curitiba, 19 de Setembro de 2016.

Prof. Dr. Luiz Augusto de Paula Lima Junior

Prof. Dr. Alcides Calsavara

Prof. Dr. Henri Eberspacher

Prof. Dr. Elias Procópio Duarte Junior

ANEXO À ATA DE SESSÃO PÙBLICA
CORREÇÕES E MELHORIAS RECOMENDAS PELA BANCA EXAMINADORA

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 Fax.: (41) 3271 2121 www.ppgia.pucpr.br

Sumário

Resumo

Aplicações distribuídas – especialmente aquelas que requerem grandes quantidades de recursos – são intrinsecamente complexas em termos de comunicação. Portanto, a redução da interdependência de componentes de comunicação e a simplificação de suas interações em uma aplicação distribuída é uma preocupação sempre presente. Desacoplamentos temporal e espacial de componentes do sistema podem ser providos por modelos de memória distribuída compartilhada como aqueles introduzidos por Linda e o chamado *Espaço de tuplas*. Espaços de tuplas simplificam a comunicação entre processos e reduzem o impacto da substituição e/ou inclusão de novos processos em um sistema distribuído. No entanto, conforme o número de tuplas aumenta, o desempenho das operações de recuperação de tuplas pode ser particularmente afetada de forma negativa e pode ser uma das razões que explicam porquê filas (apesar de suas deficiências) são geralmente adotadas ao invés de espaços de tuplas em sistemas distribuídos desacoplados. Técnicas "bioinspiradas" baseadas em inteligência de enxames foram então propostas na literatura a fim de melhorar a eficiência das operações de tuplas. No entanto, estas abordagens produzem uma importante degradação do desempenho quando o número de tuplas "similares" é relativamente alto. Este documento mostra que alguns gargalos podem ser evitados através da aplicação de "campos magnéticos virtuais" no comportamento de enxame de modo a reorganizar em *clusters* expansíveis que fornecem balanceamento de carga entre os nós de suporte. A técnica proposta apresenta um desempenho melhor que a abordagem de enxame simples em cenários de alto desempenho, como os resultados da simulação indicam.

Palavras-chave: Linda; Espaço de tuplas; Inteligência de enxame; Campos magnéticos virtuais.

Abstract

Distributed applications – especially those that require large amounts of resources – are intrinsically complex in terms of communication. Therefore, the reduction of the interdependence of communicating components and the simplification of their interactions in a distributed application is an ever-present concern. Such a spatial and temporal uncoupling of system components can be provided by shared distributed memory models like those introduced by Linda and the so-called *Tuple Spaces*. Tuple spaces simplify inter-process communication and reduce the impact of the replacement of components and/or the inclusion of new ones into a distributed system. However, as the number of tuples increases, the performance of retrieval operations may be particularly affected in a bad way and that may be one of the reasons that explain why queues (despite their shortcomings) are generally preferred over tuple spaces in loose-coupled distributed systems. "Bioinspired" techniques based on swarm intelligence were then proposed in the literature in order to improve the efficiency of tuple operations. Nevertheless, these approaches produce an important performance degradation when the number of "similar" tuples is relatively high. This paper shows how some bottlenecks can be avoided by applying "virtual magnetic fields" to the swarm behavior so that tuples will be rearranged in expandable clusters that provide load balancing among supporting nodes. The proposed technique performs better than the simple swarm approach in high performance environments, as the simulation results indicate.

Keywords: Linda; Tuple space; Swarm intelligence; Virtual magnetic fields.

Capítulo 1

Introdução

O modelo de comunicação desenvolvido em (GELERNTER; BERNSTEIN, 1982) inspirou o desenvolvimento de muitas pesquisas envolvendo espaços de tuplas. Neste modelo, aplicações distribuídas podem ser construídas com um elevado nível de abstração na comunicação. A complexidade na comunicação passa a ser uma responsabilidade do espaço de tuplas, simplificando o desenvolvimento de sistemas distribuídos. Os desacoplamentos no espaço e tempo, propiciados pelo modelo, permitem reduzir o grau de dependência entre os módulos que compõem um sistema. Assim, as restrições de execução e manutenção dos módulos são simplificadas. No entanto, a capacidade de escalar uma aplicação passa a ser limitada pela escalabilidade relativa ao espaço de tuplas. A escalabilidade é um desafio que precisa ser enfrentado para viabilizar a oferta de espaços de tuplas em ambientes de nuvem, como indica (HARI, 2012), permitindo que aplicações científicas, comerciais e de telecomunicações possam tirar proveito de uma infraestrutura elástica.

Segundo (NAVLAKHA; BAR-JOSEPH, 2014), trabalhos recentes na computação distribuída e em sistemas biológicos têm analisado a capacidade de modelos de comunicação extremamente leves para solucionar importantes problemas na computação. A maioria desses modelos bioinspirados são naturalmente distribuídos, sendo habilitados para efetuar decisões sem a necessidade de um controle centralizado e de uma visão global do estado do sistema. De maneira geral, estas características são desejáveis a muitas aplicações distribuídas, como espaços de tuplas distribuídos. Para (MENEZES; WOOD, 2006), a escalabilidade de espaços de tuplas pode ser melhorada através da incorporação de técnicas inspiradas na biologia. Neste segmento, há abordagens como (MENEZES; TOLKSDORF, 2003) que empregam agentes inspirados no comportamento das formigas para implementar as funcionalidades do espaço de tuplas.

1.1 Problema

Embora a maioria dos espaços de tuplas permitam a construção de sistemas distribuídos, estes não necessariamente possuem uma implementação distribuída. A necessidade de construir aplicações distribuídas escaláveis exige implementações de espaço de tuplas preparadas para usufruir de recursos distribuídos. Neste contexto, é necessário que essas implementações sejam habilitadas para distribuir a carga de maneira eficiente, evitando degradações de desempenho. No entanto, muitas dessas implementações permitem uma concentração da carga no espaço de tuplas (JIANG et al., 2006; ATKINSON, 2008), comprometendo o desempenho. No entanto, simplesmente distribuir as tuplas entre os diversos servidores que compõem um espaço de tuplas não garante a eficiência, pois dificulta a recuperação dessas tuplas. Desta forma, a eventual melhoria na distribuição das tuplas provoca um consumo excessivo de processamento na recuperação de tuplas, inviabilizando a abordagem.

1.2 Motivação

A distribuição de tuplas consiste num importante desafio para escalar espaços de tuplas. Contudo, muitas abordagens empregam uma distribuição envolvendo funções de espalhamento que permitem a concentração de carga em determinados pontos, provocando uma degradação do desempenho. A motivação deste trabalho consiste na possibilidade de desenvolver uma distribuição mais eficiente de tuplas, permitindo um aperfeiçoamento do desempenho de espaços de tuplas distribuídos.

1.3 Hipótese

A hipótese no qual esse trabalho se baseia é que a aplicação do conceito de campos magnéticos virtuais, para afetar o comportamento de agentes bioinspirados responsáveis pela distribuição de tuplas, pode aprimorar o desempenho de um espaço de tuplas distribuído. O campo magnético virtual produzido por um nó com elevada carga é intenso, provocando uma considerável magnetização dos nós próximos. Nestes, os agentes responsáveis pela distribuição das tuplas acabam influenciados pela força do campo magnético virtual e tendem a depositar as tuplas em nós próximos a um nó com elevada concentração de tuplas similares. Esta abordagem nos permite criar *clusters* de nós com tuplas similares, evitando a degradação do desempenho provocada pela concentração excessiva de tuplas em poucos nós.

1.4 Organização

O restante deste documento está organizado da seguinte forma. No Capítulo 2, será apresentada uma síntese dos conceitos e das principais implementações de espaços de tuplas distribuídos, assim como a essência de algoritmos bioinspirados e duas abordagens de espaços de tuplas inspiradas no comportamento das formigas. Além disso, serão apresentados os conceitos de campos magnéticos aplicados à computação e os efeitos destes no comportamento das formigas. No Capítulo 3, será apresentada a definição da abordagem proposta, bem como as equações que fundamentam o modelo magnético concebido. Em seguida, no Capítulo 4, serão apresentados os cenários para avaliação do desempenho e os resultados obtidos serão analisados. Por fim, será apresentada a conclusão sobre a abordagem introduzida no presente trabalho.

Capítulo 2

Estado da Arte

Este capítulo tem por objetivo apresentar as características de diferentes espaços de tuplas, assim como apresentar conceitos de inteligência de enxames e campos magnéticos, para respaldar as decisões pertinentes a proposta. Na primeira parte deste capítulo será apresentado um modelo de comunicação empregado pela linguagem Linda para manipulação de espaços de tuplas, assim como as principais características de desacoplamento propiciadas pelo modelo. Adicionalmente, serão apresentadas quatro plataformas inspiradas neste modelo de comunicação, bem como uma comparação de características das plataformas expostas será efetuada. Em seguida, serão apresentadas inspirações provenientes da biologia que são aplicadas na computação. Uma plataforma de espaço de tuplas, denominada de SwarmLinda, que emprega conceitos de inteligência de enxames será exposta conjuntamente às inspirações da biologia. Na sequência, uma abordagem, incluindo dois algoritmos, de balanceamento de carga inspirados em campos magnéticos, assim como as consequências da exposição de formigas a diferentes formas de campo magnético, serão apresentados. Por fim, serão apresentadas as considerações finais envolvendo todos os assuntos abordados.

2.1 Espaços de tuplas

O paradigma de espaço de tuplas é uma abstração de memória compartilha distribuída que permite a comunicação entre processos distribuídos através da inserção e recuperação de informações estruturadas. Nesta seção serão apresentados o modelo de comunicação de Linda (GELERNTER; BERNSTEIN, 1982), que introduziu as operações mais empregadas em espaços de tuplas, e as principais plataformas distribuídas que implementam o modelo.

2.1.1 Linda

A linguagem Linda é composta por um reduzido conjunto de operações que permitem a um processo manipular o estado de um espaço de tuplas. Por meio dessas operações é possível construir uma aplicação distribuída que dispensa o emprego de *sockets* ou *remote procedure call (RPC)*. Neste cenário, as diversas instâncias de uma aplicação comunicam-se de maneira indireta, utilizando-se do espaço de tuplas como um meio para propagar a troca de informações. Essas informações são armazenadas em objetos, nomeados de tuplas, que podem conter qualquer tipo e quantidade de dados que a aplicação necessite.

As interações dos processos com o espaço de tuplas são efetuadas por meio de uma pequena quantidade de operações, presentes no modelo de comunicação empregado em Linda. As principais operações de Linda são: i) `out` - inserir uma tupla; ii) `in` - retirar uma tupla; iii) `rd` - ler uma tupla; De acordo com (ROWSTRON; WOOD, 1997), adicionalmente a este conjunto, existem as operações `inp` e `rdp`, que são versões não-bloqueantes de `in` e `rd`, porém muitas das implementações de espaço de tuplas não suportam estas operações.

Há também a operação `eval`, que permite inserir uma tupla composta por uma informação que ainda não foi computada. Nessa operação, o espaço de tuplas é responsável por instanciar processos para executarem os procedimentos definidos na própria tupla. Os resultados desses procedimentos são armazenados na tupla, que apenas então torna-se disponível no espaço de tuplas para recuperação por qualquer processo. De acordo com (ATKINSON, 2010b), embora incluída no modelo original de Linda, a operação `eval` não está presente na maioria das demais implementações, em favor de mecanismos de criação de processos mais tradicionais. Em espaços de tuplas que não disponibilizam uma implementação da operação `eval`, é possível atingir um resultado equivalente com a utilização da primitiva `fork`, presente na maioria dos sistemas operacionais, em conjunto com a operação `out`.

As operações `in` e `rd` utilizam-se de um elemento denominado de *template*, que permite à aplicação especificar o formato da tupla que é requerida, assim como impor restrições em relação ao conteúdo da tupla. Sendo que uma tupla deve atender as seguintes condições para ser considerada compatível com um determinado *template*: i) a quantidade de campos do *template* deve ser igual a da tupla; ii) os valores especificados no *template* devem ser iguais aos da tupla. Os campos nulos do *template* comportam-se como um *curinga*, ou seja, não é necessário que a tupla possua um valor especifico no campo correspondente para ser considerada compatível. Na maioria dos cenários, os campos especificados com valor nulo são os que contém as informações requeridas pela

aplicação.

Quando um processo invoca a operação `in` ou `rd` para um determinado *template*, informado por parâmetro, que não possui nenhuma tupla compatível disponível no espaço de tuplas, este processo aguarda bloqueado a inserção de uma tupla compatível. Desta forma, a operação retorna apenas quando existir pelo menos uma tupla que atenda as restrições impostas pela aplicação. Na perspectiva da aplicação, é impossível concluir se existia uma tupla no momento em que a operação foi invocada, pois o bloqueio ocorre de maneira transparente para a aplicação. A única percepção possível, por parte da aplicação, é de que a operação consumiu um tempo maior quando comparada as execuções passadas, porém a aplicação é incapaz de inferir se o atraso ocorreu devido a ausência de uma tupla compatível, ou se ocorreu devido a uma flutuação na carga do espaço de tuplas.

Além da restrição de compatibilidade com o *template*, fornecido pela aplicação, não existe nenhum outro critério durante a seleção de uma tupla. Conforme explica (ROWSTRON; WOOD, 1997), a escolha é não-determinística em cenários onde existem muitas tuplas compatíveis com determinado *template*, assim como é não-determinística a escolha de um processo quando existem muitos processos competindo por uma única tupla. Esta característica permite que espaços de tuplas distribuídos efetuem parte de suas operações em âmbito local, ou em uma determinada porção do espaço de tuplas, propiciando implementações sofisticadas que minimizem a quantidade de comunicações entre os nós que compõem o espaço de tuplas.

No desenvolvimento de aplicações distribuídas, o espaço de tuplas permite eliminar a necessidade de mecanismos que possibilitam a uma instância conhecer a localização das instâncias com as quais a primeira deseja comunicar-se. Além disso, é desnecessário que a aplicação mantenha qualquer informação a respeito dos demais processos, permitindo que cada processo preocupe-se apenas com as informações requeridas para suas operações, assim como a decisão de quais resultados devem ser disponibilizados para os demais processos.

A característica de Linda independer de um endereçamento de processos é denominada de desacoplamento espacial. Esta característica permite que qualquer processo, independente da sua localização, disponibilize informações no espaço de tuplas. Analogamente, qualquer processo pode recuperar informações do espaço de tuplas. Desta forma, há uma simplificação na comunicação que permite abstrair completamente a origem dos dados necessários para a execução de um procedimento e o destino dos resultados computados.

A simplificação da comunicação permite que o desenvolvedor concentre seus esforços na implementação da lógica da aplicação, em oposição a despender uma significativa

parcela de tempo no mecanismo de interação entre os processos. Além disso, a disponibilização de uma informação é independente da quantidade de processos que possuirão acesso a informação. Neste modelo, é possível deslocar um processo de local sem provocar nenhum impacto nos demais, sendo necessário apenas que o processo reconecte-se ao espaço de tuplas.

Em aplicações que utilizam-se de *sockets* ou *remote procedure call* há a necessidade do processo servidor estar previamente em execução quando o processo cliente iniciar a comunicação. Esta restrição não é encontrada em aplicações construídas com o modelo de comunicação de Linda, pois não há uma comunicação direta entre dois processos. É possível que um processo disponibilize tuplas ainda que inexista demais processos em execução. As tuplas podem ser recuperadas por qualquer novo processo, ainda que este seja instanciado após a finalização do processo que disponibilizou a tupla. Essa característica, denominada de desacoplamento temporal, permite reduzir a quantidade de restrições para a execução de um processo, diminuindo a complexidade de uma aplicação distribuída.

As seções seguintes descrevem as principais plataformas distribuídas que implementam o modelo de comunicação de Linda.

2.1.2 Lime

O Lime, proposto por (PICCO; MURPHY; ROMAN, 1999), é uma implementação de espaço de tuplas inspirada em Linda que permite a comunicação entre agentes que possuem a capacidade de deslocarem-se entre dispositivos móveis heterogêneos. O objetivo é permitir o rápido desenvolvimento de aplicações móveis através do modelo de comunicação de Linda, abstraindo e gerenciado as mudanças de conectividade provocadas pela movimentação dos dispositivos. Desta forma, no desenvolvimento dos agentes é possível concentrar os esforços na mobilidade entre os *hosts*, assim como na lógica da aplicação, evitando a complexa tarefa de gerenciamento da comunicação. Além disso, o Lime também suporta a execução de agentes que permanecem contidos permanentemente em um único *host*. Para cada agente, a única percepção do ambiente é que o conjunto de tuplas disponíveis modifica-se, de maneira transparente, conforme ocorrem mudanças de conectividade entre o seu *host* e os *hosts* dos demais agentes. É possível também que o volume de tuplas altere-se em função da movimentação de agentes entre os *hosts*.

Há três níveis de espaço de tuplas disponíveis no Lime, sendo que cada agente pode possuir múltiplos espaços de tuplas. O espaço de tupla para uso exclusivo de um

agente é denominado de *Interface tuple space (ITS)*, onde cada *ITS* possui um nome associado. No momento que um agente desloca-se para um novo dispositivo, este carrega consigo os seus espaços de tuplas. É possível conectar os diversos espaços de tuplas pertencentes aos agentes que estão compartilhando um dispositivo móvel, desta forma os agentes enxergam determinado espaço de tuplas como uma combinação de todos os que compartilham um mesmo nome, sendo esta combinação denominada de *Host tuple space*. Quando um desses agentes move-se para um outro dispositivo móvel, o espaço de tuplas deste deixa de compor o *Host tuple space* do primeiro dispositivo, passando a integrar o *Host tuple space* do novo hospedeiro. Analogamente, é possível combinar os espaços de tuplas dos agentes hospedados em dispositivos que estão conectados, sendo a combinação denominada de *Federated tuple space*. Além do critério do nome do espaço de tuplas, essas diferentes formas de combinações ocorrem respeitando o formato de cada espaço de tuplas, desta maneira, um agente terá seu espaço de tuplas combinado com outros apenas se desejar. Além disso, é possível que uma agente mantenha, de acordo com suas necessidades, espaços de tuplas com diferentes abrangências.

A implementação de (PICCO; MURPHY; ROMAN, 1999) disponibiliza as operações principais de Linda que permitem abstrair a localização dos demais agentes. No entanto, Lime inclui também operações similares que permitem especificar o local onde a operação será executada, permitindo atender necessidades mais especificas de uma aplicação, assim como permitir uma melhora no desempenho. Além dessas operações, há a operação denominada de *reactsTo* que permite especificar um *template* e um procedimento que deve ser executado quando uma tupla compatível for encontrada. Esta operação retorna ainda que nenhuma tupla for encontrada no exato momento, porém possui como restrição, devido a questões de escala, a especificação de um *host* ou agente onde acontecerá o monitoramento de uma tupla compatível. A execução do procedimento registrado ocorre de maneira síncrona perante os demais procedimentos registrados, ou seja, a execução é serializada quando dois ou mais procedimentos estão aptos a serem executados. Esta operação possui um importante papel quando um agente necessita aguardar múltiplos formatos de tuplas, o que é possível apenas de atingir-se com as operações principais de Linda utilizando-se de diferentes fluxos de execução para executar múltiplos *in's*, provocando um custo adicional no desempenho e um aumento na complexidade da aplicação. Para uma semântica idêntica ao *reactsTo*, porém sem a restrição de especificar um agente ou *host*, está disponível a operação *upon* que não possui garantias de serialização das operações, o que demanda mais atenção na sua utilização.

2.1.3 B-Linda

O B-Linda, proposto por (GIBAUD; THOMIN, 2002), é uma implementação de espaço de tuplas inspirada em Linda que destina-se a aplicações de ambientes heterogêneos. Neste contexto, a ampla diversidade de processos estimula a utilização de uma ampla variedade de formatos de tuplas, aumentando a probabilidade de informações com objetivos diferentes possuírem formatos idênticos. Esse conflito permite que um processo recupere e processe uma tupla destinada a um processo que possui um propósito diferente, provocando uma inconsistência na aplicação. A recuperação indevida de uma tupla, denominada de *false matching*, produz como consequência uma saída equivocada do processo que recebeu a tupla, além de impedir que a tupla seja processada adequadamente quando a recuperação da tupla ocorreu de maneira destrutiva, ou seja, a tupla foi removida do espaço de tuplas.

No modelo empregado por Linda, o critério de compatibilidade entre os campos de uma tupla e de um *template*, considerando seus valores e tipos, é suficiente para a distinção e recuperação das tuplas para os processos adequados. A responsabilidade de evitar o *false matching* pertence ao desenvolvedor da aplicação, que deve escolher formatos que não colidam entre si. Em cenários onde há uma quantidade reduzida de formatos é plausível manipular a ordem dos campos para evitar colisões de formatos. No entanto, em ambientes complexos pode ser necessária a inclusão de mais um campo na tupla para identificar o propósito da tupla, permitindo que os processos possam recuperar apenas as tuplas que sejam compatíveis com o seus propósitos. Para (GIBAUD; THOMIN, 2002) esta solução é inadequada, pois permite a definição de um propósito ambíguo, ou seja, que ainda é suscetível ao problema de *false matching*.

Com o objetivo de solucionar o problema de *false matching*, o B-Linda emprega um elemento, denominado de *b-type*, que passa a fazer parte do critério de compatibilidade entre uma tupla e um *template*, formando um particionamento do espaço de tuplas que não necessita ser explicitado. Sendo o b-type composto por: i) *St* - que representa os diversos tipos dos campos que compõem a estrutura da tupla; ii) *Se* - que representa a semântica associada a tupla; iii) *Sc* - que representa o escopo da tupla, ou seja, as informações armazenadas. Este elemento permite especificar, por exemplo, que os processos *P1* e *P2* comunicam-se empregando um determinado formato de tupla. Desta forma, ainda que os demais processos utilizem-se de tuplas com um formato idêntico, não existe a possibilidade de impactos na comunicação entre os processos *P1* e *P2*, pois a semântica de utilização da tupla é diferente. Para suportar este modelo, todas as operações disponibilizadas pelo B-Linda (`in`, `rd`, `out` e `eval`) possuem um parâmetro adicional *T*, que define o *(St,Se)*.

2.1.4 Dtuples

O Dtuples, proposto por (JIANG et al., 2006), é uma implementação de espaço de tuplas inspirada em Linda que é construído sobre uma *distributed hash table* que permite a distribuição das tuplas em ambientes *peer-to-peer*. O objetivo é simplificar a cooperação e coordenação de tarefas entre agentes em ambientes distribuídos. Além das principais operações presentes em Linda, o Dtuples disponibiliza versões assíncronas dessas operações e uma operação adicional denominada de *copy-collect*. Há também uma restrição imposta pelo Dtuples, no qual o elemento da posição inicial da tupla deve ser uma *string* com o nome da tupla, sendo que este valor não necessita ser único. Esta restrição deve-se ao fato da implementação utilizar o *hash* deste primeiro elemento da tupla para determinar o local de armazenamento da tupla. Na recuperação de uma tupla também é necessário informar este nome no primeiro campo do *template*, para permitir que o espaço de tuplas saiba em qual local deve ser executado o processo de busca de uma tupla compatível.

No DTuples existe uma divisão do espaço de tuplas em dois níveis, sendo que no nível denominado de *public tuple space*, o acesso é permitido a todos os nós que compõem o espaço de tuplas. Este nível comporta-se como uma implementação convencional de espaço de tuplas. No nível denominado de *subject tuple space* podem existir múltiplas instâncias de espaços de tuplas, onde cada uma possui um *subject name* único que identifica o espaço de tuplas. Estas instâncias são criadas pelos agentes que utilizam o espaço de tuplas, sendo combinados os espaços de tuplas que compartilham um *subject name*. As interações entre os agentes podem ocorrer nos dois níveis do espaço de tuplas, porém algumas operações possuem comportamentos distintos dependendo do nível no qual estas são executadas. A operação `out` pode ser executada para inserir uma tupla no *public tuple space* ou no *subject tuple space*, de acordo com a necessidade da aplicação. As operações `rd` e `in` podem ser executadas nos dois níveis, porém apresentam diferenças no resultado da execução de acordo com o nível. Quando estas operações são executadas sobre um *subject tuple space*, estas podem recuperar tuplas deste espaço específico ou do *public tuple space*. No entanto, quando executadas sobre o *public tuple space*, estas operações ficam restritas ao conjunto de tuplas disponíveis neste espaço de tuplas.

Além das operações principais de Linda, o Dtuple dispõe da operação *copy-collect*, que permite copiar todas as tuplas de um espaço de tuplas que são compatíveis com determinado *template* para um outro espaço de tuplas. Esta operação não pode ser substituída pela execução de muitas operações de `rd`, pois o modelo de comunicação de Linda não garante que uma sequência de execuções da operação `rd` retornará apenas tuplas distintas. Por exemplo, é possível que o espaço de tuplas retorne tuplas idênticas para

a execução de duas operações `rd` para um determinado *template*, sendo que a aplicação é incapaz de distinguir se os retornos são idênticos devido a existência de duas tuplas idênticas no espaço de tupla, ou se os dois retornos são referentes a uma única tupla.

2.1.5 Tupleware

O Tupleware, proposto por (ATKINSON, 2008, 2010a, 2010b), é uma *middleware* inspirado em Linda que implementa um espaço de tuplas que dispõe de um algoritmo descentralizado para recuperação de tuplas. O objetivo é prover um ambiente escalável para a execução de aplicações numéricas e científicas que possuem uma elevada demanda de poder computacional. O Tupleware exige que o desenvolvedor implemente os componentes *Master* e *Worker*, que extrapolam os conceitos relacionados a espaço de tuplas. O processo *Master* é responsável por segmentar uma tarefa computacional e distribuir aos *Workers* para execução, além de combinar os resultados disponibilizados pelos *Workers*. A comunicação entre todos os processos envolvidos ocorre através do espaço de tuplas, sendo que cada processo dispõe de uma instância que compõe o espaço de tuplas.

As tuplas inseridas por um processo são armazenadas em uma instância local que compõe o espaço de tuplas. Quando um processo requisita uma determinada tupla, o espaço de tuplas verifica a existência de uma tupla compatível com o *template*, fornecido pela aplicação, na porção local do espaço de tuplas. Em execuções onde a tupla não está disponível localmente, é efetuada uma busca nos demais servidores que compõem o espaço de tuplas, de forma transparente a aplicação. Para cada processo membro do espaço de tuplas é mantido um vetor com as referências para os demais servidores, sendo que este vetor é ordenado pelo chamado *success factor* associado a cada elemento. No momento em que o algoritmo de busca é executado, este procura uma tupla compatível no processo com o maior valor associado. Na sequência, o *success factor* do elemento é recalculado, sofrendo um incremento se uma tupla foi encontrada, ou um decremento em cenário oposto. O procedimento efetua buscas nos nós, referenciados pelos elementos do vetor, até encontrar uma tupla adequada ou esgotar os elementos do vetor.

Para realizar a busca sequencial nos processos que compõem o espaço de tuplas, o processo local invoca um método de busca não bloqueante, independentemente da chamada original ser bloqueante ou não-bloqueante. Se a tupla for encontrada, esta é simplesmente retornada. No entanto, existe uma diferença no comportamento da busca, em relação ao tipo da chamada original, quando a tupla não é encontrada. Para uma chamada assíncrona, como por exemplo `inp`, o retorno é nulo para uma tupla não encontrada. No entanto, para uma chamada síncrona, como por exemplo `in`, o processo local realiza

uma segunda busca em todos os demais processos, diferenciando-se da primeira chamada por tratar-se da invocação de um método bloqueante, que apenas retornará quando for inserida uma tupla compatível com o *template* fornecido pela aplicação.

2.1.6 Conclusão

Nesta seção foi apresentado o modelo de comunicação de Linda, assim como plataformas que empregam este modelo, que propiciam a construção de uma camada de abstração para a comunicação entre diversos processos distribuídos, permitindo o desenvolvimento de aplicações com desacoplamentos temporal e espacial. Adicionalmente, o conceito de espaço de tuplas pode ser adaptado para cenários que necessitam de outras formas de abstrações, como por exemplo, as relacionados à ausência de garantias na interconexão de dispositivos (seção 2.1.2). Desta maneira, a linguagem de comunicação com o espaço de tuplas permite ser estendida, adicionando funcionalidades que aumentam a capacidade de abstração do modelo, como aplicado no Lime, ou modificando as características que podem ser consideradas indesejadas dependendo da aplicação, como no B-Linda.

A possibilidade de segmentar o espaço de tuplas em múltiplos domínios é uma funcionalidade comum em muitas implementações de espaço de tuplas, como observado na tabela 2.1, apesar desta não ser um recurso presente no modelo original de Linda. Esta funcionalidade permite uma maior conveniência no desenvolvimento de uma aplicação, pois minimiza a complexidade para evitar problemas de *false matching*. As modificações empregadas em B-Linda, apesar de solucionarem o problema de *false matching*, provocam uma dependência da aplicação de uma implementação especifica de espaço de tuplas, assim como rompem com o formato das principais operações de Linda, o que pode ser indesejável. No DTuples toda tupla possui uma identificação, que é utilizada como critério para a distribuição das tuplas, o que contribui para evitar o *false matching*. Porém, esta abordagem aumenta as responsabilidades da aplicação, que passa a interferir na distribuição das tuplas, o que eventualmente pode provocar que um espaço de tuplas que deveria ser distribuído torne-se centralizado em determinado ponto devido as necessidades, ou decisões inadequadas de *design*, de uma aplicação.

A operação `eval` é notadamente não suportada pela maioria dos espaços de tuplas distribuídos, possivelmente por tratar-se de uma operação que depende da capacidade serialização de código da linguagem de implementação do espaço de tuplas, ainda que

Tabela 2.1: Características dos Espaços de Tuplas

Tuple Space (TS)	**Múltiplos *Tuple Spaces***	**Linda API** (`out`, `in` e `rd`)	**Operações assíncronas**	**Operação** `eval`
Dtuples	explícito	não suporta	suporta	não suporta
B-Linda	implícito	não suporta	não suporta	suporta
Tupleware	não suporta	suporta	suporta	não suporta
Lime	explícito	suporta	não suporta	não suporta

presente no modelo tradicional de Linda. As operações assíncronas não são amplamente suportadas, podendo ser consideradas como um recurso adicional oferecido por uma implementação de espaço de tuplas.

Muitas pesquisas têm sido feitas em Espaços de Tuplas. Em virtude do tema escolhido para o nosso trabalho, consideramos aquelas relacionadas à inteligência de enxames (*swarm intelligence*).

2.2 Inteligência de Enxames (Swarm Intelligence)

Swarm Intelligence é uma área da computação que dedica-se ao estudo de algoritmos bioinspirados que permitem a solução de problemas complexos por meio da interação entre agentes simples. Segundo (VASSEV et al., 2012), este segmento fornece um novo modelo comportamental para sistemas multiagentes decorrentes de interações locais entre indivíduos com conjuntos de regras simples e sem conhecimento global. O comportamento dos agentes simula o de animais que resolvem problemas similares aos encontrados na computação, permitindo incorporar as características desejáveis de determinados animais. Para (HARTMANN, 2005), o comportamento dos insetos sociais tem muitas características atraentes, como robustez e confiabilidade através de redundância.

A comunicação entre indivíduos sociais pode ocorrer normalmente de maneira indireta, por meio da modificação do ambiente em que estes encontram-se. Essas modificações são executadas através de um conjunto de ações intrínsecas ao agente. Para determinar qual ação deve ser executada, cada agente realiza uma tomada de decisão baseada no seu estado interno, que normalmente mantém apenas uma pequena quantidade de informações, e nas suas percepções do ambiente. Esta análise do ambiente ocorre apenas em âmbito local, sendo desnecessário que o agente conheça o estado global do sistema.

A execução de uma tarefa complexa é proporcionada pelo comportamento coletivo que emerge da interação entre diversos indivíduos de baixa complexidade que não necessariamente objetivam a execução de uma tarefa global. A forma de interação en-

tre os indivíduos é simples, como por exemplo, o das formigas que simplesmente seguem um caminho de feromônio construído pelos demais indivíduos. No entanto, as diversas interações entre os agentes provocam o surgimento de um comportamento aparentemente racional, ainda que suas ações não possuam uma mecanismo explicito de coordenação.

2.2.1 Otimização por colônia de formigas (Ant colony optimization - ACO)

A ACO é uma metaheurística empregada em problemas de otimização que permite construir e avaliar soluções por meio de uma comunicação indireta entre agentes inspirada no forrageamento das formigas. Nesta abordagem, um conjunto de formigas virtuais é responsável por explorar soluções e identificá-las por meio de uma marcação no espaço de exploração. A sinalização no ambiente permite que qualquer formiga avalie a qualidade de uma solução conhecida, intensificando a marcação em soluções de maior qualidade. Esta avaliação é realizada paralelamente à exploração de novas soluções, porém ocupando diferentes proporções no decorrer do tempo. No início da exploração há uma maior parcela das formigas elencando novas soluções, porém com o decorrer das interações, existe uma parcela maior das formigas avaliando as soluções encontradas. A inspiração do processo de forrageamento das formigas deve-se ao fato dessas serem capazes de encontrarem um caminho curto entre o ninho e uma fonte de alimento. A formação desses caminhos emerge de um comportamento descentralizado das formigas, sendo desnecessária qualquer forma de planejamento ou supervisão. Quando uma formiga encontra uma fonte de alimento, esta regressa ao ninho depositando um substância química denominada de feromônio, formando uma trilha de feromônio no decorrer do percurso. Esta substância sinaliza para as demais formigas que uma fonte de alimento foi encontrada, que no contexto das formigas é a solução de um problema, permitindo que as demais explorem a solução. Segundo (PARPINELLI; LOPES; FREITAS, 2002), a ACO é baseada nas seguintes ideias: i) Cada caminho seguido por uma formiga é associado à solução candidata para um dado problema; ii) Quando uma formiga segue um caminho, a quantidade de feromônio depositada no caminho é proporcional à qualidade da solução candidata correspondente para o problema alvo; iii) Quando uma formiga tem que escolher entre dois ou mais caminhos, o caminho com a maior quantidade de feromônio tem uma probabilidade maior de ser escolhida pela formiga.

As diversas trilhas de feromônio competem entre si, sendo que os caminhos demarcados que possuem quantidades mais concentradas de feromônio tendem naturalmente a atrair uma quantidade maior de formigas. Esta intensidade de feromônio está relacionada fortemente com a avaliação que as formigas realizam sobre as soluções encontradas.

Sempre que uma formiga segue uma trilha existente, esta deposita uma quantidade adicional de feromônio que torna o caminho mais atrativo. No entanto, esta característica não é suficiente para que as formigas descubram um caminho curto, ou a solução com boa qualidade na perspectiva da computação, pois desta maneira os caminhos que inicialmente fossem mais intensificados, possivelmente por serem descobertos no início do forrageamento, dominariam as formigas que eventualmente estariam intensificando uma caminho mais curto. Neste contexto, o ambiente exerce um papel indispensável para que os caminhos curtos sobressaiam-se, pois provoca o decaimento, por evaporação, da quantidade de feromônio depositado. Os caminhos mais longos sofrem um impacto maior da evaporação, pois as formigas são capazes de realizar menos jornadas entre a fonte de alimento e a colônia quando comparado aos caminhos mais curtos, ou seja, a reposição da quantidade de feromônio depositado pode ser insuficiente para repor a evaporação, provocando o desaparecimento de caminhos longos.

A ACO destaca-se pela aplicabilidade em diferentes segmentos da computação. Por exemplo, na área da mineração de dados, o Ant-Miner, proposto por (PARPINELLI; LOPES; FREITAS, 2002), emprega uma colônia, formada por uma única formiga, para explorar regras de classificação com o intuito de dar suporte a tomada de decisão do usuário. Nesta implementação, a quantidade de feromônio depositada pela formiga é proporcional a uma avaliação efetuada sobre a qualidade da solução. Esta estratégia pode ser aplicada em segmentos que extrapolam a mineração de dados, pois independe dos critérios de avaliação da solução e da área de aplicação da metaheurística. Desta forma, soluções consideradas de baixa qualidade interferem de maneira menos significante em agentes que poderiam estar explorando novas soluções. Além disso, o tempo para que todas as formigas convirjam para a solução que mais destaca-se pode ser reduzida, pois esta passa a receber aditivos de feromônio mais expressivos.

2.2.2 Cemitério de formigas

O cemitério de formigas é um fenômeno no qual os corpos dos indivíduos mortos de uma colônia são agrupados em determinadas regiões sem que exista uma forma de coordenação explicita para a formação do local. O local surge a partir de um comportamento natural dos membros da colônia que, segundo (MARTIN; CHOPARD; ALBUQUERQUE, 2002), possuem a seguinte forma de inteligência: i) Eles preferencialmente pegam um corpo que pertence a um pequeno grupo de corpos; ii) A probabilidade de depositar um corpo aumenta conforme o tamanho do grupo de corpos. Como a probabilidade de uma formiga decidir carregar um determinado corpo é inversamente proporcional à quan-

tidade de corpos agrupados no local, é provável que a maior parte dos corpos carregados por uma formiga foram originados de pequenos agrupamentos. Desta forma, os locais que possuem uma elevada quantidade de corpos são predispostos a sofrerem poucas remoções de corpos. Quando uma formiga está carregando um corpo, a probabilidade desta formiga depositar o corpo em um local é proporcional à quantidade de corpos presentes nas imediações, ou seja, locais com grandes quantidades de corpos possuem uma tendência a sofrerem incrementos frequentes na quantidade de indivíduos mortos.

Na proposta de (CHATTY et al., 2011), o fenômeno de cemitérios é empregado na coordenação de robôs que possuem a responsabilidade de agrupar objetos. Apesar da tarefa ser complexa, as ações dos indivíduos robôs são muito simples, sendo compostas apenas por operações que permitem carregar e depositar objetos. As percepções que os robôs possuem do ambiente são limitadas, permitindo detectar apenas objetos ou agentes que estão dentro de um pequeno campo de visão. Esta característica e o baixo requerimento de comunicação permite a construção de agentes que demandam uma pequena quantidade de recursos. A comunicação ocorre de maneira indireta, no qual cada objeto depositado no ambiente atua como uma marcação, influenciado a decisão dos demais.

Em (MONMARCH, 1999) é proposta uma abordagem inspirada no cemitério de formigas para classificação de dados. Locais que possuem uma elevada concentração de um determinado tipo de dado tendem a atrair dados similares, que são depositados por agentes com uma maior frequência. Inicialmente a técnica bioinspirada é empregada para dividir os dados em grupos, porém há ainda elementos em locais equivocados. Sobre cada grupo é aplicada uma técnica não baseada em *swarm intelligence* para reduzir o número de erros, provocando um aumento da já elevada quantidade de grupos. Após estas duas fases é aplicada uma versão modificada do algoritmo de cemitério de formigas para reduzir o número de grupos equivalentes. Nesta terceira fase as formigas podem movimentar apenas os grupos resultantes da segunda fase, permitindo que os grupos similares sejam combinados ao final desta etapa. Analogamente à segunda fase, há erros na classificação que são solucionados pela aplicação de uma etapa similar à segunda, porém que permite apenas movimentar os grupos resultantes da segunda fase. Desta forma é possível aproveitar das características distribuídas do cemitério de formigas e reduzir o número de erros com uma técnica complementar.

Este fenômeno de cemitério pode ser aplicado em cenários onde é necessária a agregação de objetos reais ou abstratos, demonstrando a flexibilidade da abordagem. A característica do algoritmo ser completamente distribuído e não requerer recursos de comunicação direta entre os agentes permite o desenvolvimento de aplicações escaláveis. Além disso, a simplicidade das formas de interação propicia a construção de agentes

com um baixo nível de complexidade e que não necessitam manter grandes quantidades de informações, permitindo que estes desloquem-se de maneira mais rápida quando distribuídos em ambientes de rede, onde há um custo associado ao volume de tráfego. Além disso, é possível adaptar o algoritmo para trabalhar com múltiplos tipos de objetos, sendo necessário apenas um procedimento para determinar a similaridade entre dois objetos.

2.2.3 SwarmLinda

O SwarmLinda, proposto por (MENEZES; TOLKSDORF, 2003), é uma implementação de espaço de tupas que emprega técnicas de *swarm intelligence* para proporcionar uma solução mais escalável. A ideia central consiste em reduzir o *overhead* da comunicação, entre as instâncias do espaço de tuplas, por meio da tomada de decisões simples que são baseadas exclusivamente em informações locais. Nesta abordagem, o funcionamento do espaço de tuplas emerge das interações entre indivíduos muito simples, inspirados no comportamento das formigas, e que atuam de maneira descentralizada. Contrapondo-se a implementações em que a distribuição das tuplas é realizada por meio de um função *hash*, no SwarmLinda o critério para armazenar uma tupla em um determinado local é baseado na similaridade entre esta tupla e as tuplas presentes no local. Segundo (CASADEI et al., 2007), o objetivo é que as tuplas similares sejam armazenadas próximas umas das outras, formando um *cluster* baseado nos seus *templates*.

Quando uma operação do espaço de tuplas é executada, uma formiga artificial é instanciada com a responsabilidade de produzir uma alteração no espaço de tuplas compatível com a operação. Este agente percorre os diversos servidores do espaço de tuplas, no denominado *movement phase*, para cumprir seu objetivo, empregando critérios de exploração inspirados na metaheuristica ACO. As transições entre os servidores ocorrem sempre baseadas em informações disponíveis localmente.

A operação `in` ou `rd` é efetuada através de um agente *template-ant*, que possui a responsabilidade de percorrer os servidores, que compõem o espaço de tuplas, em busca de uma tupla compatível com o *template* fornecido pela aplicação. Para evitar que uma *template-ant* procure continuamente uma tupla que ainda não foi produzida, a formiga pode parar a busca após um longo período percorrendo o espaço de tuplas. Após decidir parar a busca, a formiga dorme por um determinado período antes de retomar a busca, ou renasce em outro local quando a formiga já dormiu muitas vezes. Ao encontrar uma tupla compatível, a formiga deve regressar ao seu local de origem para entregar a tupla, de forma transparente, para a aplicação. A tupla entregue é apenas uma cópia da tupla localizada em situações nas quais a formiga, responsável pela tarefa, foi originada a partir

de uma operação `rd`.

A operação `out` é efetuada através de um agente *tuple-ant*, que possui a responsabilidade de percorrer os servidores, que compõem o espaço de tuplas, em busca de um local adequado para depositar a tupla fornecida pela aplicação. Este processo utiliza uma abordagem inspirada na capacidade das formigas de agruparem itens, como ocorre no fenômeno do cemitério de formigas, que permite agrupar as tuplas que possuem formatos similares. Para determinar a probabilidade de uma *tuple-ant* depositar a tupla em servidor *TS*, a formiga computa a intensidade de similaridade (equação 2.1) entre sua tupla e as tuplas disponíveis no servidor. Sendo que a função γ, que permite avaliar a similaridade entre duas tuplas, não possui uma implementação padrão no SwarmLinda. Segundo (CASADEI et al., 2007), um estudo sobre uma função de similaridade apropriada para uma aplicação deve ser feita considerando questões como número de *templates*, topologia da rede e o domínio da aplicação.

$$F = \sum_{\forall t \in TS} \delta(tu, t) \tag{2.1}$$

Para garantir um limite máximo de transições necessárias para a efetivação da operação, a probabilidade da formiga depositar a tupla, definida pela equação 2.2, depende também da quantidade de servidores percorridos pela *tuple-ant*. Para atingir este efeito, a variável K é inicializada com a quantidade máxima de saltos que é permitida, sendo decrementada após a formiga realizar um transição entre dois servidores. Na proposta de (MENEZES; TOLKSDORF, 2003), o autor considera que a probabilidade de depositar uma tupla depende da memória de curto prazo da formiga, sendo que a equação 2.2 representa a probabilidade para uma formiga sem memória. Em um cenário onde é configurada uma pequena memória para as formigas, a probabilidade de depositar uma tupla é obtida pelo um cálculo ponderado envolvendo as probabilidades obtidas nos servidores mais recentemente percorridos pela formiga, incluindo o servidor atual. Este critério permite que a *tuple-ant* deposite a tupla considerando não apenas a similaridade das tuplas presentes em um servidor, mas também dos servidores próximos que foram percorridos.

$$P_D = \left(\frac{F}{F + K} \right)^2 \tag{2.2}$$

No momento em que uma formiga alcança seu objetivo, esta regressa ao seu servidor de origem utilizando-se de sua memória. Durante este percurso, a formiga deposita um feromônio que sinaliza para as demais formigas o nível de sucesso proporcionado pela trilha, sendo empregado pelos agentes como critério para a escolha de caminhos. Segundo

(MENEZES; TOLKSDORF, 2003), esses caminhos podem ser exploradas para otimizar o desempenho do sistema: "em vez de consultas para conjuntos de réplicas, a *template-ant* vai diretamente para onde se espera uma tupla compatível". Esta marcação é especifica para o formato da tupla que está envolvida na operação, proporcionando apenas a atração de formigas que estão relacionadas a este formato. Para a *template-ant* e *tuple-ant* esse formato são, respectivamente, o próprio *template* e o *template* mais genérico da tupla.

A formação de *clusters* de tuplas é elemento chave na concepção do SwarmLinda, pois em conjunto com o feromônio permite que *template-ants* explorem o espaço de tuplas de maneira mais efetiva. No entanto, esta abordagem torna-se um problema quando há uma concentração excessiva de tuplas, pois permite que uma pequena parcela dos servidores fiquem sobrecarregados devido a um intenso fluxo de formigas. Em (CASADEI et al., 2007) é proposto o SwarmLinda com *Anti-Over-Clustering* para evitar a concentração excessiva de tuplas em determinados pontos do espaço de tuplas, porém esta abordagem provoca um adiamento do armazenamento da tupla por parte de um agente *tuple-ant* que encontre-se em um servidor sobrecarregado, provocando um movimento excessivo de formigas quando comparado a abordagem de (MENEZES; TOLKSDORF, 2003). O critério empregado na decisão de depositar uma tupla, na proposta de (CASADEI et al., 2007), consiste em uma probabilidade determinada pela similaridade entre a tupla carregada e as tuplas armazenadas no servidor em que o agente encontra-se, considerando também uma capacidade máxima de tuplas por servidor. Esta capacidade é utilizada para ajustar os parâmetros da equação 2.3.

$$P'_D = P_D - \left[0.01 + \left(\frac{P_D - 0.01}{(1 + 0.5e^{-b(X-2m)^2})}\right)\right] \tag{2.3}$$

O parâmetro m é configurado considerando o valor de X, que representa a quantidade de tuplas depositadas em um servidor, quando este apresenta o maior valor da curva derivativa. A equação considera também a probabilidade original de depositar a tupla (P_D). O valor de b depende da quantidade máxima de tuplas esperada em um servidor, sendo que b é menor conforme maior é o valor do número máximo de tuplas. Desta forma, ainda que determinado servidor possua muitas tuplas similares a tupla carregada por uma formiga, a probabilidade da tupla ser depositada neste servidor é pequena quando a quantidade de tuplas estiver próxima do máximo estabelecido. É necessário destacar que esta quantidade máxima necessita ser configurada previamente, sendo que a definição de um valor adequado depende intrinsecamente da aplicação que emprega o espaço de tuplas. Além disso, ao contrário do que sinaliza o nome, este valor não é efetivamente um limite superior para a quantidade de tuplas, uma vez que a probabilidade de depositar uma

tupla apenas aproxima-se de zero conforme a quantidade máxima é atingida.

A abordagem de (CASADEI et al., 2007) impõe um custo adicional, pois quando uma *tuple-ant* encontra um servidor sobrecarregado que possui um elevado nível de tuplas similares, esta formiga tende a continuar a exploração do espaço de tuplas ao invés de armazenar a tupla em um servidor próximo. Este comportamento provoca um aumento na quantidade de saltos necessários para que o agente alcance seu objetivo, reduzindo o desempenho do espaço de tuplas. É necessário destacar ainda que o *over-clustering* seja indesejável para o SwarmLinda, a formação de *clusters* é uma característica necessária para o funcionamento desta abordagem de espaço de tuplas.

2.2.4 Conclusão

Nesta seção foram apresentados dois fenômenos envolvendo o comportamento das formigas e exemplos de aplicações. Foram expostas duas variações de uma implementação singular de espaço de tuplas que emprega agentes com características comportamentais inspiradas nas formigas para a execução das principais operações disponibilizadas por Linda. Observou-se que esta abordagem utiliza-se de uma técnica de otimização com trilhas de feromônio, porém esta é suscetível a problemas de *over-clustering* devido ao critério utilizado para depositar uma tupla. Além disso, foi apresentada uma abordagem alternativa de SwarmLinda que possui uma funcionalidade de *Anti-Over-Clustering*, porém esta introduz um *overhead* devido ao aumento no tempo necessário para depositar uma tupla. A lacuna provocada pela ausência de um mecanismo eficiente para distribuição de tuplas mostra-se como uma importante oportunidade que deve ser explorada para o desenvolvimento de espaços de tuplas mais eficientes.

2.3 Campos Magnéticos

O nosso trabalho visa alterar a distribuição de tuplas em espaços de tuplas. A expectativa é fazê-lo por meio do conceito de campos magnéticos virtuais.

2.3.1 Aplicação na Computação

Com inspirações provenientes da física, o conceito de campos magnéticos virtuais foi proposto em (LIMA; CALSAVARA, 2010). Este conceito é aplicado na computação em problemas como balanceamento de cargas, replicação de dados e roteamento de mensagens (conforme apresentado na Tabela 2.2).

Tabela 2.2: Aplicações de Campos Magnéticos Virtuais

Referência	Aplicação
(LIMA; CALSAVARA, 2010)	Introdução do conceito para balanceamento de carga
(CALSAVARA; LIMA, 2010)	Roteamento baseado em atração de mensagens
(GALPERIN; LIMA; CALSAVARA, 2011)	Seleção de *Score Managers* em sistemas de reputação
(CALSAVARA; LIMA, 2011)	Balanceamento de carga em sistemas de larga escala
(COAN; CALSAVARA; LIMA, 2012)	Uso eficiente da bateria em redes de sensores sem fio
(GALPERIN; LIMA; CALSAVARA, 2013)	Reputação de nós em redes magnéticas
(ANGONESE, 2013)	Balanceamento de carga em réplicas de serviços
(MICHELON et al., 2014)	Replicação de dados em *Mobile Ad Hoc Networks*
(OLIVEIRA et al., 2014)	*Anycasting* em *Delay Tolerant Networks* (DTNs)
(MICHELON et al., 2016)	Realocação de réplicas em *Mobile Ad Hoc Networks*

Em (CALSAVARA; LIMA, 2011) é apresentada uma abordagem para balanceamento de carga que utiliza-se do conceito de campo magnético virtual. O objetivo é realizar a distribuição de carga sem a utilização de um escalonador centralizado, evitando um gargalo no sistema e um ponto único de falha. Nesta abordagem, cada servidor está associado a uma força dinâmica que representa a capacidade de processamento disponível. As tarefas que necessitam ser processadas são atraídas para servidores que possuem campos magnéticos mais intensos, ou seja, que dispõem de mais capacidade de processamento. Quando um servidor está dentro de um intenso campo magnético que pertence à outro servidor, as tarefas do primeiro são atraídas em direção ao segundo, ou seja, permitindo que um servidor auxilie na execução de tarefas pertencentes a outro servidor.

As relações de influência magnética entre os servidores formam a denominada *magnetization network* que pode ser representada por um dígrafo. Os vértices do dígrafo correspondem aos servidores da rede, enquanto que as arestas representam os relacionamentos de magnetização. A origem das arestas que chegam em qualquer servidor x representam os servidores que magnetizam o servidor x, enquanto que o destino das arestas que partem do servidor x representam os servidores que são diretamente magnetizados pelo servidor x. Há também os servidores magnetizados indiretamente por x, ou seja, são os servidores que podem ser alcançados percorrendo o grafo a partir do servidor x. Analogamente, há servidores que magnetizam indiretamente o servidor x.

As tarefas que devem ser processadas são representas por mensagens que devem ser encaminhadas a um servidor adequado. Sempre que um servidor recebe uma mensagem, este reencaminha-a para o servidor que possui a maior força associada dentre os servidores

que o magnetizam, sendo este servidor denominado de pivô global. Cada servidor mantém uma lista dos denominados pivôs parciais, que são os pivôs globais dos servidores que o magnetizam. Como consequência natural, o pivô parcial que possui a maior força dentre todos os pivôs parciais de um determinado servidor é o pivô global deste servidor. Devido à natureza dinâmica do campo magnético o pivô global e os pivôs parciais de cada servidor modificam-se no decorrer do tempo, sendo necessário um algoritmo para atualizar o estado da *magnetization network*. Em (CALSAVARA; LIMA, 2011) são apresentados dois algoritmos para este propósito, sendo denominados respectivamente de *QuickPath* e *ShortPath*.

O *QuickPath* é um algoritmo que propaga modificações na *magnetization network* por meio de mensagens de notificação que contém a força e identificação do pivô global do servidor remente da notificação. As notificações enviadas por um determinado servidor são destinadas à apenas os servidores magnetizados por este. Quando um servidor recebe uma notificação, este atualiza a identificação e a força do pivô parcial referente ao remente. Se a notificação provocar uma modificação na identificação ou força do pivô global do destinatário, então este encaminha uma mensagem de notificação para seus servidores magnetizados diretamente. Este critério é empregado para minimizar a quantidade de notificações na *magnetization network*. De acordo com (CALSAVARA; LIMA, 2011), este procedimento produz duas desejáveis propriedades: i) o algoritmo estabiliza com uma quantidade finita de tempo para um número finito de servidores; ii) o algoritmo atualiza a percepção de cada servidor x que tenha um pivô com rotas acíclicas de x para o seu pivô.

O *ShortPath* é um algoritmo que garante a entrega das tarefas pelo caminho mais curto entre um servidor e seu pivô global. As mensagens de mudança de força de um servidor x contém um *timestamp* referente ao momento da alteração e a distância entre o servidor x e o receptor da mensagem, além dos campos seguintes campos: i) identificação do remetente da mensagem; ii) identificação do destinatário da mensagem; iii) identificação do servidor x que sofreu a mudança de força; iv) força do servidor x. O *timestamp* é utilizado para detectar e descartar mensagens desatualizadas de mudança de força, pois não há uma garantia de ordem na entrega das mensagens. Segundo (CALSAVARA; LIMA, 2011), a distância é utilizada para dois propósitos: i) para determinar o caminho magnético um servidor e o seu pivô global em situações onde existe mais do que um caminho; ii) a detecção de *loops* de mensagens que podem ocorrer devido a ciclos de magnetização na rede. Quando uma mensagem de modificação de força provoca uma alteração no pivô de um servidor x, este encaminha mensagens de alteração de pivô para os servidores que são magnetizados por x. Ao receber uma mensagem de alteração de

pivô, o destinatário analisa se há necessidade de alterar a força ou a identificação do seu pivô global, propagando mensagens de modificação quando necessário.

Para evitar que mudanças sutis na força de um servidor produzam uma elevada quantidade de mensagens, o *QuickPath* e o *ShortPath* utilizam-se de um *threshold* para determinar quando uma alteração de força é relevante, não propagando alterações que estão abaixo do *threshold*. Desta maneira, pequenas flutuações na carga dos servidores deixam de impactar negativamente no desempenho global do sistema.

Devido ao objetivo do nosso trabalho, analisaremos a influência de campos magnéticos em formigas reais.

2.3.2 Influência em formigas

Nesta seção será apresentada uma inspiração na biologia para afetar o comportamento das formigas através de campos magnéticos.

De acordo (GERBIER et al., 2008), em muitas espécies de formigas, o movimento de indivíduos ao redor do ninho ocorre essencialmente de maneira coletiva, através de trilhas químicas estabelecidas pelos companheiros de ninho. Estas trilhas de feromônios atuam como um mecanismo que permite as formigas difundirem locais que atendem seus interesses, além de permitir uma otimização do caminho. Como afirma (OLIVEIRA et al., 2010), feromônios são o principal mediador da comunicação nestes animais, sendo utilizados por uma forrageadora para transferir informações até as demais trabalhadoras sobre fontes de alimentos ou para avisar as formigas sobre um ataque iminente.

A formação de trilhas químicas constitui-se para as formigas como um impontante mecanismo de orientação para a exploração do ambiente externo ao ninho. No entanto, há estratégias complementares para a orientação das formigas, que segundo (WAJNBERG et al., 2010), dependendo do seu *habitat*, as formigas podem usar pontos de referência, feromônios, vibrações, gravidade, bússola solar e luz polarizada para orientação. Esses mecanismos possuem diferentes contribuições dependendo da espécie e do ambiente, sendo que um mecanismo de orientação secundário pode adquirir maior relevância quando parte dos demais sentidos são inutilizados.

Para (BANKS; SRYGLEY, 2003), as formigas cortadeiras possuem um sistema de referência direcional, que permite manter um mínimo de orientação quando há uma deterioração ou destruição das trilhas químicas. Além disso, este sistema complementar é especialmente relevante quando a formação das trilhas ainda está em processo inicial. Em (BANKS; SRYGLEY, 2003) é apresentado um experimento sobre os efeitos da exposição

de formigas forrageadoras, da espécie *Atta colombica*, a um campo magnético reverso em momentos de ausência de feromônio. Os resultados demonstram que esta espécie de formiga sofre uma reorientação significativa no seu deslocamento, provocada pelo campo magnético reverso, apenas em dias nublados. De acordo com (BANKS; SRYGLEY, 2003), este comportamento sinaliza que a *Atta colombica* possui um sistema de referência direcional baseado no campo magnético da Terra. No entanto, os resultados sugerem que este sistema é ignorado, ou possui uma menor relevância, perante pistas baseadas na luz solar.

A contribuição das informações magnéticas foi também examinada por (ANDERSON; MEER, 1993) nas formigas da espécie *Solenopsis invicta*, em períodos noturnos. Os resultados demonstram que o tempo de formação das trilhas quase dobrou quando as formigas foram expostas a um campo magnético diferente do qual estas foram previamente aclimatadas. Esta característica pode estar relacionada com os resultados apresentados por (ABRAÇADO et al., 2005), no qual uma ressonância ferromagnética (FMR) detectou a presença de material magnético em formigas da espécie *Solenopsis substituta*.

Esta capacidade de detectar a presença de um campo magnético recebe o nome de magnetorecepção. Segundo (ABRAÇADO et al., 2005), a magnetorecepção é um mecanismo de orientação sofisticado, envolvendo um magnetoreceptor, ligado ao sistema nervoso com a amplificação do sinal. Esta habilidade complementa os sentidos que são utilizados durante a exploração do ambiente em que a colônia está localiza. De acordo com (ANDERSON; MEER, 1993), este sentido magnético fornece uma explicação para o sucesso no forrageamento durante momentos de escuridão completa. No entanto, os resultados apresentados pelos diversos autores sugerem que não há um padrão comum para todas as espécies de formigas, ainda que estes sofram influências de campos magnéticos.

2.3.3 Conclusão

Nesta seção foram apresentadas as relações de campos magnéticos com duas áreas distintas da ciência, a computação e a biologia. Na primeira, o conceito de campo magnético serviu de inspiração para uma estratégia de balanceamento de carga. Na segunda, foi apresentado não um resultado proveniente da inspiração humana aplicada a computação, mas um comportamento de elementos da natureza que pode prover inspiração para o desenvolvimento da computação.

2.4 Considerações Finais

Neste capítulo foram apresentados os principais conceitos envolvendo espaço de tuplas, assim a apresentação de diversas plataformas inspiradas no modelo tradicional de Linda. Adicionamento, foram apresentados inspirações provenientes da biologia que possuem aplicações diretas na computação, com destaque para uma implementação de espaço de tuplas que incorpora técnicas de *Swarm Intelligence*. Por fim, o conceito de campos magnéticos virtuais foi apresentado, juntamente com estudos sobre como campos magnéticos reais afetam o comportamento de certas espécies de formigas. As relações entre espaço de tuplas, campo magnético virtual e *swarm intelligence* envolvendo formigas serão mais exploradas no Capítulo 3.

Capítulo 3

O Modelo Magnético

A abordagem distribuída utilizada pelo SwarmLinda não requer um mecanismo de coordenação centralizado ou uma visão global de todo o sistema. Todas as funcionalidades do espaço de tuplas são completamente distribuídas e a tomada de decisão não depende de um processamento centralizado. Este é um requesito essencial para sistemas escaláveis. No entanto, no SwarmLinda, escalabilidade é limitada pelo fato de que quanto maior for o número de tuplas semelhantes armazenadas em um único nó, maior é a probabilidade de novas tuplas semelhantes serem depositadas neste nó. Embora a concentração de tuplas similares no mesmo nó possa favorecer o processo de encontrar um local onde determinado tipo de tupla pode ser armazenado, a busca de uma tupla dentro de um grande conjunto de tuplas semelhantes pode ser caro, afetando assim a escalabilidade geral do sistema.

Portando, a fim de estender e explorar o potencial de distribuição completa do SwarmLinda, um método original (também bioinspirado) denominado de *Magnetic SwarmLinda* foi definido pela agregação do conceito de *Virtual Magnetic Fields* ao SwarmLinda. A razão por trás da abordagem magnética é prevenir a concentração excessiva de tuplas em um único nó, que pode degradar consideravelmente o desempenho do processo de recuperação de tuplas, em particular. Ainda que a memória seja considerada um fator importante (uma vez que são necessárias grandes quantidades de memória quando existe uma elevada concentração de tuplas em uma única máquina), o poder de processamento necessário não pode ser ignorado: uma vez que quando há uma grande concentração de tuplas de determinado tipo em um único nó, há provavelmente também muitas formigas visitando este nó com o objetivo de encontrar uma tupla compatível dentre as tuplas disponíveis.

Considerando que $G = (V, E)$ é o grafo da rede, onde V é o conjunto de nós e $E \subseteq V \times V$ é o conjunto de arestas bidirecionais entre pares de nós. $N = |V|$ é o número de nós e $M = |E|$ é o número de arestas de G. $NH(i)$ é definido como o conjunto de

vizinhos diretos do nó $i \in V$ (i.e., $NH(i) = \{j : (i,j) \in E \vee (j,i) \in E\}$.

As próximas seções detalham o modelo do *Magnetic SwarmLinda* através da especificação do comportamento da operação `out`, e das equações que suportam o processo de tomada de decisão para depositar uma tupla, selecionar um caminho e interferência magnética.

3.1 Comportamento das tuple-ants

Quando um servidor recebe uma requisição para executar a operação `out`, os seguintes passos são tomados:

1. O servidor atribui a tupla informada a um novo agente *tuple-ant*, sendo que esta formiga é responsável por depositar a tupla em algum nó no espaço de tuplas. Neste momento, é atribuído o tempo de vida máximo (*Time to Live* - TTL) da formiga em número de saltos.

2. Em seguida, a formiga verifica se há muitas tuplas depositadas no nó corrente que são similares à tupla que está sendo carregada. Esta informação é utilizada para decidir se a tupla deve ser depositada no nó corrente ou não. A probabilidade da tupla ser depositada aumenta conforme a quantidade de tuplas similares cresce (de acordo com a Equação (3.3) detalhada abaixo).

3. Se a formiga tiver depositado a tupla, sua última tarefa antes de morrer é sinalizar o local através da dispersão de feromônio no nó corrente e nos nós vizinhos. Este processo reforça o nó atual como um local adequado para tuplas similares, permitindo que outras formigas interessadas nesse tipo de tupla tenham uma maior probabilidade de encontrar este nó.

4. Se a formiga decidir não depositar a tupla no nó corrente, esta deve escolher um nó adjacente para visitar. Esta escolha é feita estocasticamente com base em informações sobre os nós vizinhos (como será detalhado na Seção 3.3). No entanto, é possível que esta etapa não ocorra devido à ocorrência de um comportamento estranho (que será apresentado na Seção 3.5).

5. Como a formiga move-se para um novo nó, esta torna-se mais velha, i.e., seu TTL é decrementado. Se o TTL atingir zero, a formiga deposita sua tupla independentemente do nó corrente e dispersa feromônio referente ao tipo da tupla (como descrito na etapa 3) antes de morrer. Caso contrário, a formiga continua a partir da etapa 2.

Este comportamento das *tuple-ants* pode ser simplificado dependendo da equação que define a probabilidade de depositar uma tupla. A verificação do TTL pode ser omitida (conforme o fluxograma da Figura 3.1) caso a equação que define a probabilidade de depositar uma tupla garanta que a tupla será depositada quando o TTL atingir zero. No modelo proposto, a equação que define a probabilidade de depositar uma tupla (que será apresentada na Seção 3.2) garante que toda tupla será depositada quando o TTL da formiga responsável pela tupla atingir o valor zero.

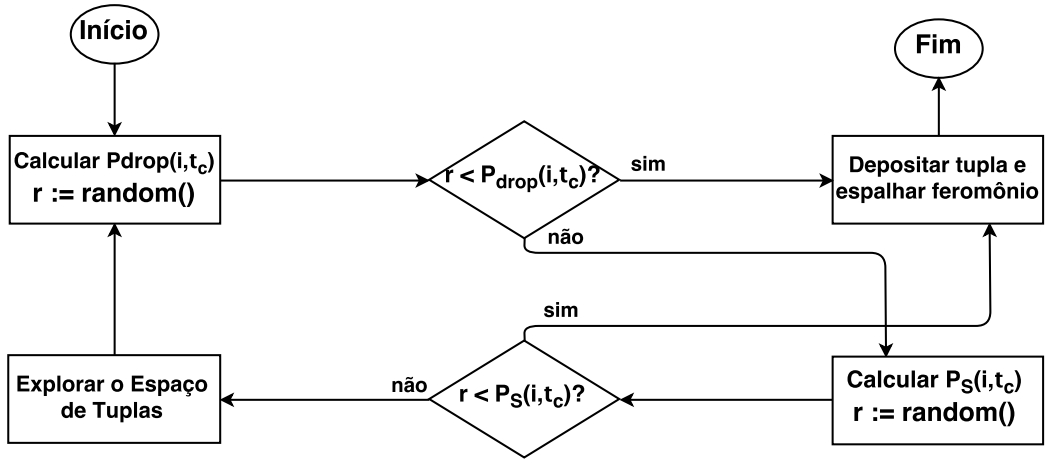

Figura 3.1: Comportamento simplificado das *tuple-ants*

A decisão de depositar uma tupla baseada na concentração de tuplas ocorre de forma probabilística. A partir da concentração de tuplas similares é calculado a probabilidade $P_{drop}(i, \tau_c)$ de depositar a tupla τ_c no nó atual i (conforme será descrito na Seção 3.2). Na sequência, a formiga sorteia um número r ($0 <= r < 1$) e compara com a probabilidade calculada. Se r for inferior à $P_{drop}(i, \tau_c)$, a formiga deposita a tupla e espalha feromônio referente ao tipo da tupla. Caso contrário, a formiga continua a exploração do espaço de tuplas (conforme será descrito na Seção 3.3) desde que não sofra um comportamento estranho (como será descrito na Seção 3.5). Assim como ocorre na avaliação do $P_{drop}(i, \tau_c)$, a análise da probabilidade de comportamento estranho $P_S(i, \tau_c)$ é realizada comparando o valor da probabilidade com um número aleatório r ($0 <= r < 1$).

3.2 Probabilidade de depositar uma tupla

Como mencionado na Seção 2.2.3, a probabilidade de depositar uma tupla no nó corrente depende da concentração de tuplas similares no referido nó. A *função de similaridade* – designada por $sim(\tau_A, \tau_B) \in [0, 1]$ – deve ser definida de modo que o grau de similaridade entre duas tuplas τ_A e τ_B possa ser avaliado. Para os experimentos descritos

no Capítulo 4, a função de similaridade binária da Equação (3.1) foi utilizada. Neste caso, tuplas são consideradas similares se suas quantidades de campos e seus respectivos tipos de cada campo são idênticos.

$$sim(\tau_A, \tau_B) = \begin{cases} 1, & if\ template(\tau_A) = template(\tau_B) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

A concentração de tuplas em determinado nó i de tuplas que são similares à τ_c (a tupla carregada) – designado por $C(i, \tau_c)$ – é dada pela Equação (3.2). $C(i, \tau_c)$ é determinado através da comparação de τ_c com cada tupla τ_s armazenada em $i \in V$.

$$C(i, \tau_c) = \sum_{\forall \tau_s \in i} sim(\tau_c, \tau_s) \tag{3.2}$$

A probabilidade de uma formiga depositar sua tupla τ_c em algum nó i – designado por $P_{drop}(i, \tau_c)$ – é dada pela Equação (3.3). Observe que $P_{drop}(i, \tau_c)$ depende do valor de TTL.

$$P_{drop}(i, \tau_c) = \left(\frac{C(i, \tau_c)}{C(i, \tau_c) + TTL} \right)^2 \tag{3.3}$$

Claramente, a probabilidade de depositar a tupla aumenta conforme TTL diminui. Consequentemente, TTL representa o limite superior de saltos que uma formiga percorre com o objetivo de depositar sua tupla. O valor do TTL previne que formigas movam-se indefinidamente pelos nós do espaço de tuplas, evitando assim uma sobrecarga de todo o sistema.

Por outro lado, a probabilidade de depositar uma tupla é maior quando a concentração de tuplas similares é elevada, uma vez que a influência de TTL em $P_{drop}(i, \tau_c)$ diminui, permitindo assim a formação de *clusters* de tuplas similares.

3.3 Movimentação das formigas

Se uma formiga não atingir seu objetivo no nó atual, ela precisa escolher um nó vizinho para visitar com o intuito de continuar tentando atingir seu objetivo. A fim de aumentar a chance de uma formiga atingir seu objetivo, ela deve mover-se em direção ao local que muitos outros indivíduos portadores de tuplas similares foram. Além disso, a formiga deve considerar o número de tuplas similares no potencial nó de destino. A Equação (3.4) define a probabilidade de uma formiga em um nó i carregando uma tupla τ_c de mover-se para um outro nó j, onde $Ph(i, \tau_c)$ representa a quantidade atual de feromônio

de tipo τ_c presente no nó $i \in V$.

$$P(\tau_c)_{i,j} = \frac{C(j,\tau_c) + Ph(j,\tau_c)}{\sum_{\forall n \in NH(i)}(C(n,\tau_c) + Ph(n,\tau_c)} \tag{3.4}$$

Conforme o diagrama da Figura 3.2, a escolha do próximo nó a ser visitado depende de um valor aleatório r ($0 <= r < 1$). A formiga percorre o vetor $NH(i)$, que contém os nós adjacentes do nó corrente i, comparando o valor de r com a soma P da probabilidade $P(\tau_c)_{i,j}$ do nó que está sendo avaliado ($index$ atual) e das probabilidades nós que já foram avaliados. O nó escolhido é determinado pela iteração no qual o valor de P atinge um valor igual ou superior ao valor de r. É importante destacar que se $index$, que é utilizado como índice para avaliar os vizinhos de i, atingir o valor máximo ($|NH(i)| - 1$), o valor de P atingirá 1. Desta forma, não há a possibilidade da formiga não escolher o próximo nó a ser visitado.

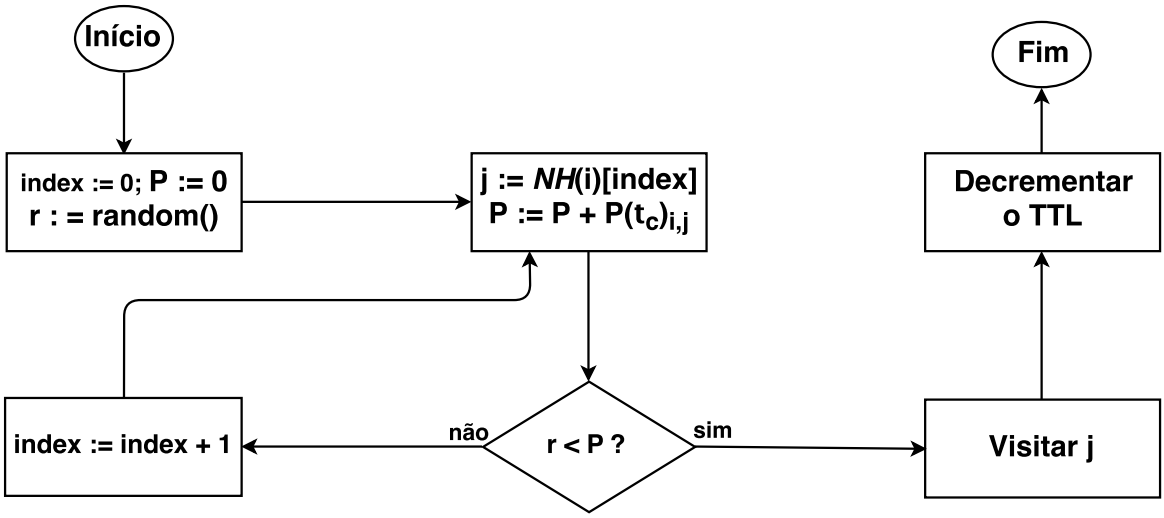

Figura 3.2: Procedimento de exploração do Espaço de Tuplas

Quando a formiga encontra o nó j que satisfaz a condição $r < P$, esta desloca-se para o nó j. Além disso, a formiga decrementa o seu TTL, ou seja, a formiga torna-se mais velha.

3.4 Evaporação de Feromônio

Conforme discutido na Seção 2.2.1, a evaporação de feromônio é uma mecanismo essencial para tornar o sistema adaptável, uma vez que trilhas referentes a regiões que já não tem uma concentração significante de tuplas devem desaparecer. Um mecanismo de "evaporação" previne que agentes tenham um comportamento caótico que comprometeria

o funcionamento adequado do sistema. Além disso, o mecanismo de evaporação torna os caminhos curtos mais atraentes, otimizando assim a quantidade de saltos necessários para que um agente atingir as regiões desejadas do grafo. A Equação (3.5) define como os feromônios de cada nó i desaparecem conforme o tempo t avança.

$$Ph_t(i, \tau_c) = Ph_{(t-1)}(i, \tau_c)(1 - \rho) \tag{3.5}$$

Todos os nós do sistema decrementam as suas quantidades de feromônio de acordo com a taxa de evaporação ρ ($\rho \in [0, 1]$). É importante destacar que ρ não deve ser elevado ao ponto de provocar que novas trilhas nunca sejam exploradas. Se ρ é muito baixo, o deslocamento dos indivíduos pode ser afetado negativamente por trilhas que levam à regiões que deixaram de possuir grandes quantidades de tuplas similares.

3.5 Interferência Magnética

Interferência Magnética é um mecanismo cujo objetivo é evitar a concentração de tuplas em poucos (possivelmente sobrecarregados) nós. A abordagem "tradicional" do SwarmLinda determina que a probabilidade de um nó receber uma nova tupla é proporcional ao número de tuplas semelhantes que este detém. Esta propriedade melhora o desempenho das subsequentes operações de recuperação de tupla, pois permite que o processo de busca ocorra orientado a apenas uma determinada região do espaço de tuplas. No entanto, quando há uma excessiva quantidade de tuplas em um único nó, é provável que a capacidade de processamento do nó torne-se um gargalo.

A sobrecarga de processamento está relacionada ao fato que um nó contendo muitas tuplas provavelmente receberá uma elevada quantidade de formigas tentando recuperar e/ou depositar tuplas. Isto é particularmente grave para *template-ants* que naturalmente exigem mais processamento que as *tuple-ants*, já que uma *template-ant* precisa executar varias operações de comparação para encontrar uma tupla que satisfaça as restrições especificas em seu *template*. E isso é computacionalmente dispendioso para o espaço de tuplas.

O *nível magnético* a respeito de uma determinada tupla τ_c que algum nó i está exposto é definido pela Equação (3.6).

$$M_L(i, \tau_c) = Max\{C(n, \tau_c) : n \in NH(i)\} \tag{3.6}$$

O nível magnético para um nó i corresponde a máxima concentração de tuplas que são similares à tupla τ_c nos nós vizinhos de i. M_L é utilizado para determinar a

força magnética sofrida pelas formigas presentes no nó i, designado por $F_M(i)$ (Equação (3.7)). Esta força magnética é responsável por produzir um "comportamento estranho" em uma *tuple-ant*, fazendo com que sua tupla seja depositada no nó atual, ou seja, antes do esperado.

$$F_M(i, \tau_c) = M_c(i) * \frac{M_L(i, \tau_c)}{M_c(i) + M_L(i, \tau_c)} \tag{3.7}$$

A força magnética que afeta as *tuple-ants* no nó i tende à zero quando não existe nenhum campo magnético de nível significante produzido pelos vizinhos do nó i. $F_M(i, \tau_c)$ depende da constante de *restrição magnética* $M_c(i)$ que representa o número máximo de tuplas que um nó i pode armazenar sem ser considerado "sobrecarregado".

A probabilidade de um "comportamento estranho" durante a fase de movimentação de um indivíduo que está atualmente no nó i carregando uma tupla τ_c é definido na Equação (3.8).

$$P_S(i, \tau_c) = \frac{F_M(i, \tau_c) + C(i, \tau_c)}{\sum_{\forall n \in NH(i)}(C(n, \tau_c) + Ph(n, \tau_c)) + F_M(i, \tau_c) + C(i, \tau_c)} \tag{3.8}$$

O "comportamento estranho" de uma *tuple-ant* consiste na decisão de depositar a tupla carregada τ_c no nó corrente ao invés mover-se para outro nó como é esperado. Um nó i que possui um nó sobrecarregado, por exemplo, está sobre uma forte interferência magnética (calculada a partir de $F_M(i, \tau_c)$) e, consequentemente, uma formiga neste nó possui uma elevada probabilidade de apresentar um comportamento estranho.

Obviamente, se $C(i, \tau_c)$ é elevado, então é muito provável que uma formiga carregando uma tupla τ_c apresentará um comportamento estranho em um nó i. Devido a este comportamento estranho, há uma tendência de formação de *clusters* de nós que contém *clusters* de tuplas similares, ou seja, a sobrecarga de nós é evitada através de *clusters* de *clusters*.

Observe, no entanto, que a interferência magnética não restringe completamente o comportamento dos indivíduos, pois a adaptabilidade do sistema é alcançado exatamente pela tomada de decisões estocásticas. Portanto, ainda que exista uma intensa interferência magnética, uma formiga pode apresentar um comportamento normal conforme apresentado na Seção 2.2.3, embora com uma pequena probabilidade.

É importante notar que o papel dos campos magnéticos na abordagem proposta difere do apresentado na Seção 2.3.1). Na literatura, os campos magnéticos são normalmente utilizados com o intuito de atrair algo. No entanto, na abordagem proposta, o campo magnético atua como um "campo de força" que protege cada nó sobrecarregado, evitando uma concentração excessiva de carga.

3.6 Considerações Finais

Neste capítulo foram apresentadas as equações que definem o comportamento das formigas em relação à abordagem proposta. Foi apresentado a estratégia de movimentação das formigas pelo espaço de tuplas, bem como o comportamento estranho sofrido pelas *tuple-ants* perante a presença de um intenso campo magnético. Além disso, foram apresentados as equações necessárias para a tomada de decisão de uma formiga para armazenar ou não sua tupla em um nó, levando em consideração a quantidade de tuplas similares. Além do comportamento das formigas, também foi apresentado a atuação do ambiente no sistema por meio da da evaporação do feromônio.

Capítulo 4

Avaliação de Desempenho

A fim de avaliar o desempenho da abordagem proposta, denominada de Magnetic SwarmLinda, um simulador para sistemas multiagente foi construído com controle de tempo baseado em eventos. No simulador, o tempo é medido em *unidades de tempo ideal* (UTI), que representa o intervalo de tempo necessário para uma mensagem ser entregue a partir de um determinado nó para um vizinho direto. A abordagem do simulador é similar à utilizada pelo *SimPy*, que é um *framework* para simulação de eventos discretos construído sobre o recurso de *generators* do Python (ROSSUM; EBY, 2005). Estes *generators* são funções especiais – também conhecidas como "*coroutines*" em outras linguagens de programação – que podem retornar uma série de valores em vez de apenas um. Esta propriedade permite que uma chamada de função retorne um valor sem perder o contexto local. Assim, quando uma função é chamada novamente, ela continua a partir do ponto em que a chamada anterior havia parado.

Para fins de simulação, o valor da restrição magnética $M_c(i)$ foi definido como a quantidade "desejável" de tuplas a ser depositada em cada nó. Este valor foi obtido considerando-se uma distribuição uniforme de todas as tuplas pelo espaço de tuplas (a quantidade total de tuplas depositadas no espaço de tuplas é conhecida num cenário controlado). De tal modo, $M_c(i) = T/N, \forall i \in V$, onde T é o número total de tuplas no sistema (independentemente do seu "tipo") e $N = |V|$ é o número de nós (como será detalhado na Seção 4.2).

4.1 Geração de carga

Com o objetivo de avaliar o desempenho da abordagem proposta, um conjunto de processos clientes foi definido para executar operações no espaço de tuplas. Na simulação, considerou-se que, para cada nó $i \in V$, há um processo cliente (exemplificado na Figura

4.1) que é responsável por inserir (operação `out`) e recuperar (operação `in`) tuplas. Em topologias formadas por um número maior de nós (e.g., as topologias que serão definidas na Seção 4.2), a proporção de um processo cliente para cada nó é mantida.

Processos clientes executam periodicamente de forma alternada operações de inserção e recuperação de tupla. A fim de medir o desempenho das operações de recuperação de tupla, nas execuções de simulação, os processos clientes só irão tentar recuperar tuplas que já estão disponíveis no espaço de tuplas, uma vez que atrasos dependentes da aplicação seriam introduzidos se os clientes tivessem que aguardar a produção de uma tupla compatível, aumentando arbitrariamente o atraso médio das operações de recuperação de tupla.

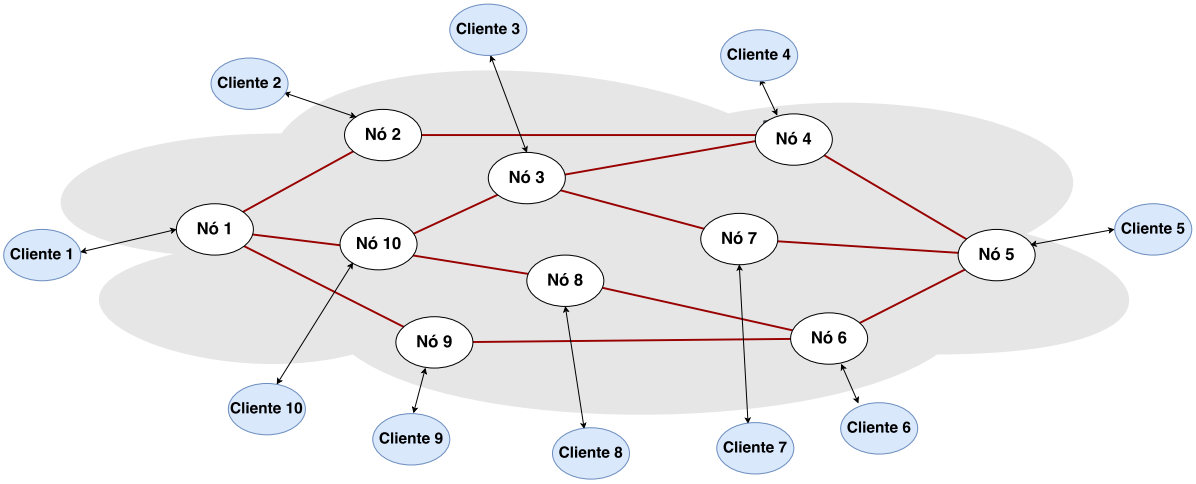

Figura 4.1: Cenário de simulação

No entanto, é importante destacar que o modelo proposto não restringe o número ou disposição dos processos clientes e nem a topologia da aplicação (exceto que o grafo deve ser conexo). Em outras palavras, cenários em que alguns nós têm vários processos clientes conectados e outros nós sem qualquer processo cliente são igualmente possíveis. O ambiente particular da Figura 4.1 permite avaliar a implementação do Magnetic Swarm-Linda em cenários com produção e consumo massivo de tuplas, ou seja, com uma elevada frequência de operações sobre o espaço de tuplas.

4.2 Cenários de Avaliação

Para os cenários de avaliação, grafos aleatórios foram gerados utilizando o NetworkX (HAGBERG; SCHULT; SWART, 2013), que é uma biblioteca em Python para "criação, manipulação, e estudo de estruturas, dinâmicas, e funções de redes complexas." A topologia escolhida para o espaço de tuplas foi a *Watts-Strogatz's Small-World* (WATTS;

STROGATZ, 1998), que representa a estrutura e dinâmica de redes sociais, biológicas, e redes de infraestrutura. Um grafo *small-world* aleatório $G = (V, E)$ é gerado através da criação de uma rede em anel com $N = |V|$ nós, sendo que cada nó é conectado com seus k vizinhos mais próximos (se k é par). Em seguida, cada aresta $(u, v) \in E$ é substituída com probabilidade p por uma nova aresta (u, w), onde $w \in V$ é escolhido aleatoriamente.

A Tabela 4.1 define os parâmetros utilizados para gerar os grafos da simulação. A probabilidade p de reescrever uma aresta da topologia foi escolhida de uma forma que a topologia resultante apresentará um certo grau de "aleatoriedade".

É importante destacar que os valores apresentados na Tabela 4.1, bem como os que serão definidos na Tabela 4.2, são apenas valores *default*, podendo estes serem "sobrescritos" em função dos cenários de avaliação que serão definidos nesta seção (ver Tabela 4.3).

Tabela 4.1: Parâmetros para geração de grafo

Parâmetro	Valor	Descrição
k	4	número médio de vizinhos por nó
N	16	número de nós que compõem o espaço de tuplas
p	30%	probabilidade de reescrever uma aresta da topologia

Para cada uma das S simulações executadas, uma semente diferente para o gerador de números pseudo-aleatórios foi utilizada. Uma vez que cada processo cliente tem a mesma probabilidade de produzir qualquer um dos τ^t tipos de tuplas e seus τ^v possíveis valores quando executa uma operação `out`. Analogamente, cada processo cliente também possui uma probabilidade igual de produzir qualquer tipo e valor de *template* durante a execução da operação `in`.

Inicialmente, antes de recuperar alguma tupla com a operação `in`, os processos clientes executam apenas operações `out` até que T tuplas tenham sido depositadas no espaço de tuplas. Isso é necessário para que o desempenho medido reflita a operação normal do sistema, ignorando eventuais oscilações que podem ocorrer nos estágios iniciais da formação de *clusters* e trilhas de feromônio. Além disso, a quantidade total de tuplas T no espaço de tuplas tende a permanecer inalterada, uma vez que cada processo cliente executa alternadamente operações `in` e `out`. Todos os valores padrão desses parâmetros estão descritos na Tabela 4.2.

A fim de ser capaz de observar as variações no tempo de resposta devido à sobrecarga de certos nós, é necessário que cada nó tenha uma capacidade computacional limitada, assim como os nós no mundo real. Para este efeito, OP é um parâmetro definido na simulação para representar a quantidade de operações que um nó pode processar den-

Tabela 4.2: Parâmetros *default* de simulação

Parâmetro	Valor	Descrição
ρ	20%	taxa de evaporação de feromônio
S	50	o número de execuções de simulação
τ^t	8	o número de tipos de tuplas
T	100,000	o número de tuplas depositadas no espaço de tuplas
τ^v	1,000	o número de valores possíveis para uma tupla
D	1,000	a duração da simulação em termos de número de operações `in`
OP	1,000	o número máximo de operações por unidade de tempo ideal
I	10	o intervalo de tempo entre as operações consecutivas em UTIs

tro de uma unidade de tempo. Estas operações incluem a verificação de correspondência entre uma tupla e um *template*, selecionar um nó vizinho para onde mover-se e decidir depositar ou não uma tupla. Um excessivo número de tuplas em um nó provoca um impacto negativo no desempenho que está relacionado ao processo das *template-ants* de encontrar uma tupla compatível com o *template* solicitado.

Para analisar o comportamento do Magnetic SwarmLinda em diferentes contextos, seis cenários distintos foram definidos (ver Tabela 4.3). Em cada um, a abordagem magnética é comparada com o SwarmLinda "Tradicional" e com o SwarmLinda com a funcionalidade de *Anti-Over-Clustering*. Em cada cenário, o impacto de variar um determinado parâmetro escolhido foi avaliado, no caso de seu valor padrão definido na Tabela 4.1 ou na Tabela 4.2 (dependendo do parâmetro) é sobrescrito. Os outros parâmetros (fixos) são definidos para seus valores padrão.

Tabela 4.3: Cenários de Avaliação de Desempenho

Cenário	Parâmetro	Valores	Avaliação de desempenho para
1	I	3, 4, 5, 6 e 7	altas taxas de requisição de operações
2	I	5, 15, 25 e 35	altas e baixas taxas de requisição de op.
3	τ^v	500, 1000, 1500 e 2000	diferentes valores para cada tipo de tupla
4	k	4, 6, 8 e 10	diferentes números médios de arestas
5	T	(1, 2, 3 e 4) * 10^5	diferentes números de tuplas no sistema
6	τ^t	3, 4, 5, 6, 7 e 8	diferentes números de tipos de tuplas

Nos Cenários 1 e 2, o desempenho das três abordagens é avaliado para diferentes níveis de carga. A fim de fazer isso, o parâmetro de simulação I que representa os intervalos de tempo entre duas operações consecutivas é definido com diferentes valores. Quanto menor é o valor de I, maior é a frequência de operações e, portanto, maior é a carga sobre o espaço de tuplas. O Cenário 1 difere do Cenário 2 no que diz respeito aos

intervalos de tempo escolhidos. No Cenário 1, o espaço de tuplas é extremamente exigido.

Cenário 3 avalia o tempo de resposta para recuperação de tuplas "raras". Quanto maior for o τ^v, maior é a quantidade de possíveis valores que uma tupla pode assumir e menor é o número de tuplas idênticas (uma vez que a quantidade total de tuplas é fixa).

No Cenário 4, todas as três abordagens são avaliadas para diferentes topologias de espaço de tuplas. Com valores de k menores, a topologia da rede torna-se menos densa.

Cenário 5, avalia como o número de tuplas T depositadas no espaço de tuplas na fase de aquecimento afeta o desempenho das três abordagens. Uma vez que cada processo cliente executa alternadamente operações de inserção e recuperação, a quantidade total de tuplas no espaço de tuplas é constante durante a execução da simulação. A partir disto, é possível avaliar o desempenho para cada abordagem em diferentes cargas de tuplas no espaço de tuplas.

Finalmente, no Cenário 6, o desempenho é avaliado para diferentes números de tipos de tuplas. Isso permite medir o impacto de diferentes quantidades de *clusters* de nós, uma vez que cada *cluster* (no modelo magnético) está associado a um determinado tipo de tupla.

Adicionalmente aos cenários definidos na Tabela 4.3, há uma avaliação de desempenho extra que é relativa ao tempo de viagem das formigas. Nesta avaliação, o tempo necessário para avaliar a compatibilidade entre uma tupla e um *template* é desconsiderado. Desta forma, é avaliado apenas o tempo necessário para movimentação das formigas pelo espaço de tuplas. Os parâmetros utilizados nesta avaliação de desempenho são idênticos aos definidos no Cenário 2.

4.3 Resultados de Desempenho

Nesta seção serão apresentados os resultados de desempenho do espaço de tuplas para recuperação de tuplas. Cada gráfico de resultados, que é referente à um dos cenários definidos na Seção 4.2, apresenta o tempo médio para recuperação de uma tupla. Por fim, são apresentados os resultados referentes ao tempo de viagem necessário para recuperar uma tupla, desconsiderando o tempo de processamento para a verificação de compatibilidade entre uma tupla e um *template*. Em cada gráfico é apresentado um intervalo de confiança com um coeficiente de confiança de 95%.

4.3.1 Cenário 1

Os resultados obtidos para o Cenário 1 estão representados na Figura 4.2 que mostra o tempo médio para recuperar uma tupla do espaço de tuplas quando os processos clientes executam operações com intervalos muitos curtos. Então, esses são os resultados de desempenho para um cenário de elevada demanda.

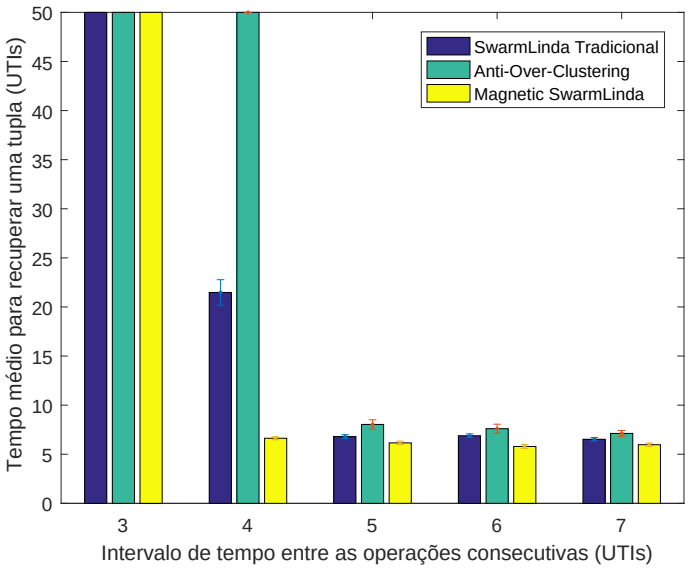

Figura 4.2: Cenário 1 - Altas taxas de requisição de operações

A partir da análise dos resultados obtidos é possível identificar que o ponto de saturação de todas as abordagens. Este ponto é alcançado quando o intervalo de tempo entre as inserções é maior que o tempo médio para recuperar uma tupla. É importante notar que o intervalo de tempo de inserções é o dobro do intervalo de tempo entre operações, uma vez que as operações são alternadas entre inserção e recuperação de uma tupla. Então, neste ponto de saturação, o espaço de tuplas não consegue absorver a carga, provocando uma degradação cumulativa do desempenho do sistema. O ponto de saturação para a abordagem do Magnetic SwarmLinda acontece quando $I = 3$. Para as outras duas abordagens, este ponto ocorre quando $I = 4$. É importante mencionar que o valor de eixo y foram "aparados" em 50 a fim mostrar melhor o comportamento das três abordagens quando $I \geq 4$. Além disso, para a abordagem tradicional, quando $I = 4$, o tempo médio para recuperar uma tupla tende ao infinito para simulações mais longas. Isso acontece porque com $I = 4$ os processos clientes estão inserindo tuplas a cada 8 UTIs, enquanto que o tempo médio para recuperar uma tupla é cerca de 20 UTIs.

4.3.2 Cenário 2

Os resultados de desempenho para o Cenário 2 são apresentados na Figura 4.3. Observe que a abordagem magnética apresentou um desempenho superior em comparação com as outras duas abordagens neste cenário em que o espaço de tuplas não está saturado.

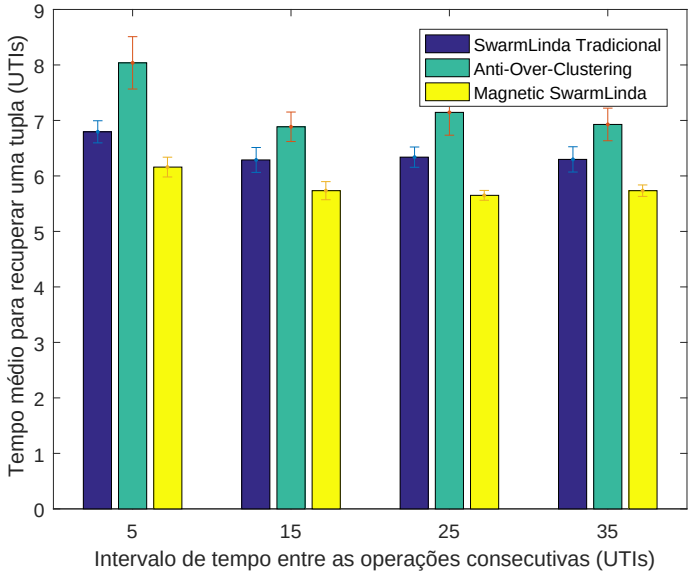

Figura 4.3: Cenário 2 - Altas e baixas taxas de requisição de operações

Conforme esperado, todas as abordagens têm um tempo de resposta cada vez mais rápido para cenários em que a execução das operações por processos clientes é menos frequente. Isto é consequência de um cenário de carga menor que o espaço de tuplas é exposto quando a frequência das operações é baixa. No entanto, esta melhora no desempenho é menos significante que a apresentada no Cenário 1, uma vez que a carga sobre o espaço de tuplas está longe do ponto de saturação do sistema.

4.3.3 Cenário 3

Os resultados de desempenho para o Cenário 3, que são apesentados na Figura 4.4, evidencia que a abordagem proposta apresenta um desempenho melhor quando há uma quantidade menor de possíveis valores para cada tipo de tupla. Este comportamento deve-se ao fato que há uma quantidade maior de tuplas idênticas quando o número de valores possíveis é menor. Desta forma, a quantidade de nós que uma *template-ant* precisa visitar para obter uma tupla compatível tende a ser menor, uma vez que é maior a probabilidade de existirem tuplas compatíveis em múltiplos nós do *cluster* magnético.

Por outro lado, a abordagem magnética apresenta um desempenho inferior as outras abordagens quando há uma pequena quantidade de tuplas idênticas, pois a *template-*

ant é obrigada a visitar mais servidores do *cluster* magnético.

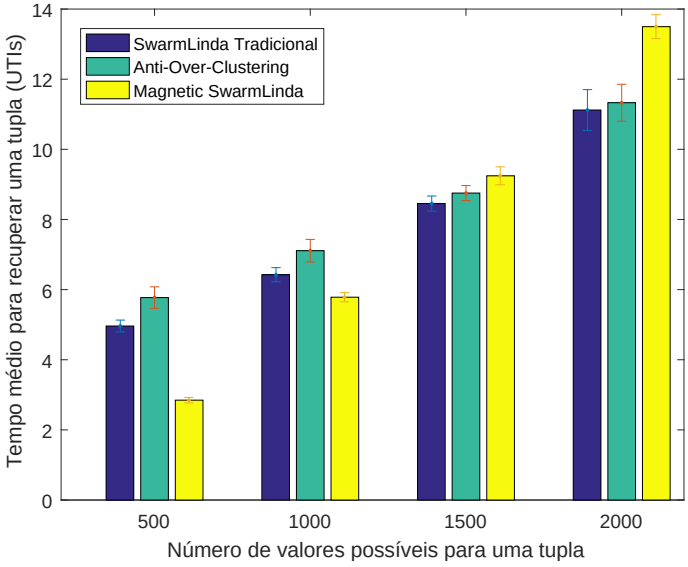

Figura 4.4: Cenário 3 - Diferentes quantidades de valores possíveis por tipo de tupla

O SwarmLinda Tradicional e o SwarmLinda com *Anti-Over-Clustering* possuem um desempenho muito similar entre si quando há uma grande quantidade de valores possíveis por tipo de tupla. Isto deve-se à uma degradação do desempenho do SwarmLinda Tradicional que precisa processar por mais tempo os *template-ants* (em apenas poucos nós) para encontrar uma tupla compatível, uma vez que as tuplas são "raras", o que acaba compensando a tendência do SwarmLinda Tradicional apresentar um desempenho superior ao SwarmLinda com *Anti-Over-Clustering*.

4.3.4 Cenário 4

Os resultados de desempenho para o Cenário 4, que são apresentados na Figura 4.5, demonstram que a abordagem magnética possui um melhor desempenho no tempo de resposta para recuperação de tuplas em diferentes topologias de espaço de tuplas. Este comportamento deve-se ao fato que mesmo em topologias nos quais os nós possuem uma pequena quantidade de vizinhos, a probabilidade de um agente alcançar qualquer *cluster* magnético com um pequeno número de saltos é alta. Como cada *cluster* magnético é composto de múltiplos nós, é provável que exista pelo menos um caminho curto entre qualquer nó do espaço de tuplas e pelo menos um nó de cada *cluster* magnético.

O SwarmLinda com *Anti-Over-Clustering* foi a abordagem que mais beneficiou-se de grafos mais densos, pois uma *tuple-ant* em um nó sobrecarregado possui uma alta probabilidade de encontrar um caminho para outro nó com muitas tuplas similares à sua tupla e que possivelmente não está sobrecarregado. Além disso, um aumento da

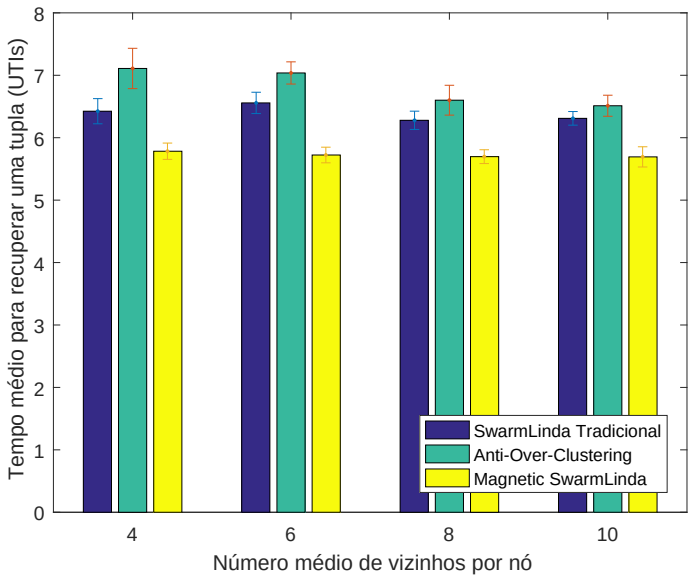

Figura 4.5: Cenário 4 - Diferentes quantidades (médias) de arestas por nó

probabilidade de que nós com tuplas similares estarem próximos favorece as *template-ants* para procurar uma tupla compatível. Enquanto isso, a abordagem tradicional apresenta um desempenho intermediário em relação às demais abordagens.

4.3.5 Cenário 5

Os resultados de desempenho para o Cenário 5, que são apresentados na Figura 4.6, demonstram que a abordagem proposta possui um desempenho superior às demais abordagens para diferentes quantidades de tuplas depositadas no espaço de tuplas. Além disso, o tempo médio para recuperar uma tupla diminui de forma muito expressiva no Magnetic SwarmLinda conforme a quantidade de tuplas aumenta.

O SwarmLinda Tradicional e o SwarmLinda com *Anti-Over-Clustering* apresentam uma melhora menos expressivas quando há muitas tuplas disponíveis no espaço de tuplas. As tendências de melhora no desempenho dessas abordagens são muitos similares entre si para cenários com muitas tuplas, ainda que a abordagem tradicional apresente um desempenho intermediário em relação à todas as abordagens.

4.3.6 Cenário 6

Os resultados de desempenho para o Cenário 6, que são apresentados na Figura 4.7, demonstram que a abordagem proposta é capaz de tirar proveito de cenários onde o número de tipos de tuplas τ^t é menor que o número nós $N = |V|$ que compõem o espaço de tuplas. Isto deve-se ao fato que conforme menor é a quantidade de tuplas, menor é a

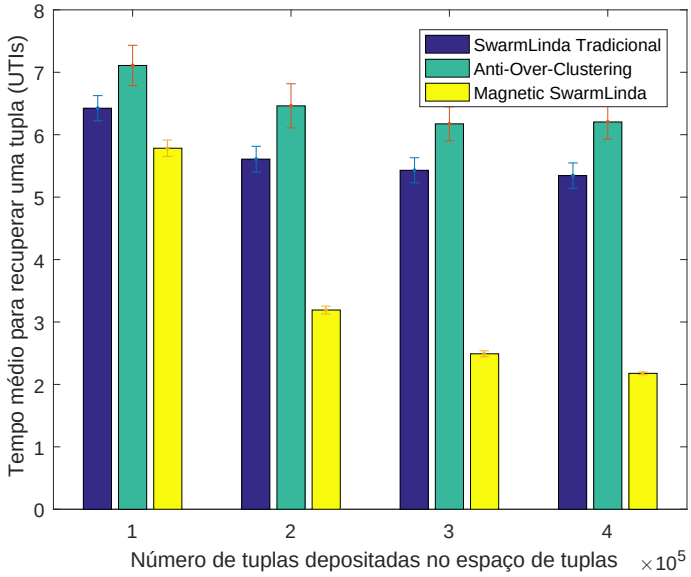

Figura 4.6: Cenário 5 - Diferentes números de tuplas depositadas no espaço de tuplas

quantidade de *clusters* de nós. Desta forma, cada *cluster* magnético é formado por um número maior de nós, o que provoca uma expressiva melhora no desempenho.

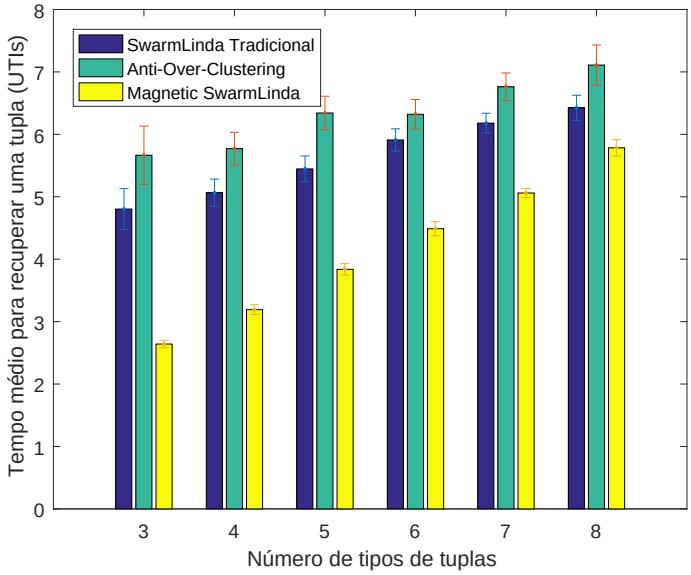

Figura 4.7: Cenário 6 - Diferentes números de tipos de tuplas

As outras abordagens também beneficiam-se de pequenas quantidades de tipos de tuplas, porém a melhora ocorreu de maneira menos expressiva quando comparada a abordagem magnética.

4.3.7 Tempo de viagem

Os resultados do tempo de viagem (apresentados na Figura 4.8) demonstram que o SwarmLinda com *Anti-Over-Clustering* apresenta um desempenho inferior às outras duas abordagens mesmo quando o tempo de processamento é desconsiderado. Isto deve-se ao fato dos *clusters* ficarem dispersos pelo espaço de tuplas na abordagem com a funcionalidade de *Anti-Over-Clustering*.

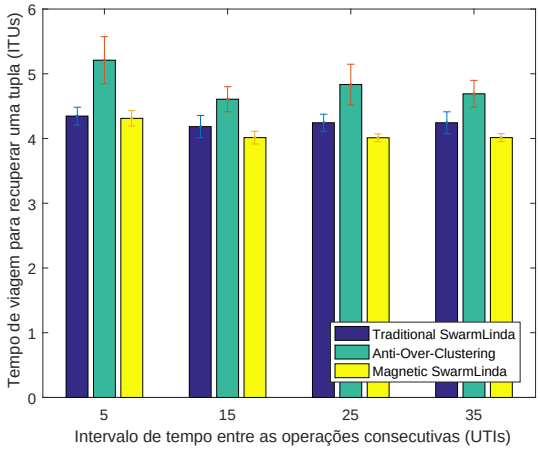

Figura 4.8: Tempo de viagem para recuperação de tupla

Considerando o SwarmLinda com *Anti-Over-Clustering*, quando uma *template-ant* está em um *cluster* formado por tuplas similares ao seu *template*, há uma degradação do desempenho se a formiga não encontrar uma tupla compatível. Isto ocorre devido a necessidade da formiga continuar a exploração do espaço de tuplas até encontrar um *cluster* que contenha uma tupla compatível.

Enquanto isso, o SwarmLinda Tradicional e o modelo proposto não sofrem deste problema de dispersão de *clusters*. Na abordagem tradicional, a formação de *clusters* similares não é promovida. Na abordagem proposta, a concentração excessiva de tuplas é evitada através da formação de *clusters* de nós que contém tuplas similares. Assim, o impacto no desempenho quando uma formiga não encontra uma tupla compatível no primeiro *clusters* visitado é prevenido, uma vez que *clusters* de tuplas similares estão próximos ao nó corrente.

Nas três abordagens analisadas, a operação `in` falha sempre que uma *template-ant* não consegue encontrar uma tupla compatível com seu o *template*. As abordagens SwarmLinda Tracional, Magnetic SwarmLinda e SwarmLinda com *Anti-Over-Clustering* apresentaram no pior caso (considerando todos os cenários), respectivamente, as taxas de erro de 0,7%, 0,9% e 0,4%. É importante notar que a taxa de erro do modelo magnético

é pouco expressiva, ainda que ligeiramente maior, como as taxas de erro das outras abordagens.

4.4 Considerações Finais

Neste capítulo foi apresentado o procedimento de geração de carga utilizado para avaliação do desempenho da abordagem proposta em comparação aos do SwarmLinda Tradicional e do SwarmLinda com a funcionalidade de *Anti-Over-Clustering*. Adicionalmente, foram definidos seis cenários distintos para avaliação do tempo médio para recuperação de uma tupla. Por fim, foram apresentados os resultados, bem como uma análise de cada resultado, referentes aos cenários definidos neste mesmo capítulo.

Capítulo 5

Conclusões

Espaços de tuplas constituem um importante paradigma para o desenvolvimento de aplicações distribuídas, pois permitem uma simplificação na comunicação entre os processos distribuídos. Além disso, os desacoplamentos temporal e espacial propiciados pelo paradigma permitem reduzir a dependência entre os componentes de um sistema, facilitando uma eventual substituição de algum módulo. Com a demanda crescente por aplicações escaláveis, é necessário que as implementações de espaços de tuplas atendam adequadamente às demandas de escalabilidade.

Neste trabalho foi proposta uma abordagem bionspirada, denominada de Magnetic SwarmLinda, que introduz um mecanismo de distribuição de tuplas e, consequentemente, de balanceamento de carga. Nesta abordagem, a sobrecarga dos nós do espaço de tuplas produz um campo magnético virtual que é responsável por perturbar o comportamento dos agentes do espaço de tuplas. Essa pertubação permite a formação de *clusters* de nós que contém tuplas similares, evitando desta maneira a concentração de carga em apenas poucos nós. Além disso, a formação de *clusters* de nós tem por objetivo evitar um impacto negativo na capacidade de exploração do espaço de tuplas pelos agentes, o que provocaria uma degradação no desempenho similar à do SwarmLinda *Anti-Over-Clustering*.

Para a avaliação de desempenho da abordagem proposta optou-se pela construção de um simulador baseado em eventos discretos, que permitiu analisar o desempenho do espaço de tuplas para diferentes cenários de carga. Além disso, a proposta foi avaliada para diferentes perfis de aplicação e topologia do espaço de tuplas. Os resultados de tempo médio para recuperação de tupla do Magnetic SwarmLinda foram comparados com os resultados da implementação Tradicional do SwarmLinda e do SwarmLinda com *Anti-Over-Clustering*, permitindo avaliar a melhora no desempenho propiciada pela distribuição de tuplas baseada em campos magnéticos virtuais.

5.1 Contribuições

Os resultados apresentados na Seção 4.3 demonstram que o desempenho da abordagem proposta é superior às outras abordagens analisadas em diferentes cenários, apresentando um desempenho inferior apenas em cenários onde há uma pequena quantidade de tuplas idênticas disponíveis no espaço de tuplas (como discutido na Seção 4.3.3). Além disso, foi demonstrado na Seção 4.3.5 que a formação de *clusters* de nós que contém tuplas similares introduz um expressivo aperfeiçoamento do desempenho do espaço de tuplas para cenários onde há elevada quantidade de tuplas disponíveis.

Conforme discutido na Seção 4.3.6, há também uma significativa melhora no desempenho para cenários onde a quantidade de nós que compõem o espaço de tuplas é maior que a quantidade de tipos de tuplas, tirando vantagem de *clusters* formados por uma quantidade maior de recursos. Assim, a abordagem proposta demonstra um nível de escalabilidade adequado para sistemas que manipulam elevadas quantidades de dados. Ademais, conforme exposto na Seção 4.3.4, a abordagem proposta não necessidade de uma topologia muito densa, simplificando a construção e gerenciamento da rede utilizada pelo espaço de tuplas.

A formação dos *clusters* ocorre de uma maneira a evitar o aumento de carga em nós com tendência de sobrecarga. Isso deve-se ao fato do comportamento estranho ocorrer na formiga antes desta alcançar um nó que esta próximo da sua capacidade máxima de tuplas. Desta forma, até mesmo o mecanismo de decisão do comportamento estranho não provoca cargas adicionais em nós com elevada carga. Além disso, há uma antecipação do armazenamento da tupla que produz uma redução da carga total do espaço de tuplas.

Por fim, a abordagem proposta apresenta um novo segmento de aplicação para os campos magnéticos virtuais, bem como de estudo sobre espaços de tuplas. Por consequência, permite que avanços tecnológicos envolvendo campos magnéticos virtuais permitam, eventualmente, evoluções em espaços de tuplas bioinspirados que utilizem este conceito.

5.2 Trabalhos Futuros

Uma futura linha de pesquisa da abordagem proposta é relacionada ao desenvolvimento de uma mecanismo para definir dinamicamente o valor da restrição magnética. A determinação deste valor poderia ser feito através de uma análise histórica da relação entre a quantidade de tuplas armazenadas e o nível de ocupação de CPU. Assim, seria possível estimar o ponto em que devido a uma quantidade elevada de tuplas ocorre uma

saturação da capacidade de processamento do nó.

O desenvolvimento de uma heurística para análise de similaridade entre tuplas que adapte-se ao perfil da aplicação é outro trabalho a ser explorado no futuro. Desta forma, seria possível retirar do desenvolvedor a responsabilidade de implementação da função de similaridade, que não está presente no modelo original de Linda. Além disso, simplificaria o processo de substituição de um espaço de tuplas qualquer, que é utilizado por uma determinada aplicação, por um espaço de tuplas bioinspirado.

O impacto negativo das tuplas raras no desempenho da abordagem proposta é uma relevante oportunidade de trabalho futuro. Uma possível abordagem para solucionar este problema seria através da detecção, pelo espaço de tuplas, da utilização de tuplas raras pela aplicação. Desta forma, poderia ocorrer uma habilitação seletiva dos campos magnéticos virtuais no Magnetic SwarmLinda. Quando a utilização de tuplas raras fosse predominante, o espaço de tuplas desabilitaria a funcionalidade de campos magnéticos virtuais e apresentaria um desempenho equivalente ao SwarmLinda Tradicional. Por outro lado, a funcionalidade de campos magnéticos virtuais permaneceria habilitada em cenários com muitas tuplas idênticas. Assim, o espaço de tuplas apresentaria o melhor desempenho entre o SwarmLinda Tradicional e o Magnetic SwarmLinda, independentemente do perfil da aplicação que está empregando o espaço de tuplas.

Além disso, é desejável o desenvolvimento de uma plataforma extensível para implementação e avaliação de espaços de tuplas com suporte à aplicações distribuídas com diversas demandas de comunicação. Desta forma, seria possível avaliar o desempenho de futuras abordagens de espaço de tuplas através de aplicações reais. Além disso, esta plataforma poderia unificar os procedimentos de avaliação de desempenho de espaços de tuplas. Assim, não seria necessário reimplementar todas as abordagens relevantes sempre que um pesquisador desejasse avaliar uma nova abordagem. Isto permitiria otimizar os esforços na elaboração de novas abordagens, uma vez que a plataforma permitiria que cada pesquisa desfrutasse dos esforços despendidos anteriormente por outras pesquisas.

Referências Bibliográficas

ABRAÇADO, L. et al. Magnetic material in head, thorax, and abdomen of solenopsis substituta ants: A ferromagnetic resonance study. *Journal of Magnetic Resonance*, Elsevier, v. 175, n. 2, p. 309–316, 2005.

ANDERSON, J.; MEER, R. V. Magnetic orientation in the fire ant, solenopsis invicta. *Naturwissenschaften*, Springer, v. 80, n. 12, p. 568–570, 1993.

ANGONESE, C. Workload balance in cloud-replicated services. *Revista Tecnologia*, v. 34, n. 1/2, p. 53–62, 2013. Disponível em: ⟨http://dx.doi.org/10.5020/23180730.ano.pi⟩.

ATKINSON, A. Tupleware: A distributed tuple space for cluster computing. In: IEEE. *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008. Ninth International Conference on*. [S.l.], 2008. p. 121–126.

ATKINSON, A. A dynamic, decentralised search algorithm for efficient data retrieval in a distributed tuple space. In: AUSTRALIAN COMPUTER SOCIETY, INC. *Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing-Volume 107*. [S.l.], 2010. p. 21–30.

ATKINSON, A. K. *Tupleware: a distributed tuple space for the development and execution of array-based applications in a cluster computing environment*. Tese (Doutorado) — University of Tasmania, 2010.

BANKS, A. N.; SRYGLEY, R. B. Orientation by magnetic field in leaf-cutter ants, atta colombica (hymenoptera: Formicidae). *Ethology*, Wiley Online Library, v. 109, n. 10, p. 835–846, 2003.

CALSAVARA, A.; LIMA, L. A. de P. Routing based on message attraction. In: IEEE. *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*. [S.l.], 2010. p. 189–194.

CALSAVARA, A.; LIMA, L. A. de P. Scalability of distributed dynamic load balancing mechanisms. In: *ICN 2011, The Tenth International Conference on Networks*. [S.l.: s.n.], 2011. p. 347–352.

CASADEI, M. et al. On the problem of over-clustering in tuple-based coordination systems. In: IEEE. *Self-Adaptive and Self-Organizing Systems, 2007. SASO'07. First International Conference on*. [S.l.], 2007. p. 303–306.

CHATTY, A. et al. Emergent complex behaviors for swarm robotic systems by local rules. In: IEEE. *Robotic Intelligence In Informationally Structured Space (RiiSS), 2011 IEEE Workshop on*. [S.l.], 2011. p. 69–76.

COAN, W. S.; CALSAVARA, A.; LIMA, L. A. P. Roteamento em redes de sensores sem fio baseado no caminho mais forte em energia. In: *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Anais do XVII Workshop de Gerência e Operação de Redes e Serviços*. [S.l.: s.n.], 2012.

GALPERIN, H.; LIMA, L. A. de P.; CALSAVARA, A. Score manager discovery in eigentrust using virtual magnetic fields. In: *The First International Conference on Advanced Communications and Computation*. [S.l.: s.n.], 2011. p. 52–57.

GALPERIN, H.; LIMA, L. A. de P.; CALSAVARA, A. Applying reputation to virtual magnetic networks. In: IEEE. *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*. [S.l.], 2013. p. 831–836.

GELERNTER, D.; BERNSTEIN, A. J. Distributed communication via global buffer. In: ACM. *Proceedings of the first ACM SIGACT-SIGOPS Symposium on Principles of distributed computing*. [S.l.], 1982. p. 10–18.

GERBIER, G. et al. Are ants sensitive to the geometry of tunnel bifurcation? *Animal Cognition*, Springer, v. 11, n. 4, p. 637–642, 2008.

GIBAUD, A.; THOMIN, P. Communications directed by bound types in linda: presentation and formal model. *Parallel and Distributed Systems, IEEE Transactions on*, IEEE, v. 13, n. 8, p. 828–843, 2002.

HAGBERG, A.; SCHULT, D.; SWART, P. Networkx. *URL http://networkx. github. io/index. html*, 2013.

HARI, H. *Tuple space in the cloud*. 2012. ⟨http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Auu%3Adiva-175874⟩. [Online; acessado em 30 de Julho de 2016].

HARTMANN, V. Evolving agent swarms for clustering and sorting. In: ACM. *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. [S.l.], 2005. p. 217–224.

JIANG, Y. et al. Dtuples: A distributed hash table based tuple space service for distributed coordination. In: IEEE. *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*. [S.l.], 2006. p. 101–106.

LIMA, L. A. de P.; CALSAVARA, A. Autonomic application-level message delivery using virtual magnetic fields. *Journal of Network and Systems Management*, Springer, v. 18, n. 1, p. 97–116, 2010.

MARTIN, M.; CHOPARD, B.; ALBUQUERQUE, P. Formation of an ant cemetery: swarm intelligence or statistical accident? *Future Generation Computer Systems*, Elsevier, v. 18, n. 7, p. 951–959, 2002.

MENEZES, R.; TOLKSDORF, R. A new approach to scalable linda-systems based on swarms. In: ACM. *Proceedings of the 2003 ACM Symposium on Applied computing*. [S.l.], 2003. p. 375–379.

MENEZES, R.; WOOD, A. The fading concept in tuple-space systems. In: ACM. *Proceedings of the 2006 ACM Symposium on Applied computing*. [S.l.], 2006. p. 440–444.

MICHELON, G. A. et al. Weighted centrality by potential for replica relocation in manets. *International Journal of Computer and Electrical Engineering*, IACSIT Press, v. 8, n. 2, p. 151, 2016. Disponível em: ⟨http://dx.doi.org/10.17706/IJCEE.2016.8.2.151-160⟩.

MICHELON, G. A. et al. A strategy for data replication in mobile ad hoc networks. In: IEEE. *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*. [S.l.], 2014. p. 486–489.

MONMARCH, N. On data clustering with artificial ants. In: *Data Mining with Evolutionary Algorithms: Research Directions–Papers from the AAAI Workshop*. [S.l.: s.n.], 1999. p. 23–26.

NAVLAKHA, S.; BAR-JOSEPH, Z. Distributed information processing in biological and computational systems. *Communications of the ACM*, ACM, v. 58, n. 1, p. 94–102, 2014.

OLIVEIRA, J. A. de et al. Anycasting in dtns using virtual magnetic fields. In: IEEE. *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*. [S.l.], 2014. p. 505–506.

OLIVEIRA, J. F. de et al. Ant antennae: are they sites for magnetoreception? *Journal of The Royal Society Interface*, The Royal Society, v. 7, n. 42, p. 143–152, 2010.

PARPINELLI, R. S.; LOPES, H. S.; FREITAS, A. A. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 6, n. 4, p. 321–332, 2002.

PICCO, G. P.; MURPHY, A. L.; ROMAN, G.-C. Lime: Linda meets mobility. In: ACM. *Proceedings of the 21st international conference on Software engineering*. [S.l.], 1999. p. 368–377.

ROSSUM, G. van; EBY, P. J. Pep 342 – coroutines via enhanced generators. *Python Developer's Guide*, 2005.

ROWSTRON, A. I.; WOOD, A. M. Bonita: A set of tuple space primitives for distributed coordination. In: IEEE. *System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on*. [S.l.], 1997. v. 1, p. 379–388.

VASSEV, E. et al. Swarm technology at nasa: building resilient systems. *IT Professional Magazine*, IEEE Computer Society, v. 14, n. 2, p. 36, 2012.

WAJNBERG, E. et al. Magnetoreception in eusocial insects: an update. *Journal of the Royal Society Interface*, The Royal Society, p. rsif20090526, 2010.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of small-world networks. *Nature*, Nature Publishing Group, v. 393, n. 6684, p. 440–442, 1998.