**UNIVERSIDADE DE BRASÍLIA**
**FACULDADE DE TECNOLOGIA**
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

# ANÁLISE DE TÉCNICAS BASEADAS EM METAHEURÍSTICAS E DOMINAÇÃO DE GRAFOS PARA *CLUSTERING* EM REDES *AD HOC*

**HELTON FABIANO GARCIA**

**ORIENTADOR: PAULO ROBERTO DE LIRA GONDIM**

**DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA**

**PUBLICAÇÃO: ENE.DM 270/06**

**UNIVERSIDADE DE BRASÍLIA**

**FACULDADE DE TECNOLOGIA**

**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

# ANÁLISE DE TÉCNICAS BASEADAS EM METAHEURÍSTICAS E DOMINAÇÃO DE GRAFOS PARA *CLUSTERING* EM REDES *AD HOC*

**HELTON FABIANO GARCIA**

**DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA.**

**APROVADA POR:**

_____ _____

**PAULO ROBERTO DE LIRA GONDIM, DOUTOR,ENE/UNB (ORIENTADOR)**

_____

**JACIR LUIZ BORDIM, PhD, CIC/UNB (EXAMINADOR INTERNO)**

_____

**MARIA EMÍLIA TELLES WALTER, DOUTORA, CIC/UNB (EXAMINADOR EXTERNO)**

**BRASÍLIA/DF, 18 DE AGOSTO DE 2006.**

## FICHA CATALOGRÁFICA

GARCIA, HELTON FABIANO

Análise de Técnicas Baseadas em Metaheurísticas e Dominação de Grafos para Clustering em Redes Ad Hoc [Distrito Federal] 2006.

xx, 197 p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2006). Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

| | |
|---|---|
| 1. Metaheurísticas | 2. Otimização Combinatória |
| 3. Clustering | 4. Redes Ad Hoc |
| I. ENE/FT/UnB | II. Título (série) |


## REFERÊNCIA BIBLIOGRÁFICA

GARCIA, H.F. (2006). Análise de Técnicas Baseadas em Metaheurísticas e Dominação de Grafos para Clustering em Redes Ad Hoc. Dissertação de Mestrado em Engenharia Elétrica, Publicação ENE.DM-270/2006, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 197 p.

## CESSÃO DE DIREITOS

AUTOR: Helton Fabiano Garcia.
TÍTULO: Análise de Técnicas Baseadas em Metaheurísticas e Dominação de Grafos para Clustering em Redes Ad Hoc.

GRAU: Mestre ANO: 2006



Helton Fabiano Garcia
SQN 103 bl A apt 506 – Asa Norte
70732-010 – Brasília - DF – Brasil.

À minha mãe pela dedicação e

abnegação em tudo o que faz

Ao meu pai pela perseverança

e vontade de ser um vencedor

Em especial, à Carine pelo

amor, carinho e paciência

nas horas mais difíceis

# AGRADECIMENTOS

Agradeço a Deus pela saúde e oportunidade de chegar até aqui.

Ao Prof. DSc. Paulo Gondim pela orientação, apoio, paciência, constante confiança na minha capacidade e principalmente por entender os fatores humanos aos quais podem influenciar na realização deste trabalho.

À Prof ª DSc. Maria Emília pelo apoio oportuno, disponibilidade e orientações objetivas e também por transmitir a segurança de estarmos no caminho certo.

À minha família, pela força, paciência, amor e a certeza de estarem sempre torcendo por mim e igualmente, por entender os momentos de minha ausência.

À grande surpresa Carine, por ter surgido na minha vida e dado equilíbrio, amor e motivação para seguir em frente.

Ao amigo Carlo Kleber pelas sugestões sempre criteriosas, esclarecedoras e construtivas e pela preocupação com a qualidade do meu trabalho.

Ao amigo Alexandre Zaghetto por estar sempre disposto a me ajudar e pelas discussões oportunas, sempre em clima de respeito e amizade.

Ao grande amigo Junier Amorim pelo apoio e sugestões na fase experimental.

À "grande mentora de transformações" Marta, meus sinceros agradecimentos.

Ao Centro de Desenvolvimento de Sistemas, em particular ao Exmo. Sr. Gen Jaldemar, por aprovar a minha iniciativa de ingressar neste curso de mestrado, no período de 2004-2005.

À Faculdade FAJESU, particularmente ao amigo Prof. João Batista, pelo constante apoio, preocupação e reconhecimento da importância deste trabalho.

De forma especial, sou grato à minha "família de Brasília". Aos grandes amigos que estiveram sempre presentes, nas horas de alegria para sorrir e compartilhar bons momentos e na tristeza para dar o ombro amigo e resgatar o equilíbrio a quem necessita.

A todos que, de alguma forma, contribuíram para a realização deste trabalho.

"*A persistência é o caminho do êxito*."

*Charles Chaplin (1889-1977)*

# SUMÁRIO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABELAS

# ÍNDICE DE SÍMBOLOS E ABREVIATURAS

| | |
|---|---|
| $\Delta_{\psi}(t)$ | Variação da Diferença de Fluxo $\psi(t)-\psi(t-1)$ |
| $\psi(t)$ | Diferença de fluxo $\psi_{intra}(t)-\psi_{inter}(t)$ |
| $\Gamma(v)$ | Estado de *v* |
| $P(v)$ | Energia de *v* |
| $\pi(v)$ | Falha de *v* |
| $\zeta_v$ | Disponibilidade de *v* |
| $\psi_{inter}$ | Fluxo de Dados *Inter-cluster* |
| $\psi_{intra}$ | Fluxo de Dados *Intra-cluster* |
| $\tau$ | *Threshold* |
| $\rho(h(v))$ | Potência do sinal do *clusterhead h(v)* recebido por *v* |
| $<DS>_w$ | *Weakly Induced dominant Set* |
| $<f>_{MST}$ | Fragmento de um *MST* |
| $<S>_w$ | *Weakly Induced Subgraph* |
| ACO | *Ant Colony Optimization* |
| Ativo- | Ativo com economia de energia |
| Ativo* | Estados ativo e ativo com economia de energia |
| BL | Busca Local |
| BLEP | Busca Local por Extremidades de Partida |
| BSAMM | *Boundless Simulaton Area Mobility Model* |
| C(v) | *Cluster* de *v* |
| CDS(G) | *Connected Dominant Set* |
| CF | *Core Fragment* |
| $C_i$(v) | *Cluster i* de *v* |
| CN | *Core Nodes* |
| CV | Coeficiente de Variação |
| DS | *Dominant Set* |
| GA | *Genetic Algorithm* |
| GAdHoc | *Genetic Algorithm for Ad Hoc Networks* |
| GHS | *Gallager-Humblet-Spira Algorithm* |
| GLS | *Guided Local Search* |
| GRASP | *Greedy Randomized Adaptive Search Procedure* |
| h(v) | *Clusterhead do nó v* |
| HDA | *Highest-Degree Algorithm* |
| HGA | *Hybrid Genetic Algorithm* |
| IDS(G) | *Independent Dominant Set* |
| ILS | *Iterated Local Search* |
| LIA | *Lowest-ID Algorithm* |
| LT | Lista Tabu |
| MA | *Memetic Algoritm* |
| MTSO | *Mobile Telecommunications Switching Office* |
| MH | Metaheurística |
| MHAdHoc | Metaheurísticas em Redes *Ad Hoc* |
| MOE | *Minimum Outgoing Edge* |

| | |
|---|---|
| MST | *Minimum Spanning Tree* |
| MT | Movimento *Tabu* |
| N(v) | *Closed Neighborhood of v* |
| NBA | *Neighborhood Based Algorithms* |
| $N_{closed}(v)$ | *Closed Neighborhood of v* |
| $N_{open}(v)$ | *Open Neighborhood of v* |
| NReaf | Número de Reafiliações |
| $NReaf_c$ | Número de Reafiliações com rearranjo de topologia |
| $NReaf_s$ | Número de Reafiliações sem rearranjo de topologia |
| OE | *Outgoing Edge* |
| $OE(C_i(w))$ | Extremidades de Partida a partir do nó w do Cluster i |
| OEL | *Outgoing Edge List* |
| PBA | *Population Based Algorithms* |
| PR | *Path Relinking* |
| RCL | *Restricted Candidate List* |
| RDP | Reafiliação por Diferença de Potência |
| RMBA | *Random Move Based Algorithms* |
| RPGM | *Reference Point Mobility Model* |
| RQS | Reafiliação por Queda de Sinal |
| RWaMM | *Random Walk Mobility Model* |
| RWyMM | *Random Waypoint Mobility Model* |
| s | Solução |
| s* | Melhor Solução encontrada (TS) |
| $s_0$ | Solução atual |
| SA | *Simulated Annealing* |
| SAdHoc | *Simulated Annealing for Ad Hoc Networks* |
| SAMAX | Número máximo de iterações para o SA |
| SJ | *Simulated Jumping* |
| SLEP | Seleção Local por Extremidades de Partida |
| SRMM | *Smooth Random Mobility Model* |
| SS | *Scatter Search* |
| ST | *Status Tabu* |
| $T_c$ | Temperatura de Congelamento |
| $T_{exec}$ | Tempo de Execução |
| Tg | Granulosidade de $T_{exec}$ |
| TS | *Tabu Search* |
| TSAdHoc | *Tabu Search for Ad HocNetworks* |
| TSMAX | Número máximo de iterações para o TS |
| ut | Unidade de tempo |
| VA | Vizinho Aleatório |
| $VT(s_0)$ | Vizinho *Tabu* de $s_0$ |
| WCA | *Weighted Clustering Algorithm* |
| WCDS(G) | *Weakly Connected Dominant Set* |
| WPBA | *Weighted and Penalty Based Algorithms* |
| ZCA | *Zonal Clustering Algorithm* |

# RESUMO


**ANÁLISE DE TÉCNICAS BASEADAS EM METAHEURÍSTICAS E DOMINAÇÃO DE GRAFOS PARA *CLUSTERING* EM REDES *AD HOC***

**Autor : Helton Fabiano Garcia**
**Orientador: Paulo Roberto de Lira Gondim**
**Programa de Pós-Graduação em Engenharia Elétrica**
**Brasília, Agosto de 2006.**

As redes *ad hoc* são caracterizadas pela ausência de infra-estrutura de comunicação. Uma forma de comunicação entre os nós, assim como a manutenção de mudanças de conexão podem utilizar uma estrutura hierárquica baseada em *clusters* [EPH87]. Um *cluster* agrupa dinamicamente um conjunto de nós em torno de um nó central, responsável pelo roteamento de dados, chamado de *clusterhead* [CHA00]. Os demais membros deste *cluster* são denominados *clusternodes*. O conjunto de *clusterheads* de uma rede é chamado de *dominant set*. Esta estrutura forma um *backbone virtual* [CHE02].

O problema do particionamento de uma rede em *clusters* é NP-completo [REE93], fazendo com que a busca por uma solução ótima para a organização em *clusters* de uma rede *ad hoc* com topologia móvel seja um desafio. Uma estratégia para a resolução deste problema é a aplicação de técnicas baseadas em metaheurísticas. Desta forma, obter uma "boa" solução, dentro de um cenário com domínio de busca limitado, mostra-se conveniente em boa parte dos casos [REE93].

Este trabalho usa técnicas baseadas em metaheurísticas, algoritmos genéticos [HOL75], *simulated annealing* [KIR83] e busca tabu [GLO89] na proposição de algoritmos para o particionamento em *clusters*, levando em consideração o grau de mobilidade da rede, reafiliações, transmissão de dados, disponibilidade, energia e ciclo de vida. Basicamente, os algoritmos buscam a minimização do fluxo de dados *inter-clusters*. Dados os *clusters* já formados, determinam-se os *clusterheads.* Apresentam-se, também, simulações comparando os algoritmos propostos entre si, assim como com outras técnicas de particionamento.



**Palavras-chave** – metaheurísticas, *clustering* em redes *ad hoc*, algoritmos genéticos, *simulated annealing*, busca tabu.

# ABSTRACT


**METAHEURISTICS AND GRAPH DOMINATION TECHNIQUES ANALISYS FOR CLUSTERING IN WIRELESS MOBILE AD HOC NETWORKS**

**Author : Helton Fabiano Garcia**
**Supervisor: Paulo Roberto de Lira Gondim**
**Programa de Pós-Graduação em Engenharia Elétrica**
**Brasília, August of 2006.**

*Wireless ad hoc networks are characterized for a lack of fixed communication structure. One of the strategies for communications between nodes and the maintenance of connection changes is to adopt a hierarchy structure based in clusters* [EPH87]. *A cluster dinamically gathers a set of nodes around a local coordinator of data transmission, called clusterhead. All other members of this cluster are called clusternodes or members. The set of clusterheads on a network is called dominant set* [CHA00]. *This structure forms a virtual backbone* [CHE02].

*The clustering partitioning in wireless ad hoc networks is a NP-complete problem* [REE93], *leading to research for an optimal solution for a mobile generic topology as a challenge. An approach to solve this problem is applying metaheuristics techniques. So, to obtain a "good" solution within a scenario with a limited search range proves convenient for several cases and the use of metaheuristics is a powerful instrument to do so* [REE93].

*This work presents a study about metaheuristics algorithms, such as genetic algorithms* [HOL75], *simulated annealing* [KIR83], *and tabu search* [GLO89] *to determinate cluster partitioning in a generic wireless mobile ad hoc network, taking in consideration mobility models, data transmission, availability, energy and life cycle of nodes. Basically, the proposed models are based on inter-clusters data flow minimization strategy. In a second phase, clusterheads are determined, once clusters has already formed. Simulation results are presented to compare these techniques and some existing models to prove results.*



***Keywords*** *- metaheuristics, clustering in wireless ad hoc networks, genetic algorithms, simulated annealing, tabu search.*

# 1-INTRODUÇÃO

## 1.1. CONTEXTO

Uma rede *ad hoc* não apresenta infra-estrutura fixa de comunicação [RFC2501], sendo indicada para situações em que não é adequado instalar uma rede cabeada, tais como situações de resgate e salvamento, estudantes e professores em uma sala de aula e campos de batalha.

Os dispositivos móveis que as constituem são capazes de se comunicarem diretamente entre si ou através de múltiplos saltos, de forma a compensar a inexistência da infra-estrutura fixa.

Uma das estratégias para a comunicação entre os nós e a manutenção de mudanças de conexão é a adoção de uma estrutura hierárquica baseada em *clusters* [EPH87], [GER95], [RAM98]. Cada *cluster* contém um nó responsável pela coordenação local da transmissão de informações, chamado de *clusterhead* ou líder. Os demais membros deste cluster são denominados *clusternodes* ou membros. O conjunto de *clusterheads* de uma rede é chamado de *dominant set* [CHA00]. Este conceito difere do modelo de célula [GER97], adotado em uma rede celular, por não possuir diferenciação dos seus constituintes sob o ponto de vista do *hardware* e por haver seleção dinâmica de *clusterheads* de acordo com as mudanças da topologia da rede, devido, por exemplo, ao deslocamento dos seus nós.

Por outro lado, há problemas, tais como, um nó poderá sofrer sobrecarga se permanecer como *clusterhead* por um grande período de tempo ou se tiver que suportar uma grande quantidade de dados durante o período de tempo em que é líder [REE93]. Outro problema: em uma arquitetura hierárquica baseada em *clusters* o problema do particionamento em *clusters* é NP-completo [REE93], sendo assim, não é possível obter uma solução ótima em tempo polinomial. Desta forma, obter uma "boa" solução, dentro de um cenário com domínio de busca limitado, mostra-se conveniente em boa parte dos casos e o emprego de metaheurísticas é um poderoso instrumento para isso [REE93], [GOL93].

Esta dissertação realiza análise sobre técnicas baseadas em metaheurísticas para resolver o problema do particionamento de *clusters* em redes *ad hoc*. Em particular, as seguintes técnicas: algoritmos genéticos [HOL75], *simulated annealing* [KIR83] e busca tabu [GLO89], [GLO90], [GLO97].

Apresentar-se-ão, ainda, alguns modelos de *clustering* para redes *ad hoc* [EPH87], [GER95], [CHA00], [CHE03b], para comparação de desempenho com as técnicas propostas.

## 1.2. OBJETIVOS

A presente dissertação tem como objetivo principal utilizar técnicas baseadas em metaheurísticas para propor algoritmos de particionamento de *clusters* em redes *ad hoc*.

Os objetivos específicos desta dissertação incluem:

- Propor algoritmos de particionamento em *clusters* baseado na aplicação de diversas técnicas baseadas em metaheurísticas: *simulated annealing*, busca tabu, algoritmos genéticos para a determinação do particionamento de uma rede *ad hoc* em *clusters* com certo grau de mobilidade, visando à minimização do fluxo de dados *inter-clusters*;
- Propor uma técnica para a determinação de *dominant set* para o roteamento dos dados entre *clusters*, considerando os *clusters* já formados;
- Implementar os algoritmos para simulações;
- Utilizar métodos estatísticos para tratamento dos resultados obtidos;
- Comparar os resultados obtidos entre as técnicas baseadas metaheurísticas adotadas;
- Comparar os resultados obtidos dos algoritmos propostos com algoritmos de *clustering* já existentes, com o objetivo de validar o proposto por este trabalho.

## 1.3. ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho está organizado da seguinte forma:

O capítulo 2 apresenta um breve histórico sobre a origem das redes *ad hoc*. São apresentados conceitos sobre a evolução de modelos propostos segundo uma taxonomia sobre tipos de algoritmos de *clustering* [CHE03], características do *dominant set,* além de alguns conceitos e propriedades importantes sobre *clustering* e alguns algoritmos de *clustering* para redes *ad hoc*, particularmente, os envolvidos neste trabalho: *Lowest-ID algorithm* [EPH87], *Highest-Degree Algorithm* [GER95], *Weighted Clustering Algorithm* (WCA) [CHA00] e *Zonal Clustering Algorithm* [CHE03b].

O capítulo 3 apresenta diversas técnicas de metaheurísticas dentro da classificação proposta por [MIL04]: *Random Move Based Algorithms* (RMBA), *Population Based Algorithms* (PBA), *Neighborhood Based Algorithms* (NBA) e *Weighted and Penalty Based Algorithms* (WPBA). Apresentar-se-á cada um dos algoritmos, suas propriedades e uma breve crítica, com vantagens, desvantagens, aplicações e parâmetros envolvidos em cada técnica.

O capítulo 4 apresenta alguns tópicos sobre redes *ad hoc*: mobilidade, regras de borda, disponibilidade e métricas de desempenho.

O capítulo 5 aborda os algoritmos propostos no presente estudo. Far-se-á um aprofundamento nas técnicas de metaheurísticas empregadas e o seu emprego na solução de problemas de particionamento de *clusters* em redes *ad hoc*. Em particular, usamos as seguintes técnicas: algoritmos genéticos [HOL75], *simulated annealing* [KIR83] e busca tabu [GLO89], [GLO90], [GLO97]. Apresentar-se-ão parâmetros experimentais de cada modelo e para o modelo baseado na busca tabu desenvolvemos uma análise teórica da complexidade de tempo e espaço.

O capítulo 6 apresenta estudos de caso, isto é, simulações comparando o desempenho das técnicas propostas com outros modelos de *clustering* para redes *ad hoc*, [EPH87], [GER95], [CHA00], [CHE03b], a fim de analisar vantagens e desvantagens, dentro do desafio de minimizar a comunicação *inter-clusters* [GAR05]. O desempenho será mensurado a partir o resultado final de cada técnica, isto é, o valor da Função Objetivo $F_o$ alcançado, para medir a qualidade da técnica, e o tempo necessário para se obtê-la,

ou seja, o Tempo de Execução ($T_{exec}$) em diferentes cenários, desde uma topologia mais simples até uma rede mais complexa. É proposto também um melhoramento no processo de seleção de um vizinho *s* de $s_0$ do algoritmo *Tabu Search* acoplado ao *TSAdHoc*, chamado TS+SLEP. O objetivo é melhorar o desempenho da técnica., com base no conceito de Extremidades de Partida ($OE(C_i)$) de um *cluster* $C_i$, $C_i \in C$ , selecionado de forma aleatória em distribuição uniforme.

No capítulo 7 são apresentadas as conclusões: do trabalho realizado e sugestões de propostas para trabalhos de pesquisa futuros, além de publicações referentes a este trabalho.

O Apêndice A apresenta alguns gráficos mostrando a convergência para soluções quase-ótimas das técnicas propostas no presente estudo, em diferentes execuções dos modelos durante a fase de experimentação.

O Apêndice B apresenta um exemplo simplificado de simulações com G=(V,E), $V = \{n_0, n_1, \ldots, n_8\}$ , disposição determinística em terreno 100x100, $T_{exec}$=10, K=3.

O Apêndice C apresenta algumas topologias como resultados de simulação utilizados no processo de experimentação no Capítulo referente aos Estudos de Caso. É possível visualizar em cada solução, *clusters*, *clusterheads*, *dominators* e *CDS(G)*.

O Apêndice D apresenta alguns cenários de simulação utilizados no processo de experimentação no Capítulo referente aos Estudos de Caso.

O Apêndice E apresenta alguns filtros usando para manipulação de arquivos de *logs* (*Shell Script/Awk*/ER).

O Apêndice F apresenta *Shell Script/Awk* para plotagem de gráficos, gerados utilizando o *Gnuplot*, v4.0 for *Unix/Linux* [GNU06], a partir das informações armazenadas em *logs* de cada execução. e automatização de tarefas.

O Apêndice G apresentará uma análise mais detalhada das complexidades de tempo e espaço realizadas no Capítulo 6.

# 2-CLUSTERING

## 2.1. INTRODUÇÃO

A primeira idéia relacionada a redes *ad hoc* foi concebida pelo DARPA há 33 anos atrás com o projeto de natureza militar PRNet (*Packet Radio Network*) [KAH78]. Este projeto motivou novos trabalhos para a implementação de redes *ad hoc*: SURAN (*Survivable Radio Networks*), TI (*Tactical Internet*), GloMo (*Global Mobile Information Systems*) [FRE01], WAMIS (*Wireless Adaptive Mobile Information Systems*) e Glomo ARPA [GER96].

Uma das estratégias para a comunicação entre os nós e a manutenção de mudanças de conexão é a adoção de uma estrutura hierárquica baseada em *clusters* [EPH87], [GER95], [RAM98].

*Clustering* é o particionamento de nós em *clusters* e tem como objetivo facilitar o gerenciamento de nós e melhorar o desempenho na comunicação de dados de uma população de unidades móveis, agregando a distribuição de dados de forma hierárquica, pela partição da população de nós de uma rede em vários *clusters* [GER97]. Em conseqüência, este sistema torna-se mais facilmente escalável.

Este conceito difere do modelo de célula [GER97], adotado em uma rede celular, por não possuir diferenciação dos seus constituintes sob o ponto de vista do *hardware* e por haver seleção dinâmica de *clusterheads* de acordo com as mudanças da topologia da rede, devido, por exemplo, ao deslocamento dos seus nós.

Os dispositivos móveis que constituem a redes *ad hoc* são capazes de se comunicarem diretamente entre si ou através de múltiplos saltos, de forma a compensar a inexistência da infra-estrutura fixa. A adoção deste modelo hierárquico pode ser visto também em outros sistemas, com benefícios semelhantes, como na estrutura de endereçamento em redes cabeadas.

Ephremides iniciou os primeiros estudos sobre a organização de uma rede *ad hoc* em *clusters* [EPH87]. Uma topologia de uma rede pode ser representada utilizando um

grafo G=(V,E), tal que cada *cluster* seria formado por um nó *v* líder ou *clusterhead*, responsável pelo roteamento de dados dentro do *cluster* e pelos nós dentro de uma distância *k* ou "vizinhança de distância-k" *N(v),* chamados de *clusternodes* ou membros. O conjunto de *clusterheads* de *G* é denominado *dominant set*.

## 2.2. DEFINIÇÕES

Definimos agora conceitos importantes relacionadas a *clustering*:

- Processo de *Clustering,*
- *Closed Neighborhood* $N_{closed}(v)$ , $N(v)$ ,
- *Open Neighborhood* $N_{open}(v)$ ,
- *Dominant Set* $DS(G)$ ,
- *Independent Dominant Set* $IDS(G)$ ,
- *Conected Dominant Set* $CDS(G)$ ,
- *Weakly Induced Subgraph* $\langle S \rangle_w$ ,
- *Weakly Connected Dominant Set* $WCDS(G)$ ,
- Fragmento de um MST $\langle f \rangle_{MST}$ ,
- *Outgoing Edge* $OE(\langle f \rangle_{MST})$ ,
- *Minimum Outgoing Edge* $MOE(\langle f \rangle_{MST})$ ,
- Core de um Fragmento $CF\langle f \rangle_{MST}$ ,
- Core Nodes $CN(CF\langle f \rangle_{MST})$ .

### 2.2.1. Processo de Clustering

**Definição:** Seja um grafo não-direcionado $G=(V,E)$ , mapeando uma topologia de uma rede *ad hoc*, onde *V* representa o conjunto de nós da rede e *E* representa uma comunicação passante entre dois vértices. Um **processo de *clustering*** é a divisão de *V* conjuntos disjuntos ou não, $V_0, V_1, V_2, \ldots, V_{k-1}$ , onde $V=U_{i=0}^{k-1} V_i$ , tal que cada $V_i$ é um subgrafo conexo de *G* [CHE03].

### 2.2.2. Closed Neighborhood $N_{closed}(v)$

**Definição:** Seja um grafo não-direcionado e $v \in V$ , dizemos que $N_{closed}(S)$ ou $N(S)$ é um subconjunto $S \subseteq V$ de vértices adjacentes a *v* dentro de uma distância-k de *v*, incluindo o próprio nó *v*, dado por, $N(S) = U_{v \in S} N[v]$ [WES01], [CHE03]. Pode ser representada simplesmente por $N(v)$ .

### 2.2.3. Open Neighborhood $N_{open}(v)$

**Definição:** Seja um grafo não-direcionado e $v \in V$ . Dizemos que $N_{open}(v)$ é um subconjunto $S \subseteq V$ de vértices adjacentes a *v* dentro de uma distância-k de *v*, exclusive *v*, não incluindo *v,* i.e, $N_{open}(v) = N(v) \setminus v$ [WES01], [CHE03].

### 2.2.4. Dominant Set $DS(G)$

**Definição:** Seja um grafo não-direcionado e $v \in V$ . Dizemos que $DS(G)$ é um subconjunto de *V,* tal que qualquer *v* ou pertence a *DS(G)* ou é adjacente a algum vértice dele, isto é, $DS(G) \text{ é dominante} \Leftrightarrow \forall v \in V, v \in DS(G) \vee \exists v_1 \in DS(G) | adj(v, v_1)$ [WES01], [CHE03].

No presente trabalho, utilizaremos uma variante, devido à sua adequação ao modelo proposto. O *dominant set* é o conjunto de *clusterheads* [CHA00], [CHA02], [TUR02].

### 2.2.5. Independent Dominant Set $IDS(G)$

**Definição:** O conjunto dominante de G é dito independente se não houver dois vértices pertencentes a $DS(G)$ ligados diretamente, isto é , $IDS(G) \text{ é dominante} \Leftrightarrow \forall v_1, v_2 \in DS(G), \exists ! v_3 \in V | adj(v_1, v_3), adj(v_3, v_2)$ [CHE03].

Os exemplos abaixo ilustram a definição. Na Figura1. Os nós em preto representam os *clusterheads* de $IDS(G)$ . Os nós em azul são *clusternodes*. Na Figura 2, temos uma representação similar, mas onde o $DS(G)$ não é um $IDS(G)$ .

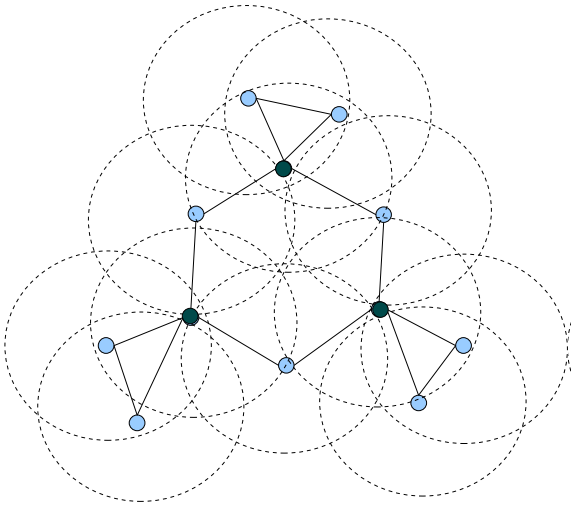

*Figura1: G com um Independent Dominant Set*

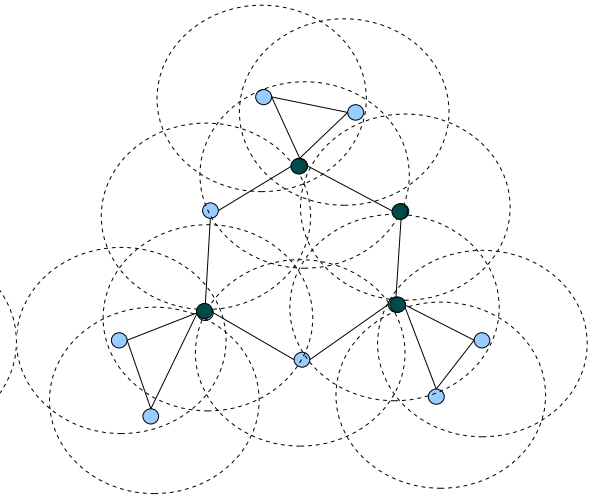

*Figura 2: DS(G) não é Independent Dominant Set*

### 2.2.6. Connected Dominant Set $CDS(G)$

**Definição:** O conjunto dominante de G, $DS(G) \subseteq V$ , é dito conexo se o subgrafo resultante de $DS(G)$ , $\langle DS(G) \rangle$ for conexo, ou seja, $CDS(G)\, é\, dominante \Leftrightarrow \langle DS(G) \rangle \, é\, conexo$ [CHE03].

Os exemplos abaixo ilustram a definição. Na Figura 3, os nós em preto representam os *clusterheads* de $CDS(G)$ . Os nós em azul são *clusternodes*. Na Figura 4, temos uma representação similar, mas em que $DS(G)$ não é $CDS(G)$ .

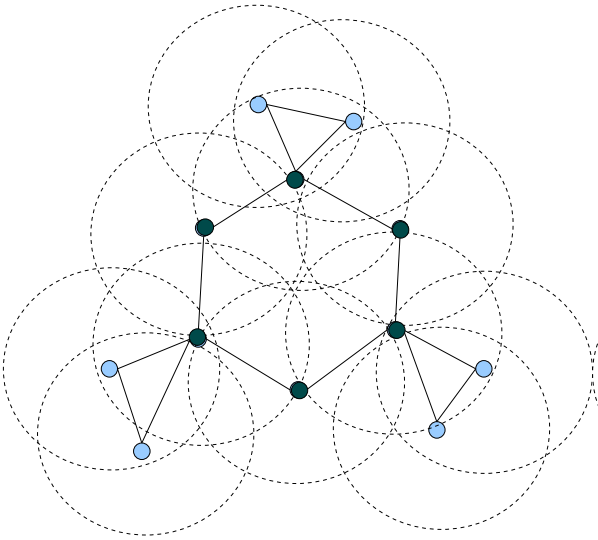

*Figura 3: G=(V,E) com um Connected Dominant Set*

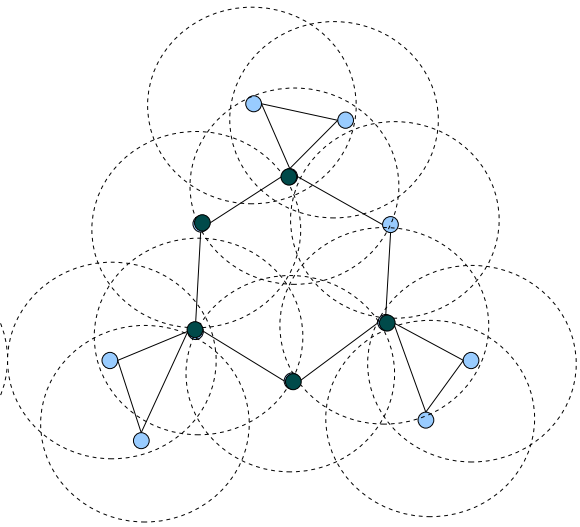

*Figura 4: DS(G) não é Connected Dominant Set*

### 2.2.7. Weakly Induced Subgraph $\langle S \rangle_w$

**Definição:** Para cada subconjunto $S \subseteq V$ , um subgrafo fracamente induzido $\langle S \rangle_w$ contém:

- Os vértices de *S;*
- Seus vizinhos; e
- Todas as arestas de *G* que contêm pelo menos uma ligação a um vértice de *S*, isto é, $\langle S \rangle_w = (N(S); E \cap (N_{open}(S) X S))$ [CHE03].

### 2.2.8. Weakly Connected Dominant Set $WCDS(G)$

**Definição:** O conjunto dominante de G é dito fracamente conexo, se o subgrafo de S, $\langle S \rangle$ , for fracamente induzido e S for dominante, isto é, $WCDS(G)\, é\, dominante \Leftrightarrow \langle S \rangle_w é\, conexo \wedge S\, é\, dominante$ [CHE03].

Os exemplos abaixo ilustram as últimas definições. Na Figura5, os nós em azul são *clusternodes* e os nós em preto, representam os *clusterheads* formam $S \subseteq V$ . Na Figura 6, as arestas de G com pelo menos um vértice em S são mostrados com linha forte. Este subgrafo é fracamente induzido $\langle S \rangle_w$ . Como $\langle S \rangle_w$ é conexo e $S$ é dominante, então $DS(G)$ é fracamente conexo $(WCDS(G))$ .

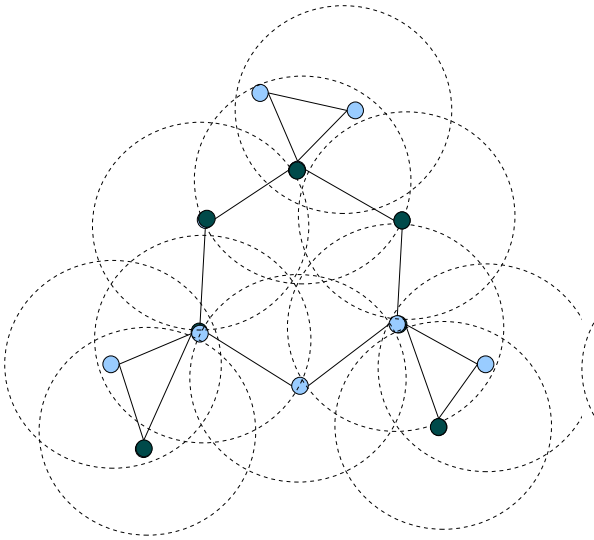

*Figura5: Exemplo de conjunto dominante fracamente conexo - WCDS(G)*

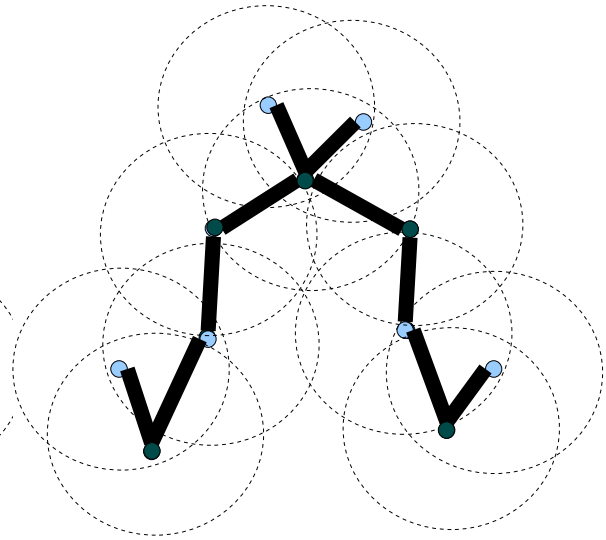

*Figura 6: Subgrafo fracamente induzido e conexo de G*

A mesma situação é exemplificada na Figura 7, agora com *clusterheads* independentes. O correspondente subgrafo fracamente induzido $\langle S \rangle_w$ do conjunto dominante é mostrado na Figura 8. Note que $\langle S \rangle_w$ é também conexo. Logo, $DS(G)$ é fracamente conexo $(WCDS(G))$ .

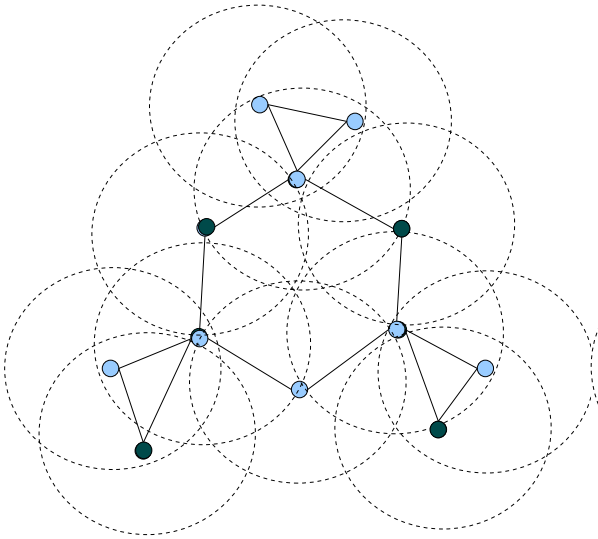

*Figura 7: Exemplo de conjunto dominante fracamente conexo - WCDS(G)*

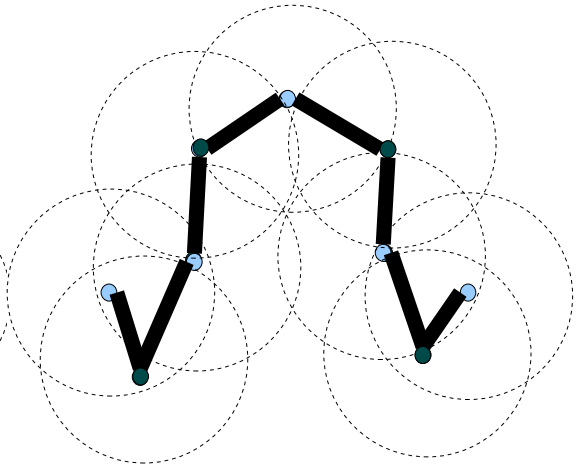

*Figura 8: Subgrafo fracamente induzido e conexo de G*

Por fim, um exemplo de um conjunto dominante não fracamente conexo é visto na Figura 9, pois o subgrafo fracamente induzido $\langle S \rangle_w$ do conjunto dominante é não conexo, conforme pode ser visto na Figura 10. Logo, $DS(G)$ não é fracamente conexo $(WCDS(G))$ .

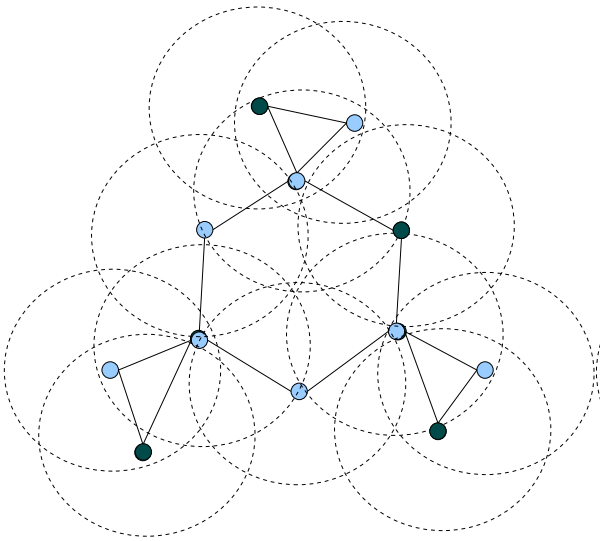

*Figura 9: Exemplo de conjunto dominante não fracamente conexo - WCDS(G)*

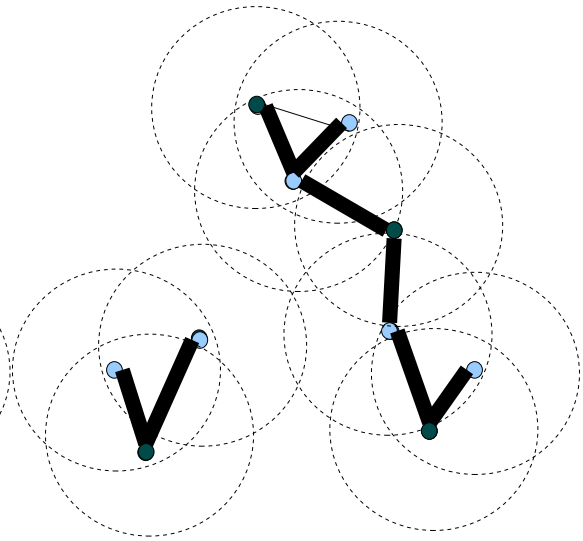

*Figura 10: Subgrafo fracamente induzido e desconexo de G*

### 2.2.9. Fragmento de uma Minimum Spanning Tree $\langle f \rangle_{MST}$

**Definição:** É uma subárvore formada por um conjunto conexo de nós e arestas de uma árvore geradora mínima. A definição de *Minimum Spanning Tree* é encontrada em [GAL83].

### 2.2.10. Outgoing Edge $OE(\langle f \rangle_{MST})$

**Definição:** Seja um grafo não-direcionado *G* e um fragmento de uma MST $\langle f \rangle_{MST}$ de *G*. Dizemos que $OE$ é uma aresta de *G* com exatamente uma extremidade no fragmento considerado , isto é, $G=(V,E), \langle f \rangle_{MST} \in G, v \in \overline{f}, v' \in f | OE(\langle f \rangle) \Leftrightarrow \overline{v, v'}$ É importante observar que uma *OE* de uma árvore geradora mínima não pode formar ciclos. Daí a condição restritiva em que apenas um vértice pertence a $\langle f \rangle_{MST}$ . É também chamada de Extremidade de Partida [GAL83].

### 2.2.11. Minimum Outgoing Edge $MOE(\langle f \rangle_{MST})$

**Propriedade:** Seja um grafo não-direcionado e um fragmento de uma MST $\langle f \rangle_{MST}$ de *G*, uma extremidade de partida é mínima se o custo (peso) da sua aresta, que liga um nó adjacente não pertencente ao fragmento de MST, for mínimo em relação a outras extremidades de partida de $\langle f \rangle_{MST}$ , isto é, $G=(V,E), \langle f \rangle_{MST} \in G,$ $v \in \overline{f}, v' \in f | OE(\langle f \rangle_{MST}) \Leftrightarrow \overline{v, v'}$ . A prova da minimalidade de $OE$ é encontrada em [GAL83].

Os exemplos abaixo ilustram as últimas definições. Na Figura 11, temos os nós em azul são os vértices $\{A, B, C, ..., J, K, L\} \in V$ . As arestas de G apresentam pesos $\overline{(v, v')} \in E, v, v' \in V | w(v, v')$ . Na Figura 12, temos uma árvore geradora mínima, MST, de G a partir do vértice A, composto pelas arestas em negrito.

Um exemplo de fragmento da MST de G gerado a partir do vértice A, $\langle f \rangle_{MST}$ , é dado na Figura 13. $\langle f \rangle_{MST}$ é composto pelas arestas em negrito e vértices A,B,C.

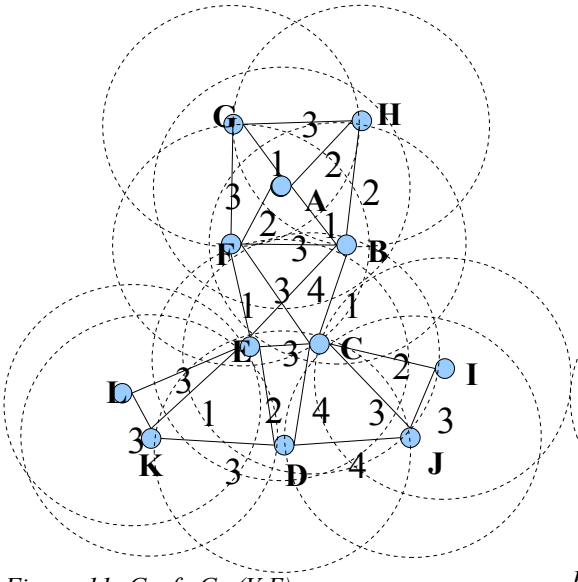


*Figura 11: Grafo G=(V,E)*

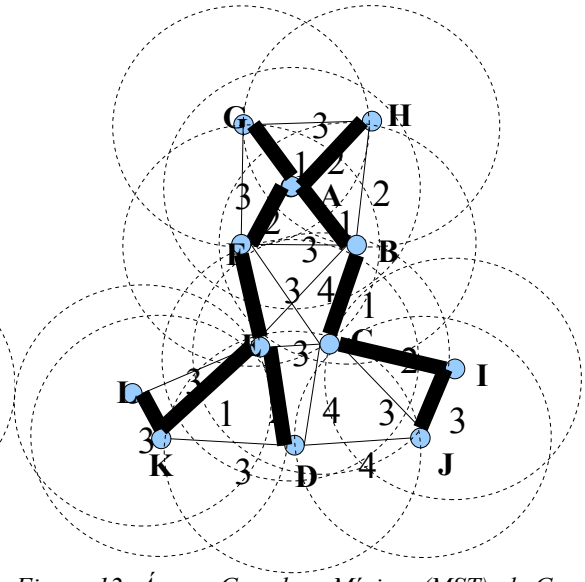


*Figura 12: Árvore Geradora Mínima (MST) de G a partir de A*

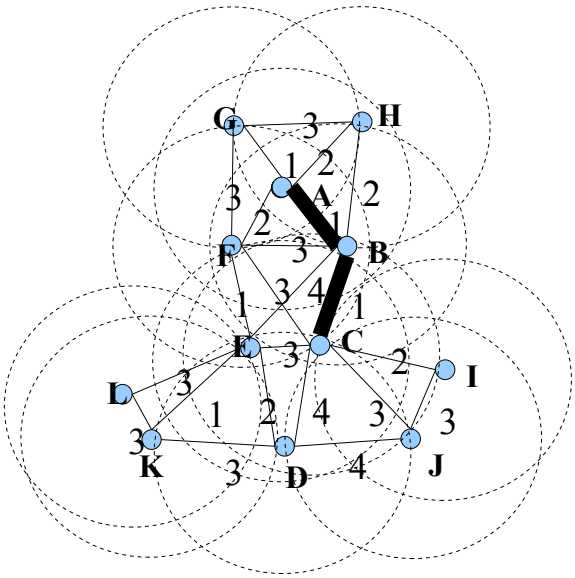


*Figura 13: Fragmento f da árvore geradora mínima de G*

A partir do exemplo dado, as Extremidades de Partida de $\langle f \rangle_{MST}$ da Figura 13 do fragmento da MST de G são mostradas na Figura 15. São compostas pelas arestas em vermelho e pontilhadas. A citar:

- $w(A,F)=2, w(A,G)=1, w(A,H)=2,$ ,
- $w(B,E)=4, w(B,F)=3, w(B,H)=2,$ ,

- $w(C,D)=4, w(C,E)=3, w(C,J)=3, w(C,I)=2$ .

Por fim, neste mesmo exemplo, a extremidade mínima de partida (MOE) na Figura 14, temos que é a aresta $\overline{AG}$ de peso $w(A,G)=1$ , pois $w(A,G)<w(A,F), w(A,H)$, $w(B,E), w(B,F), w(B,H)$, $w(C,D), w(C,E)$, $w(C,J), w(C,I)$ . Nesta figura, a aresta $\overline{AG}$ está assinalada em vermelho, pontilhado e negrito.

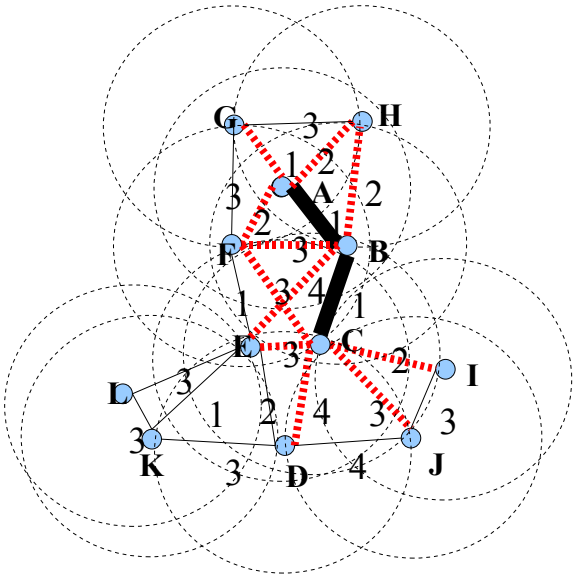


*Figura 15: Extremidades de Partida (OE) do fragmento f de MST de G*

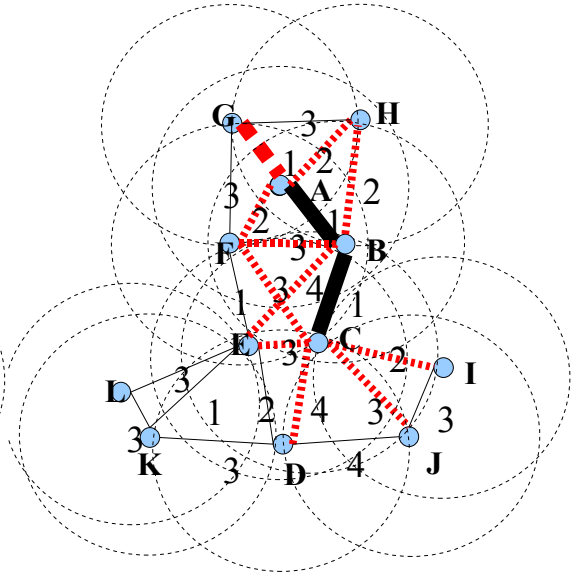


*Figura 14: Extremidade de Partida Mínima (MOE) do fragmento f da MST de G*

### 2.2.12. Core de um novo Fragmento $CF(\langle f \rangle_{MST})$

**Definição:** Seja um grafo não-direcionado e dois fragmentos de uma MST, $\langle f \rangle_{MST}$ e $\langle f' \rangle_{MST}$ , com $\langle f \rangle_{MST}, \langle f' \rangle_{MST} \in G$ , $\langle f \rangle_{MST} \cap \langle f' \rangle_{MST} = \emptyset$ . Se existir uma aresta que é simultaneamente MOE de $\langle f \rangle_{MST}$ e $\langle f' \rangle_{MST}$ , dizemos que ela é *Core* de um novo fragmento, denotado por $CF(\langle f'' \rangle_{MST})$ . Logo, $\langle f'' \rangle_{MST}$ é o resultado da união de $\langle f \rangle_{MST}$ , $\langle f' \rangle_{MST}$ e esta aresta $CF(\langle f'' \rangle_{MST})$ [GAL83], ou seja:

$$\exists e \in E | e = MOE(\langle f \rangle_{MST}) = MOE(\langle f' \rangle_{MST}) \Leftrightarrow$$

$$e = CF(\langle f'' \rangle_{MST}), \langle f'' \rangle_{MST} = \langle f \rangle_{MST} \cup \langle f' \rangle_{MST} \cup e$$

### 2.2.13. Core Nodes $CN(CF(\langle f \rangle_{MST}))$

**Definição:** Seja um grafo não-direcionado de *G*, dois fragmentos de uma MST, $\langle f \rangle_{MST}$ e $\langle f' \rangle_{MST}$, com $\langle f \rangle_{MST}, \langle f' \rangle_{MST} \in G$, $\langle f \rangle_{MST} \cap \langle f' \rangle_{MST} = \emptyset$ e um *Core* de um novo fragmento $CF(\langle f'' \rangle_{MST})$. Dois vértices de G são *Core Nodes*, $CN(CF(\langle f'' \rangle_{MST}))$, se forem ligados por um *Core* de Fragmento $CF(\langle f'' \rangle_{MST})$ [GAL83], ou seja:

$$\overline{v, v'} = CF(\langle f \rangle_{MST}) \rightarrow v, v' = CN(CF(\langle f \rangle_{MST}))$$

Na Figura 16, temos $G = (V, E)$ e os nós em azul formam $V = \{A, B, C, \ldots, J, K, L\}$. As arestas de G apresentam pesos $\overline{(v, v')} \in E, v, v' \in V | w(v, v')$. Em negrito, temos dois fragmentos, $\langle f \rangle_{MST}$ e $\langle f' \rangle_{MST}$, com $\langle f \rangle_{MST} = \{\overline{G, A}, \overline{A, B}, \overline{B, C}\}$ e $\langle f' \rangle_{MST} = \{\overline{F, E}, \overline{E, K}\}$, com suas *OE* assinaladas respectivamente por arestas pontilhadas em vermelho e azul.

Na Figura 17, temos em destaque a aresta $\overline{A, F}$. Se $\overline{A, F} = MOE(\langle f \rangle_{MST})$ e $\overline{A, F} = MOE(\langle f' \rangle_{MST})$ ,com $w(\overline{A, F}) = 1$ então, $\overline{A, F}$ é *Core* de um novo Fragmento $CF(\langle f'' \rangle_{MST})$, isto é, $\overline{A, F} = CF(\langle f'' \rangle_{MST})$, com $\langle f'' \rangle_{MST} = \langle f \rangle_{MST} \cup \langle f' \rangle_{MST} \cup \overline{A, F}$. Ainda nesta figura, temos em destaque os vértices A e F (em verde), que são *core nodes*, $CN(CF(\langle f'' \rangle_{MST}))$.

Por fim, na Figura 18, temos o novo fragmento $\langle f'' \rangle_{MST}$, com $\langle f'' \rangle_{MST} = \langle f \rangle_{MST} \cup \langle f' \rangle_{MST} \cup \overline{A, F}$.

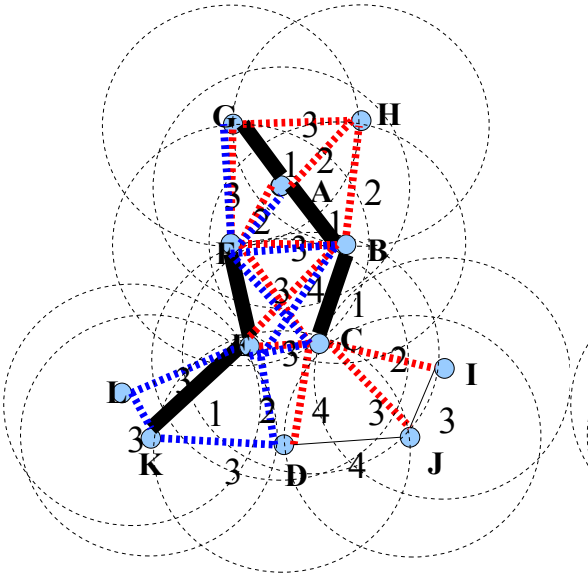


*Figura 16: Fragmentos de G e suas Extremidades de Partida (OE)*

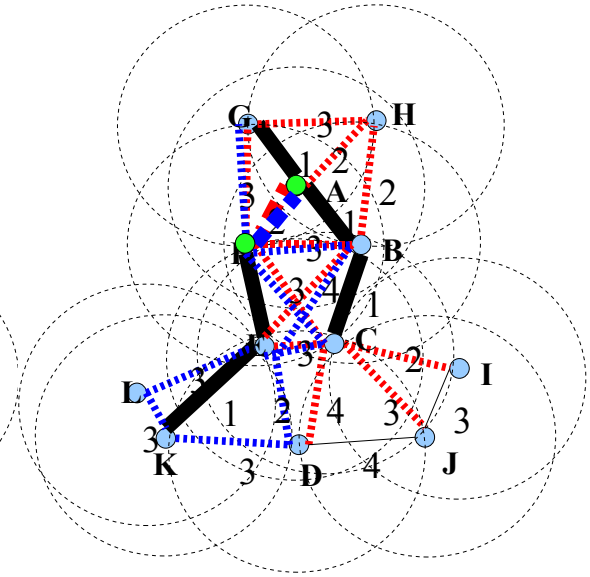


*Figura 17: Extremidades Mínimas de Partida de f e f ' na aresta AF e Core Nodes A e F*

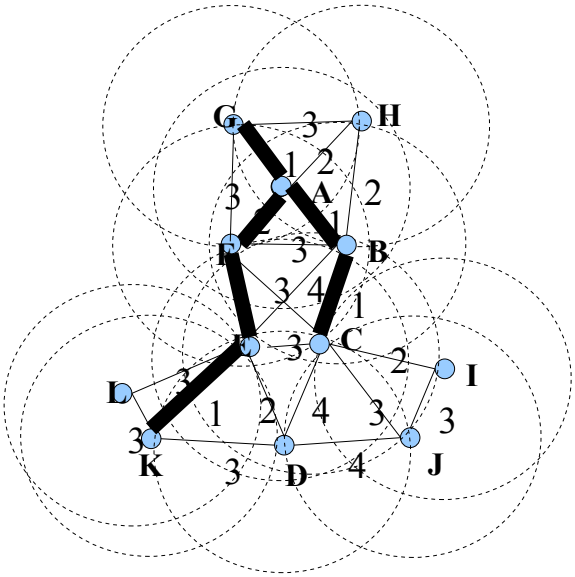


*Figura 18: Novo fragmento f''*

## 2.3. TAXONOMIA

Uma taxonomia para os algoritmos de *clustering* está dividida segundo as características do *dominant set* [CHE03]: *clustering* com conjunto dominante independente, *clustering* com conjunto dominante conexo, *clustering* com conjunto dominante fracamente conexo, *clustering* por outros métodos.

### 2.3.1. Algoritmos de Clustering usando dominant sets independentes

Vários trabalhos foram propostos para determinar um critério de seleção de *clusterheads* a partir de um *independent dominant set, v*ários: [EPH87] propôs um algoritmo segundo o critério do menor identificador do nó, ID, *Lowest-ID Algorithm,* dentro de $N_{closed}(v)$ . Um algoritmo semelhante foi proposto por [GER95], usando como critério de seleção do *degree, highest degree algorithm*, também dentro de $N_{closed}(v)$ .

[CHE99] provou que estes modelos não são aplicáveis para topologias móveis, demonstrando a ocorrência de colisões (*chain collisions*) entre *clusterheads*. Isto ocorre quando dois *clusterheads* dentro de uma "distância-k de vizinhança" se aproximam infringindo a condição de independência dos *dominant sets*. Mais tarde, este problema foi corrigido e generalizado por [BAS99c]. A geração de *dominant sets* seria condicionada à distância de k+1 *hops* entre cada *clusterhead*.

Baseando-se no trabalho realizado por [GER95], foram propostos dois modelos utilizando pesos: DCA – *distributed clustering algorithm* [BAS99a] e DMAC - *distributed mobility adaptive clustering* [BAS99b], para a organização de *clusters*. Resultados complementares foram apresentados por [BET01], comprovando a estabilidade dos modelos anteriores.

Outros resultados usando *clustering* com conjunto *independent dominant sets*: [PAP01], [GER00] , [HOU01].

### 2.3.2. Algoritmos de Clustering usando dominant sets conexos

Vários trabalhos foram propostos para determinar um critério de seleção de *clusterheads*: [GHU96] propôs dois algoritmos distribuídos para encontrar um quase-ótimo conjunto dominante conexo em grafos conexos. A estratégia básica é adicionar vértices ao *dominant set* visando maximizar o número de *clusternodes*. Uma implementação distribuída deste algoritmo para conjuntos dominantes conexos com um coordenador central é encontrada em [DAS97].

Um algoritmo distribuído foi proposto por [JWU99] para encontrar um conjunto dominante conexo em uma topologia em que cada nó somente necessita saber sua distância para os vizinhos.

[CHA00] apresentou um modelo baseado em pesos chamado WCA – *weighted clustering algorithm*. Neste algoritmo, cada nó apresenta um peso determinado por fatores. O peso $W_v$ de um nó $v$ é obtido a partir da composição de quatro fatores: o grau (*degree*) do nó, $\Delta_v$, que avalia a quantidade de vizinhos de $v$, $D_v$, que representa a soma das distâncias da vizinhança de $v$, ou seja, $D_v = \sum_{v' \in N_{open}(v)} \{dist(v, v')\}$ , a mobilidade, $M_v$, e o tempo cumulativo, $P_v$, baseado no tempo em que o nó $v$ foi líder anteriormente. Estes elementos apresentam pesos ponderados $w_1$, $w_2$, $w_3$ e $w_4$, $\sum_{i=1}^{4} w_i = 1$ , com:

$$W_v = w_1 . \Delta_v + w_2 . D_v + w_3 . M_v + w_4 . P_v$$

*Equação 1: Modelo baseado em pesos - WCA*

O nó $v$ de menor peso é selecionado e transformado em *clusterhead*. Os nós da sua vizinhança também são selecionados e se tornam *clusternodes*. O algoritmo continua até que todos os nós sejam selecionados e façam parte de algum subconjunto (*cluster)* $C \subset V$ . O conjunto de *clusterheads* formará o *dominant set*.

No WCA, a condição restritiva à conectividade do *connected dominant set* é mantida pela dualidade na comunicação entre os nós. *Clusterheads* comunicam-se diretamente em taxa de transmissão alta, mas com o inconveniente de maior consumo de energia. *Clusternodes*, por sua vez, comunicam-se dentro de $N_{closed}(v)$ do correspondente líder $v$ em baixa, com a vantagem da economia de energia.

Dois anos mais tarde, o mesmo autor do algoritmo WCA [CHA02] propôs uma

melhoria do seu modelo. Ao final da execução do algoritmo WCA, o *dominant set* gerado passaria por um processo de refinamento usando um algoritmo genético. Desta forma, alguns *clusterheads* e *clusternodes* eram permutados para garantir uma topologia mais otimizada.

Segundo [CHE03], os algoritmos de *clustering* com conjuntos dominantes conexos apresentam como vantagem a formação de um *backbone* virtual (*virtual backbone*) entre os *clusterheads,* minimizando a ocorrência de *overheads* na comunicação em *broadcast*.

### 2.3.3. Algoritmos de Clustering usando dominant sets fracamente conexos

Um tipo particular de *clustering* foi proposto por [CHE02b]. *Clustering* usando um conjunto dominante fracamente conexo foi definido a partir de *clustering* usando um *connected dominant set* de [GHU96] .

É utilizado para configuração de dispositivos com tecnologia *Bluetooth* [HAA98]. Uma rede usando esta tecnologia apresenta duas estruturas do tipo mestre-escravos: *piconet* e *scatternet*. Uma configuração *piconet* apresenta uma topologia em estrela, com apenas um mestre central e os demais são escravos. A junção de várias *piconets* constituem uma *scatternet*. O conjunto dominante fracamente conexo deste grafo será formado a partir dos nós mestres.

Um algoritmo distribuído baseado em zonas foi proposto por [CHE03b]. Cada zona Z é definida como uma subgrafo conexo S de uma topologia G. O tamanho de Z apresenta um parâmetro de controle x. O número de vértices de cada Z possui uma condição restritiva, não podendo passar de 2x, isto é, para $N_{closed}(v)$ , $N(v) \in Z, N(v) \leqslant 2x$ . Para cada Z há um vértice v chamado de *root* e conhecido por toda a vizinhança aberta de *v*. Outros algoritmos usando WCDS(G): [DUB03], [ALZ03] .

### 2.3.4. Algoritmos de Clustering por outros métodos

Um tipo de algoritmo de *clustering* classificado por [CHE03] não faz uso de *dominant sets,* isto é, não faz uso de *clusterheads*, sendo chamado de *clustering* por outros métodos. Um algoritmo distribuído deste tipo foi apresentado por [CHA97]. Neste modelo,

um *cluster* corresponde a uma clique maximal de um grafo correspondente a uma topologia G. Os nós *v* de uma mesma $N_{closed}(v)$ comunicam-se diretamente. Um nó chamado nó limite *b* (*boundary node*) é sobreposto por mais de um *cluster*. Por pertencer a duas vizinhanças fechadas $N_1(b)$ e $N_2(b)$, um nó limite b serve de *relay* para a comunicação entre quaisquer nós $v_1$ e $v_2$ pertencentes a estas duas vizinhanças distintas $N_1(v_1)$ e $N_2(v_2)$. Esta idéia é similar ao conceito de roteamento BGP [RFC1771].

[BAN00] propôs um modelo baseado em árvores geradoras *(spanning trees)* para a formação de *clusters*. Cada *cluster* é um subconjunto de vértices de um subgrafo conexo. Estes subconjuntos são determinados levando em consideração o número máximo de *clusters* ao qual um nó poderá pertencer e o tamanho de *cluster*. Ao contrário do modelo proposto por [CHA97], embora possa haver sobreposição dos *clusters* resultantes, cada vizinhança fechada de v, *N(v)*, podem não estar conectada diretamente.

Outros trabalhos relacionados a algoritmos de *clustering*, com ou sem o uso de *dominant sets* podem ser encontrados: [ALZ02a], [GAO01], [JIA01], [KUH03], [ALZ02b].

## 2.4. *LOWEST-ID ALGORITHM* - LID

Neste algoritmo, também chamado de *identifier-based clustering*, o particionamento em *clusters* ocorre a partir da escolha de nós com menor identificador (ID), atribuído a cada nó [EPH87]. O LID é um tipo de algoritmo de *clustering* usando um *independent dominant set* $IDS(G)$ [CHE03].

Em cada *cluster*, a determinação dos *clusterheads*, um por *cluster*, é feito periodicamente, com cada nó enviando em *broadcast* aos nós que pertencem à lista de sua vizinhança (incluindo a si mesmo).

Um nó que receber dos seus vizinhos somente ID's maiores do que ele será o *clusterhead*. Um nó que se comunica com dois ou mais *clusterheads* denomina-se *gateway*. Os demais nós da vizinhança aberta do nó líder são considerados membros (*clusternodes)*. Um determinado *cluster* é identificado pelo ID do seu líder.

A Figura 19 exemplifica a aplicação do algoritmo. Considere uma rede com 10 nós com ID's distintos (1, 2,..., 10) e mesma capacidade (*raio*) de transmissão (*r*). Após a

execução do algoritmo, ter-se-á um grafo conectado e três *clusters* formados, cada qual com o seu líder (nós 1, 2,4). Além disso, temos como *gateways* os nós 8 e 9. Os demais nós (3, 5, 6, 7,10) são apenas *clusternodes* e fazem parte da vizinhança aberta dos respectivos *clusterheads*.

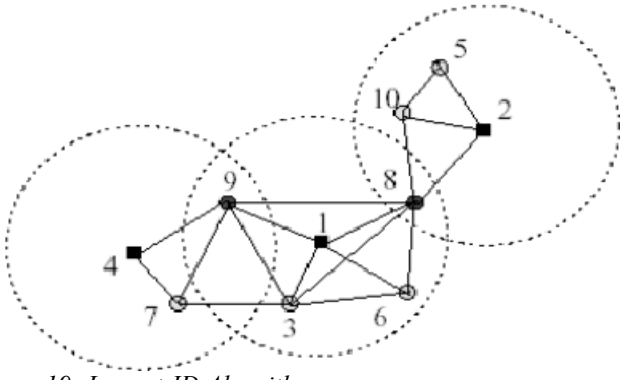


*Figura 19: Lowest-ID Algorithm*

Este algoritmo é de implementação muito simples. No entanto, apresenta alguns problemas:

- A escolha do nó com menor ID como *clusterhead* pode levar a uma mesma escolha de nós com ID's de valor baixo, algumas vezes de forma consecutiva. Desta forma, concentrar-se-á o roteamento de informações sobre os mesmos nós. Estes nós serão sobrecarregados e podem sofrer exaustão de recursos (energia) mais rapidamente em relação aos demais nós. O sistema pode ser comprometido como um todo. Para minimizar este problema, uma solução é redistribuir os ID's periodicamente em função da energia armazenada [CHA00]. O nó que apresentar maior energia receberá o menor ID e será forte candidato a ser *clusterhead* na próxima execução do algoritmo. Por outro lado, não é uma solução trivial e sucessivas permutações podem ocasionar *overheads* no sistema. Esta alternativa em alguns casos pode ser igualmente desvantajosa.

## 2.5. *HIGHEST-DEGREE ALGORITHM* - HDA

O particionamento em *clusters* é baseado na escolha do nó de mais alto *degree.* O HDA é um tipo de algoritmo de *clustering* usando um *independent dominant set* $IDS(G)$ [CHE03].

A determinação dos *clusterheads* é feita da seguinte forma: cada nó envia o seu ID em broadcast para os seus vizinhos, dentro do raio de transmissão. O nó que apresentar maior grau é escolhido como *clusterhead*. Os nós que estiverem na sua vizinhança são membros deste *cluster* e não poderão participar mais do processo de eleição de líder. Vale salientar que somente um *clusterhead* é permitido por *cluster*. A Figura 20 exemplifica a aplicação do algoritmo. Os nós 5,7 e 8 são os *clusterheads*. Os nós 2, 3, 9 e 10 são os *gateways*. Estes *gateways* têm a mesma finalidade que o algoritmo anterior. Os demais nós são simplesmente *clusternodes*.

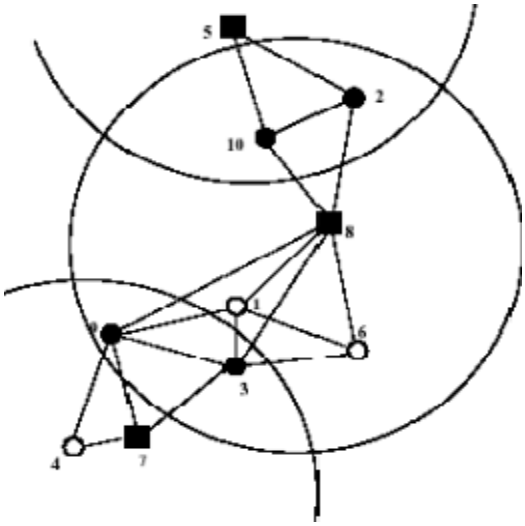


*Figura 20: Rede Ad hoc com 3 clusters usando Highest-Degree Algorithm*

Este algoritmo apresenta o seguinte problema:

- Sobrecarga de nós com um grande número de nós na sua vizinhança. Há uma grande possibilidade deste nó ser líder por várias vezes, em alguns casos consecutivamente, até que haja uma dispersão dos seus *clusternodes* por deslocamento. Neste caso, tem-se um raciocínio análogo ao algoritmo anterior. Estes nós serão sobrecarregados e podem sofrer exaustão de recursos (energia) mais rapidamente em relação aos demais nós, comprometendo como um todo o sistema. Uma solução para este problema é limitar cada nó a um determinado número de vizinhos, amenizando a sobrecarga dos *clusterheads* [CHA00]. Por outro lado, esta solução deve ser equilibrada. Há a possibilidade da geração de um grande número de *clusters*, o que não seria igualmente desejável.

## 2.6. WEIGHTED CLUSTERING ALGORITHM - WCA

Um modelo de *clustering* é o WCA - *Weighted Clustering Algorithm* [CHA00], [CHA02]. Este modelo utiliza a distribuição de pesos de nós, a partir de determinados fatores. É um tipo de algoritmo de *clustering* usando um *connected dominant set* $CDS(G)$ [CHE03].

Este modelo, para a organização de *clusters*, baseou-se nos algoritmos baseados em pesos, DCA – *distributed clustering algorithm* [BAS99a] e DMAC - *distributed mobility adaptive clustering* [BAS99b].

A distribuição de pesos é feita da seguinte forma: o peso $W_i$ de um nó *i* é obtido a partir da composição de quatro fatores:

a) O grau (*degree*) do nó, $\Delta_v$ , avalia a quantidade de vizinhos de *v*. É definido por: $\Delta_v = |d_v - \delta|$ , sendo $\delta$ o número ideal de nós em uma vizinhança aberta de $v$ e $d_v$ definido por: $d_v = |N(v)| = \sum_{v' \in V, v' \neq v} \{dist(v, v') < tx_{range}\}$ , sendo $tx_{range}$ o raio de transmissão de *v*.

b) $D_v$ representa a soma das distâncias dos *clusternodes* da vizinhança aberta *N(v)* a *v*, $D_v = \sum_{v' \in N(v)} dist(v, v')$

c) $M_v$ representa o grau de mobilidade de cada *v* dentro de uma unidade de tempo *T*: $M_v = \frac{1}{T} . \sum_{t=1}^{T} \sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2}$ , onde $(X_t, Y_t)$ e $(X_{t-1}, Y_{t-1})$ são as coordenadas de v nos tempos $t$ e $t-1$ , respectivamente.

d) $P_v$ é o tempo cumulativo em que o nó *v* foi líder anteriormente.

Estes elementos apresentam pesos ponderados $w_1$, $w_2$, $w_3$ e $w_4$, respectivamente, com $\sum_{i=1}^{4} w_i = 1$ . O peso de cada nó *v* é: $W_v = w_1 . \Delta_v + w_2 . D_v + w_3 . M_v + w_4 . P_v$ .

O objetivo é determinar o nó de menor peso $W_v$ . O nó selecionado será considerado líder e seus vizinhos serão considerados *clusternodes*, que não serão mais candidatos a se tornarem *clusterheads*. O processo continua com a escolha do nó de menor peso, dentro do conjunto de nós que ainda não foram considerados líderes ou membros de *clusters* já formados.

No WCA, a condição restritiva à conectividade do *connected dominant set* é mantida pela dualidade na comunicação entre os nós. *Clusterheads* comunicam-se diretamente em taxa de transmissão alta, mas com o inconveniente de maior consumo de energia. *Clusternodes*, por sua vez, comunicam-se dentro da $N_{closed}(v)$ do correspondente líder *v* com taxa de transmissão baixa, com a vantagem da economia de energia.

Com isso, temos a formação de um *backbone* virtual entre os *clusterheads* minimizando a ocorrência de *overheads* na comunicação em *broadcast*. Esta é uma vantagem dos algoritmos de *clustering* com conjuntos dominantes conexos [CHE03].

Um problema encontrado no *WCA* é o tratamento da mobilidade [GAR05]. A adoção de um valor escalar como indicativo de mobilidade de um nó *v*, $v \in V$ , não leva em consideração:

- A direção e sentido do nó *v*. A omissão destas informações torna o modelo de mobilidade adotado menos realístico. Este modelo propõe que nós mais aptos a se tornarem *clusterheads* são aqueles que apresentam menor mobilidade, i.e., menor valor escalar de $M_v$ , $v \in V$ . No entanto, não seria uma boa estratégia escolher um nó que está quase estático como *clusterhead*, em situações em que toda a sua vizinhança está se afastando deste [GAR05].

- O movimento relativo de um nó *v,* $v \in V$ , em relação à sua vizinhança. A omissão destas informações pode ocasionar a valorização de um nó candidato à líder que apresente velocidade escalar com valores próximos à sua vizinhança, mas está se deslocando na direção oposta dos mesmos. Supondo que a vizinhança de *v* está se deslocando em uma mesma direção e sentido [GAR05].

Outro problema encontrado neste modelo é com relação à disponibilidade dos

nós [GAR05]. Em [CHA00] e [CHA02], não é considerada a possibilidade de *falhas* eventuais de nós pertencentes à rede [HWA98]. É desejável que um modelo realístico considere a possibilidade de falhas de um nó. A omissão desta informação pode levar à escolha de um *clusterhead* falho, o que não seria desejável .

## 2.7. ZONAL CLUSTERING ALGORITHM - ZCA

Neste algoritmo, o particionamento em *clusters* ocorre a partir de zonas e um conjunto dominante fracamente conexo, formado a partir da árvore geradora mínima de G [CHE03b]. Apresenta as seguintes etapas: particionamento, determinação do $WCDS(Z)$ e fixação de limites.

O grafo G que mapeia uma topologia de rede é primeiramente particionado em zonas não-sobrepostas. Em seguida, é determinado o conjunto dominante fracamente conexo em cada região. O *dominant set* de G é constituído pela união dos conjuntos dominantes formados localmente. Por fim, a última fase é um ajuste. Alguns vértices de regiões de borda de cada zona são adicionados ao *dominant set* de G para assegurar que $DS(G)$ seja fracamente conexo.

A primeira etapa é baseada no algoritmo distribuído GHS (*Gallager-Humblet-Spira Algorithm*) [GAL83], usado para determinar a árvore geradora mínima (*minimum spanning tree* – MST) de uma topologia G. Esta solução é ótima se os pesos de todas as arestas forem distintos. Para grafos com pesos não-distintos, utiliza-se um nó raiz como referência inicial [CHE02]. Uma vez que o grafo esteja particionado e a MST gerada, a segunda etapa utiliza um método guloso para determinar o *dominant set* de G, $DS(G)$ [CHE02] e [GHU98].

O GHS baseia-se na localização de Extremidades Míninas de Partida, $MOE(\langle f \rangle)$ de fragmentos e na sua união com outros fragmentos. A execução termina quando não houver $MOE(\langle f \rangle)$ , indicando que a árvore geradora mínima – MST – de G foi encontrada.

O ZCA difere do GHS na sua condição de parada, acrescentando um parâmetro limitador experimental para o número de vértices em cada zona.

A segunda etapa utiliza um método guloso para determinar o *dominant set*

fracamente conexo de cada zona Z não-sobreposta de G, $WCDS(Z)$ ,[CHE02] e [GHU98], uma vez particionado $G=(V,E)$ e gerado $MST(Z)$ . Esta solução é ótima se os pesos de todas as arestas forem distintos. Para grafos com pesos não-distintos, utiliza-se um nó raiz como referência inicial [CHE02]. Este algoritmo associa uma cor (branco, cinza ou preto) para cada vértice. Inicialmente, todos os vértices são brancos e, à medida que a execução ocorre, podem se tornam cinza ou preto. Quando um vértice *v* se torna preto, todos os vértices da vizinhança aberta de *v* , $N(v)_{open}$ , estão cinzas. No final, os vértices pretos constituirão o *dominant set* fracamente conexo.

A terceira etapa é constituída de um ajuste. Alguns vértices de regiões de borda de cada zona são adicionados ao *dominant set* de G, DS(G), para assegurar que este conjunto seja fracamente conexo.

O ZCA é um tipo de algoritmo de *clustering* usando um conjunto dominante fracamente conexo $WCDS(G)$ [CHE03].

Inicialmente tem-se N, $N=V$ , fragmentos contendo somente um vértice. A expansão dos fragmentos é feita usando a propriedade descrita na MOE. A propriedade da unicidade da MST garante que quando dois fragmentos tiverem um vértice em comum, haverá uma nova união formando um único fragmento. Por isso, as sucessivas uniões constituirão um MST para G.

No ZCA, cada fragmento procura a sua Extremidade Mínima de Partida $(MOE(\langle f\rangle))$ e tenta realizar uniões com fragmentos vizinhos [GAL83]. O critério de união depende do nível de cada fragmento $L(\langle f\rangle)$ .

- Estado Inicial: um fragmento contendo um nó está no nível zero, i.e., $L(\langle f\rangle)=0$ ;
- Hipótese1: se $MOE(\langle f_1\rangle)=MOE(\langle f_2\rangle)$ e $L(\langle f_1\rangle)=L(\langle f_2\rangle)=L$ , então haverá união formando um novo fragmento $\langle f_{new}\rangle$ com $L(\langle f_{new}\rangle)=L+1$ ;
- Hipótese2:se $MOE(\langle f_1\rangle)=MOE(\langle f_2\rangle)$ e $L(\langle f_1\rangle)=L$ , $L(\langle f_2\rangle)=L'$ , com $L'>L$ , então $\langle f_1\rangle$ é absorvido por $\langle f_2\rangle$ . Não haverá a formação de um novo fragmento e $\langle f_2\rangle$ permanece com $L(\langle f_2\rangle)=L'$ ;

- Hipótese3:se $MOE(\langle f_1 \rangle) = MOE(\langle f_2 \rangle)$ e $L(\langle f_1 \rangle) = L$ , $L(\langle f_2 \rangle) = L'$ , com $L' < L$ , então $\langle f_1 \rangle$ não é absorvido por $\langle f_2 \rangle$ .

Se um fragmento exceder o tamanho determinado pelo parâmetro x, é considerado cheio e não poderá mais ser selecionado. O algoritmo termina quando não houver nenhuma Extremidade Mínima de Partida nos fragmentos. Algumas restrições são aplicadas durante a execução para garantir o crescimento balanceado entre os fragmentos [CHE03b].

Ao contrário do ZCA, o algoritmo GHS termina simplesmente se não houver nenhuma Extremidade Mínima de Partida nos fragmentos, indicando que o MST foi encontrado. Ao final desta etapa, obtemos a árvore geradora mínima de G. Esta solução é ótima se os pesos de todas as arestas forem distintos [GAL83].

Uma vez particionado $G=(V,E)$ e gerado $MST(G)$ , a segunda etapa utiliza um método guloso para determinar o *dominant set* fracamente conexo de cada zona Z não-sobreposta de G, $WCDS(Z)$ [CHE02] e [GHU98] . Esta solução é ótima se os pesos de todas as arestas forem distintos. Para grafos com pesos não-distintos, utiliza-se um nó raiz (*root*) como referência inicial [CHE02].

Nesta etapa, alguns conceitos são importantes:

Uma peça (*piece*) é um subgrafo de G. Um fragmento $\langle f \rangle_{MST}$ é um tipo particular de peça. Pode ser:

- Branca: é constituído simplesmente por um vértice branco;
- Cinza: são os vértices que estão dentro de uma vizinhança aberta $(N_{open}(v))$ de um vértice preto *v;*
- Preta: é formado pelos vértices pretos, cujo subgrafo fracamente induzido $\langle S \rangle_w$ é conexo pelas suas Extremidades de Partida $OE(\langle S \rangle_w)$ a qualquer vértice cinza. Vértices pretos constituem um *dominant set* fracamente conexo da zona Z, $WCDS(Z)$ .

Valor de melhoria (*improvement value*) de um vértice *v* branco ou cinza é o número de peças distintas dentro de uma $N_{closed}(v)$. O valor de melhoria é aplicável apenas a vértices brancos ou cinzas.

Inicialmente, todos os vértices são brancos. Em cada iteração, dentro de cada zona, o algoritmo guloso escolhe um vértice branco ou cinza, com valor de melhoria (*improvement value*) maximal, para se tornar preto. O critério de escolha visa reduzir o número de peças (*pieces*) até que se reduza, na medida do possível, a uma peça somente. Quando um vértice *v* se torna preto, a vizinhança branca aberta de *v*, $N(v)_{open}$ , está cinza. A condição de parada é atingida quando houver somente uma peça (*piece*) por zona. Ao final, os vértices pretos constituirão o *dominant set* fracamente conexo.

Como o algoritmo é distribuído em cada zona, o caminho mínimo determinado pela MST na fase anterior é utilizado para tráfego de mensagens em cada zona.

A terceira etapa é constituída de um ajuste, chamada de Fixação de Limites. Alguns vértices de regiões de borda de cada zona são adicionados ao *dominant set* de G para assegurar que o *dominant set* $DS(G)$ seja fracamente conexo. A zona de menor ID é responsável pela fixação de limites entre zonas adjacentes.

Isto ocorre pois a etapa anterior calcula o *dominant set* fracamente conexo , $WCDS(Z)$ , de cada zona Z não-sobreposta de G. Combinando cada $WCDS(z_i)$ , $z=0,1,...,|Z|-1$ , não necessariamente teremos um *dominant set* fracamente conexo em G, $WCDS(G)$ . Uma solução é adicionar alguns vértices limítrofes brancos ou cinzas para compor o $WCDS(G)$ .

Por exemplo, na Figura 21, suponha que cada zona $R_1, R_2, ..., R_5$ contenha *dominant sets* fracamente conexos. Seus vértices estão assinalados em preto. A zona $R_1$ é adjacente às regiões $R_2, ..., R_5$ . $R_1$ não forma um subgrafo fracamente conexo com as zonas $R_2, R_3$ . Então, os vértices u e v são adicionados ao *dominant set* de $R_1$ . $R_2$ não forma um subgrafo fracamente conexo com a zona $R_3$ . Então, o vértice w é adicionado ao *dominant set* de $R_2$ .

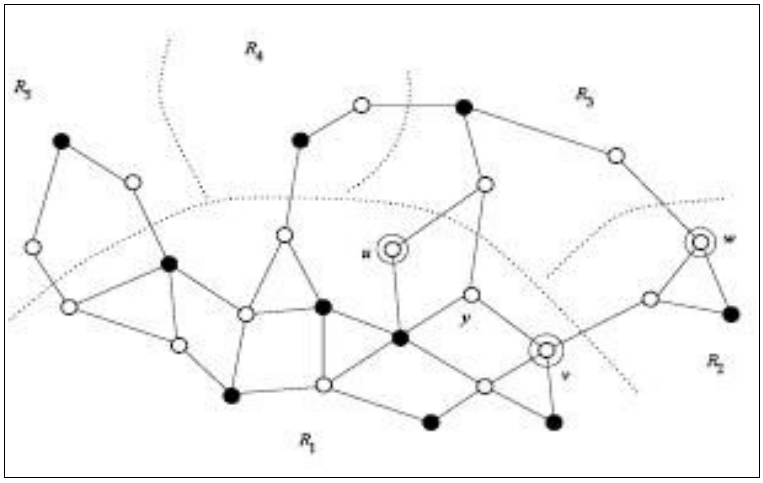


*Figura 21: ZCA:- fixando limites para gerar um dominant set global fracamente conexo - WCDS global*

Como o algoritmo é distribuído em cada zona, ao final ter-se-á um *dominant set* fracamente conexo em G, $WCDS(G)$ .

Uma vantagem da organização de *cluster* por zonas é o controle do número de vértices em cada *cluster* pelo parâmetro *x*. Dependendo do seu valor, podemos variar a granulosidade da topologia, conforme cada necessidade.

Um problema encontrado neste modelo também é com relação ao parâmetro x. Este valor é estático na topologia e não muda dinamicamente .

# 3-METAHEURÍSTICAS

## 3.1. INTRODUÇÃO

Metaheurística – MH é um tipo especial de heurística idealizada para fazer o controle da própria heurística, a fim de escapar de mínimos (ou máximos, conforme o caso) locais [MIL04].

O prefixo *meta* é utilizado para descrever uma heurística que está sobreposta a uma outra heurística, constituindo um outro nível "heurístico" [ARR02]. Em geral, uma MH constitui uma estrutura mais genérica baseada em princípios ou conceitos, sobreposta a uma heurística específica do problema em estudo.

Uma classificação abrangendo as principais técnicas de metaheurísticas e suas variações pode ser encontrado em [MIL04] e está dividida nas seguintes classes: algoritmos que utililizam algum tipo de movimento aleatório para escapar ou evitar mínimos locais (*Random Move Based Algorithms*-RMBA), algoritmos que utilizam populações de soluções (*Population Based Algorithms*–PBA), algoritmos baseados em vizinhança (*Neighborhood Based Algorithms*–NBA) e algoritmos baseados em pesos e penalidades (*Weighted and Penalty Based Algorithms*–WPBA).

## 3.2. RANDOM MOVE BASED ALGORITHMS - RMBA

RMBA são metaheurísticas que acrescentam algum parâmetro aleatório ou ruído ao processo de busca para escapar de ótimos locais. Estão nesta categoria os seguintes algoritmos:

- *Iterated Local Search (ILS),*
- *Greedy Randomized Adaptive Search Procedures (GRASP),*
- *Simulated Annealing* (SA),
- *Simulated Jumping* (SJ).

### 3.2.1. Iterated Local Search - ILS

O ILS [STU99] é uma técnica criada a partir da busca local associada a perturbações aleatórias sobre um ótimo local.

A idéia da ILS é determinar mínimos locais e modificá-los, usando um conjunto de movimentos aleatórios (perturbações) a partir dele para escapar de mínimos locais indesejados e buscar novas soluções, que o processo de busca local não seria capaz de encontrar. Com isso, ótimos locais de um problema de otimização podem ser gerados a partir de perturbações na solução ótima local corrente.

A aplicação de pequenas perturbações na solução ótima corrente é chamada de Intensificação. A aplicação de grandes perturbações na solução ótima corrente, podendo gerar até soluções aleatórias, é chamada de Diversificação.

O pseudocódigo do ILS é mostrado na Figura 22. Inicialmente, é gerada uma solução inicial e uma busca local para melhorar $x_0$ , (*GenerateInitialSolution(), LocalSearch( $x_0$ )*). Em seguida, é aplicada uma perturbação em *x* (*Perturbation(x,history)*), a partir de movimentos aleatórios levando em consideração o seu histórico. Esta solução é novamente melhorada com um procedimento de busca local (*LocalSearch(x')*). A solução obtida é então comparada com *x*, basicamente verificando $f(x), f(x'')$ e $history(x), history(x'')$ . x" torna-se, então, a nova solução. O processo continua até que a condição de parada seja satisfeita ( *ITERMAX* ). A melhor solução encontrada x" é retornada pela função.

```
IteratedLocalSeach
{     x0  = GeneratInitialSolution();
      x   = LocalSearch( x0 );
      for i=1 to ITER_MAX
      {     x'  = Perturbation( x ,history);
            x'' = LocalSearch( x' );
            x   = AcceptanceCriterion(x",x,history); }
      return x;
}
```

*Figura 22: Algoritmo Iterated Local Seach - ILS [STU99]*

A perturbação precisa ser suficientemente forte para permitir que a busca local

consiga explorar diferentes soluções e fraca o suficiente para evitar um reinício aleatório, ou seja, uma alternativa entre a Intensificação e a Diversificação, com movimentos de perturbação nem muito fracos, nem fortes.

A principal limitação do ILS é se encontramos um ponto intermediário entre Intensificação e Diversificação e as soluções locais estiverem muitos distantes de uma solução quase-ótima, os movimentos de perturbação não serão fortes o suficiente para causar a aproximação desejada [MIL04].

### 3.2.2. Greedy Randomized Adaptive Search Procedures - GRASP

Também chamado de Procedimento de Busca Gulosa Adaptativa Aleatória. Trata-se de uma MH para problemas de otimização combinatória. Originariamente foi uma heurística de busca local [RES95]. Posteriormente, incorporou novas características e se tornou uma MH [RES03].

É um método iterativo, onde cada iteração consiste de duas fases, uma fase de construção de uma solução viável *s* e uma fase de busca local sobre *s* [RES03].

Na fase de construção, uma solução viável é construída, elemento a elemento. Em cada iteração deste processo, os próximos elementos candidatos a serem incluídos na solução são colocados em uma lista, chamada de *Restricted Candidate List* (RCL), de acordo com um determinado critério. A seleção do próximo elemento da lista é baseada em uma função adaptativa gulosa, que estima o benefício da seleção de cada um dos elementos.

A fase da busca local atua de forma iterativa, substituindo a solução corrente $x$ pela melhor da sua vizinhança $x'$. Esta fase termina quando não houver mais soluções de melhora na vizinhança de $x$. Tem-se então a melhor solução encontrada *BestSolutionFound* igual a x.

O pseudocódigo do SA é mostrado na Figura 23. Cada iteração apresenta uma construção (*ConstructGreedyRandomizedSolution*), uma busca local (*LocalSearch*) e uma atualização da melhor solução se $f(x') < f(x)$ (*UpdateSolution*). A melhor solução encontrada é armazenada em *BestSolutionFound*. O procedimento para a construção (*ConstructGreedyRandomizedSolution*) é mostrado na Figura 24. Nesta função, pontos de partida são gerados e armazenados na RCL.

```
GRASP(RCLSize,MaxIter,RandomSeed)
{     InputInstance();
      InicializeDataStructures();
       BestSolutionFound=0;
      for i=1 to MAX_ITER
      {
            ConstructGreedyRandomizedSolution(RCLSize, RandomSeed);
            LocalSearch(BestSolutionFound);
            UpdateSolution(BestSolutionFound);
      }
      return BestSolutionFound;
}
```

*Figura 23: Algoritmo GRASP [RES95]*

```
Procedure ConstructGreedyRandomizedSolution (RCLSize,RandomSeed)
{     for i=1 to n
      {
            RCL=MakeRestrictedCandidateList(RCLSize);
            ⟨x,v⟩ =SelectAtRandom(RCL);
            x=AssignVariable(x,v);
            ReevaluateIncrementalCosts(x);
      }
      return x;
}
```

*Figura 24: Procedimento ConstructGreedyRandomizedSolution para GRASP [RES95]*

A otimalidade local das soluções geradas na fase de construção não é garantida. Por isso, torna-se importante a fase da busca local, que visa melhorar as soluções encontradas na fase anterior.

A eficiência da busca local depende da qualidade da solução construída, uma vez que as soluções construídas constituem bons pontos de partida para a busca local. Logo, a fase da construção tem papel igualmente importante na busca local.

Quanto ao ajuste experimental da função, o parâmetro que determina o tamanho da RCL é basicamente o único parâmetro a ser ajustado na implementação. [RES95] discutiu o efeito deste valor na qualidade da solução e na diversidade das soluções geradas durante a fase de construção. Valores limitados para RCL implicam em soluções de qualidade muito próxima do resultado do algoritmo guloso, com baixa diversidade de soluções construídas, mas com esforço computacional proporcionalmente menor. Por outro lado, uma escolha mais próxima da seleção aleatória leva a uma grande diversidade de soluções construídas. Muitas destas soluções têm qualidade inferior tornando mais lento o processo de busca local, exigindo maior esforço computacional.

Uma vantagem do GRASP sobre os algoritmos puramente guloso é a agregação de procedimentos aleatórios, garantindo diversidade de soluções (fase da construção), além de utilizar a busca local para uma convergência rápida (fase da busca local).

### 3.2.3. Simulated Annealing - SA

*Simulated Annealing* é um método de busca local probabilístico [KIR83], que admite soluções de piora do tipo $T \to 0 \Rightarrow e^{-\Delta f/T} \to 0$ para escapar de ótimos locais. Originalmente, foi derivado de simulações em termodinâmica e por esta razão o parâmetro $T$ é referenciado como temperatura e a maneira pela qual ela é reduzida é chamada de processo de resfriamento.

Trata-se de uma variante do Método de Metrópolis [MET53], que simula o comportamento de uma coleção de átomos em equilíbrio numa dada temperatura por meio da geração de variáveis aleatórias distribuídas de acordo com uma dada distribuição multivariada de probabilidade.

Se o valor da função de avaliação $f(s')$ do vizinho for menor que o da solução corrente $f(s)$ , com $\Delta f = f(s') - f(s) < 0$ , então, a solução corrente é substituída. Caso contrário, se ocorrer um movimento de piora, a solução é aceita com uma probabilidade de $e^{-\Delta f/T}$ , onde $T$ decresce gradualmente conforme o progresso do algoritmo, regulando a probabilidade de se aceitar soluções de piora. A temperatura inicial T é obtida experimentalmente [REE93]. O processo é repetido até que a condição de parada, pela função *TerminationCondition()*, seja atingida, isto é, se

- O máximo número de iterações for obtido $SAMAX$ . Neste caso, o sistema atinge o equilíbrio térmico em uma dada temperatura;
- Não houver mais movimentos de melhora após um certo número de iterações *i* (efeito da cristalização).

A temperatura $T$ é diminuída por uma razão de resfriamento $\alpha$ , tal que , $T_n \leftarrow \alpha . T_{n-1}$ sendo $0 < \alpha < 1$ . A melhor solução encontrada durante a busca é tomada como uma boa aproximação para a solução ótima.

O pseudocódigo do SA é mostrado na Figura 25:

```
Simulated Annealing Algorithm
{
    s0 = GenerateInitialSolution();
    T = StartTemperature();
    i = 0;
    do
    {   Random Select  s∈N(s0) ;
        Δf = f(s) − f(s0) ;
        if( Δf < 0 )  s0 = s ;
        else
        {   r = SelectRandomNumber[0..1];
            if( r < e^(−Δf/T) )  s0 = s ;   //accept the change
        }
        i ++;
        T = α.T ;
    }while not TerminationCondition;
}
```

*Figura 25: Algoritmo Simulated Annealing (SA) [REE93]*

Se $\alpha \ll 1$ , a convergência é rápida para o resfriamento. Por outro lado, se $\alpha \approx 1$ , a convergência será lenta.

Uma vantagem do SA em relação aos algoritmos heurísticos é a natureza da movimentação. O SA atinge a solução final, considerada quase-ótima, usando movimentos de descida (ou subida, conforme o caso) e ocasionalmente realiza alguma subida (ou descida). Por outro lado, os algoritmos heurísticos, cujo comportamento é guloso, apenas maximizam ou minimizam a função objetivo para apenas uma solução ótima local.

### 3.2.4. Simulated Jumping - SJ

O *Simulated Jumping* (SJ) [AMI99] é uma variação do SA .

Baseia-se no princípio alguns materiais apresentarem propriedades ferromagnéticas e antiferromagnéticas em estados metaestáveis, sendo muito mais difícil

encontrar um estado estável, com baixa energia, simplesmente por resfriamento. Neste caso, adota-se aquecimento e resfriamento alternadamente até atingir um estado quase-ótimo. Um material é dito ferromagnético quando exibe fenômeno de histerese onde a permeabilidade magnética depende da força de magnetização. O pseudocódigo geral do SJ é mostrado na Figura 26.

```
Simulated Jumping Algorithm
{
    s0 = GenerateInitialSolution();
    T = StartTemperature();
    i = 0;
    do
    {   Random Select  s∈N(s0) ;
        Δf = f(s) − f(s0) ;
        if( Δf < 0 )  s0 = s ;
        else
        {   r = SelectRandomNumber[0..1];
            if( r < e^(−Δf/T) )  s0 = s ;        //accept the change
            else                 T = T + R/i ;   //heat the system
        }
        i ++;
        T = α . T ;                              //cool the system
    }while not TerminationCondition;
}
```

*Figura 26: Algoritmo Simulated Jumping (SJ) [AMI99]*

O algoritmo é muito semelhante ao SA, mas diferindo nos seguintes aspectos:

- O resfriamento *(cooling)* e o aquecimento *(heating)* podem ocorrer alternadamente.
- Se não houver movimentação $(s_0 \leftarrow s)$ , T só aumenta (heating)

O processo é repetido até que a condição de parada, pela função *TerminationCondition()*, seja atingida, isto é, se

- O máximo número de iterações for obtido *SAMax* . Neste caso, o sistema

atinge o equilíbrio térmico em uma dada temperatura;

- Não houver mais movimentos de melhora após um certo número de iterações *i* (efeito da cristalização).

Os valores para *cooling* e *heating* devem ser obtidos experimentalmente de acordo com o problema proposto [AMI99].

São parâmetros típicos para o algoritmo: $T_0 = 0{,}001$ , $\alpha = [0.001, 0.2]$ , $R = 0.15$ [AMI99].

A diferença para o SA é o número de oscilações de descida (ou subida, conforme o caso) e subida (ou descida), que ocorrem com maior freqüência, até a obtenção de uma solução quase-ótima.

## 3.3. POPULATION BASED ALGORITHMS - PBA

PBA são metaheurísticas que armazenam uma população inicial de soluções para proporcionar capacidade de busca adaptiva do algoritmo.

Estão nesta categoria os seguintes algoritmos:

- *Genetic Algorithms (GA),*
- *Memetic algorithms (MA),*
- *Ant Colony Optimization Algorithm* (ACO),
- *Scatter Search* (SS),
- *Path Relinking* (PR).

### 3.3.1. Algoritmos Genéticos - GA

Algoritmos genéticos (*genetic algorithms*-GA's) são metaheurísticas utilizadas para solucionar problemas de busca e otimização de forma adaptativa [HOL75].

Baseiam-se no processo genético e evolutivo dos organismos vivos. As populações evoluem de acordo com os princípios de seleção natural e sobrevivência dos mais fortes. São eficientes na busca de "boas" soluções, conforme intervalo ou parâmetros considerados. Uma das vantagens do emprego de algoritmos genéticos é a simplificação na

formulação e solução de problemas de otimização, particularmente, os problemas conhecidos como NP-completos [REE93].

É um algoritmo de pesquisa global que utiliza populações para encontrar um mínimo (máximo) local. A solução de problemas de otimização por algoritmos genéticos, em geral, envolve três fases principais [MIC96]:

- Codificação do problema;
- Definição da Função Objetivo que se deseja maximizar ou minimizar;
- Definição do espaço de soluções associado.

Uma descrição formal do algoritmo é provida em [REE93] e [GOL93].

O pseudo-código geral do GA é mostrado na Figura 27 [REE93]. Inicialmente, temos uma população com P indivíduos (estruturas aleatórias), cada qual representando uma solução válida do problema proposto. O desempenho de cada indivíduo é avaliado com base em uma Função de Aptidão (*fitness)* - $F_{Apt}$ , usada como critério de seleção. $F_{Apt}$ não precisa ser diferenciável. Os melhores indivíduos tenderão a ser progenitores da geração seguinte, melhorando, de geração em geração, por meio das sucessivas trocas de informação. A cada iteração são selecionados pares de cromossomos a partir da população atual (*SelectPar)*, dentro de um determinado critério de aptidão. Indivíduos mais aptos têm maior probabilidade de serem escolhidos para reprodução, permitindo que esses indivíduos possam passar as suas características às próximas gerações. Isto funciona como na natureza, onde os indivíduos altamente adaptados ao seu ambiente possuem naturalmente mais oportunidades para reproduzir do que aqueles indivíduos considerados fracos. Estes pares serão combinados sexualmente para troca de material genético (*crossover*) e formação de novos descendentes, constituindo uma nova população P'. As novas gerações são geradas pela mistura de "material genético dos pais". Temos assim a propagação das características dos indivíduos mais aptos da população por meio de troca de segmentos de informações entre os casais. Alguns descendentes poderão sofrer mutação (*mutate*), para garantir a varredura do espaço de estados e evitar que o algoritmo genético convirja muito cedo para mínimos locais indesejados. A mutação ocorre alterando-se o valor de um gene de um indivíduo sorteado aleatoriamente com uma determinada probabilidade, denominada probabilidade de mutação, ou seja, vários indivíduos da nova população podem ter os seus genes alterados aleatoriamente. A função de aptidão da nova geração é calculada na função

*EvaluatePopulation*. O algoritmo finaliza quando o máximo número de iterações é atingido ou quando há a convergência para uma determinada solução, considerada quase-ótima.

```
Genetic Algorithm
{     t=0;
      P=GenerateInitialPopulation();
      EvaluatePopulation(P);
      while not done do
      {     t++;
            SelectPar(P(t));
            Crossover(P'(t));
            P''(t)=Mutate(P'(t));
            EvaluatePopulation(P''(t));
      }
}
```

*Figura 27: Pseudo-Código básico de um algoritmo genético (GA) [REE93]*

Existem diferentes $F_{Apt}$ para realizar o trabalho de busca. Por exemplo, a função que seleciona pares de cromossomos para realizar o *crossover* - chamada *SelectPar* - pode ser feita de várias formas (aleatoriamente, pelo método da roleta, etc). Estas funções estão disponibilizadas na biblioteca *libGA* [COR05]. Uma boa escolha destes parâmetros e estruturas de GA garantirá uma convergência mais rápida para a melhor solução e poderá evitar a ocorrência de resultados visivelmente indesejados.

Para a garantia de resultados satisfatórios, recomenda-se os seguintes procedimentos [MIC96]:

- Definição de uma Função de Aptidão (*fitness*) adequada para avaliar o quão boa ou ruim é uma determinada solução e garantir uma convergência rápida para uma solução quase-ótima;

- Delimitação do espaço de busca do problema, sempre que possível, dentro de um menor intervalo de valores. O ideal é direcionar o intervalo de busca para os limites direito e esquerdo da solução ótima global. Na prática, isso dificilmente é possível. No entanto, qualquer esforço é válido.

A garantia de resultados satisfatórios também depende da escolha correta de parâmetros e estruturas de GA. Entende-se aqui como parâmetros de GA: tamanho da população, taxa de reprodução, taxa de mutação de cromossomos e número de gerações. Entende-se aqui como estruturas de GA as funções contidas em cada passo do pseudo-código geral de GA mostrado na Figura 27 [REE93].

Os GAs possuem uma larga aplicação em muitas áreas científicas, entre as quais podem ser citados problemas de otimização de soluções, aprendizado de máquina, desenvolvimento de estratégias e fórmulas matemáticas, análise de modelos econômicos, problemas de engenharia, diversas aplicações na Biologia como simulação de bactérias, sistemas imunológicos, ecossistemas, descoberta de formato e propriedades de moléculas orgânicas [MIT97].

Uma inovação do GA, no que se refere ao método de busca, é a implementação de um mecanismo de seleção de soluções no qual, a curto prazo os melhores têm maior probabilidade de sobrevivência e, a longo prazo, têm maior probabilidade de descendência.

Uma diferenciação do GA para outros métodos é o fato de ser um método de busca cega, isto é, não tem conhecimento do problema a ser resolvido, mas apenas da Função de Aptidão $F_{Apt}$ [GOL93] Assim, GAs não trabalham diretamente com o domínio do problema, apenas com representações dos seus elementos, executando a busca em um conjunto de candidatos. Desta forma, pode não apresentar uma linearidade na busca de uma solução quase-ótima.

Uma grande vantagem do GA é o que podemos chamar de "segurança pela quantidade". Obtendo uma boa população de pontos bem adaptados, a chance de obter um mínimo local indesejável é menor em relação a outros métodos que trabalham com um só ponto. Esta amplitude é obtida simplesmente ignorando a informação que não se constitui parte do objetivo, ao contrário de outros métodos, que apóiam-se fortemente em um ponto.

### 3.3.2. Algoritmos Meméticos - MA

Algoritmos meméticos (*memetic algorithms)* – MA's, são um tipo de GA onde a informação necessária para a busca não está somente no mecanismo hereditário, mas também nas unidades de informação (memes), que são compartilhadas e enriquecidas por meio de uma evolução cultural [MOS89], [MOS93].

Os MA's derivam de um classe de GA denominada *knowledge-augmented GA's*, algumas vezes referida como Algoritmos Genéticos Híbridos (AGH). Dentro de uma visão simplificada, MA é um tipo de GA que utiliza uma população constituída de indivíduos que são mínimos locais [MOS89], [MOS93].

O pseudo-código geral do MA é mostrado na Figura 28 [MER99]. O algoritmo inicia com a geração da população inicial (*GenerateInitialPopulation*). A busca local (*LocalSearch*) é aplicada a cada cromossomo como forma de melhoramento "cultural". Em seguida, ocorre a seleção de pares (*SelectPar*) para o cruzamento (*Crossover*). O novo descendente *x* é acrescentado à população *P(t) (AddToPopulation)*. O processo é repetido até um número máximo de recombinações (*MAX_RECOMBINATIONS*) é atingido. A inclusão de descendentes à P(t) torna-lo-á maior. Em *SelectPopulation*, somente os melhores cromossomos são selecionados e o restante é descartado, formando P'(t). Se a distância média de *Hamming* [FAU94] para os cromossomos em P'(t) for menor que uma constante (a ser determinada experimentalmente, em geral, igual a 10, [MIL04]), P'(t) sofreu uma convergência indesejada. O algoritmo retorna ao início do ciclo para determinar novo P'(t). Caso contrário, cada x de P'(t) sofrerá mutação (*Mutate*) e um melhoramento (*LocalSearch*). O algoritmo finaliza quando o máximo número de iterações é atingido ou quando há a convergência para uma determinada solução, considerada quase-ótima.

```
Memetic Algorithm
{       t=0;
        P=GenerateInitialPopulation();
        EvaluatePopulation(P);
        foreach x in P do
            x=LocalSearch(x);
        while not done do
        {       t++;
                for i =1 to MAX_RECOMBINATIONS
                {       SelectPar(  p1 , p2∈P'(t)  );
                        x=Crossover(p1,p2);
                        x=LocalSearch(x);
                        P(t) = AddToPopulation(x);
                }
                P'(t)=SelectPopulation(P(t));
                if Converged(P'(t))
                {       foreach best x in P do
                        {       x=Mutate(x);
                                x=LocalSearch(x);
                        }
                        EvaluatePopulation(P'(t));
                }
        }
}
```

*Figura 28: Pseudo-Código básico de um algoritmo memético (MA) [MER99]*

Uma semelhança entre MA's e GA's é o emprego do mesmo processo de seleção natural. Utilizam a seleção, cruzamento, mutação, etc. O princípio básico consiste em selecionar bons indivíduos, (que mapeiam o domínio de soluções válidas para o problema de otimização), para reprodução e cruzá-los, com o propósito de obter soluções melhores que a atual geração. Esses filhos, por sua vez, tendem a ocupar o lugar dos indivíduos menos adaptados da população, melhorando a adaptabilidade de toda a população como um todo. A mutação entra como um elemento adicionador de variedade genética. Todos esses elementos agindo sobre uma dada população levarão a um processo evolutivo. Desta forma, uma população inicialmente pouco adaptada, ao fim de um certo número de gerações, constituir-se-á na sua maioria de indivíduos bem adaptados.

Uma vantagem dos Algoritmos Meméticos é a agregação do conceito de "evolução cultural" [MOS93]. A adaptabilidade de um indivíduo pode ser modificada no decorrer de sua existência dentro da população. Um indivíduo pode ser geneticamente pouco favorecido ao nascer, mas devido às condições em que vive, por trocas de informação com outros indivíduos e experiências pessoais, entre outros aspectos, pode se

tornar mais adaptado, e mais do que isso, transmitir essa experiência aos seus descendentes.

O termo *meme* [DAW76] significa uma unidade de informação que se reproduz durante um processo argumentativo e de transmissão de conhecimento [RAD94]. Portanto, pode-se dizer que, enquanto o algoritmo memético está relacionado com a evolução cultural, o algoritmo genético está baseado na evolução biológica dos indivíduos.

Uma diferença marcante entre genes e *memes* está no processo de transmissão aos seus descendentes. Quando o *meme* é transmitido, ele será adaptado pela entidade que o recebe com base no seu conhecimento e para melhor atender às suas necessidades. Quanto aos genes, no processo de evolução eles são transmitidos de uma maneira tal que o descendente gerado vai herdar muitas habilidades e características presentes em seus progenitores. Dessa maneira, o algoritmo genético é inspirado na tentativa de emulação computacional da evolução biológica e o algoritmo memético tenta fazer o mesmo em relação à evolução cultural.

### 3.3.3. Ant Colony Optimization Algorithms - ACO

É um tipo de MH inspirado no comportamento orientado das formigas para encontrar o melhor caminho, onde cada indivíduo compartilha a informação com os outros indivíduos acerca de seu caminho aleatório experimentado [DOR92].

A partir de uma população de soluções, cada indivíduo assume o comportamento de uma "formiga". Quando uma formiga anda, deixa uma trilha de um hormônio chamado "feromônio". Ao encontrar um depósito de comida, a formiga retorna com o alimento para o formigueiro, voltando logo em seguida para pegar mais comida. Quanto mais próximo o depósito de comida encontrado estiver do formigueiro, mais forte se torna o caminho. Quando outras formigas saírem da colônia, perceberão este caminho e irão seguí-lo, aumentando o feromônio depositado neste caminho. Nem todas as formigas obedecem a esta regra, contribuindo para uma certa aleatoriedade no método [DOR92].

O pseudo-código do algoritmo é mostrado na Figura 29 [MIL04]. O algoritmo inicia atribuindo a cada "formiga" uma posição inicial (*GenerateInitialPopulation*). As trilhas de feromônios são inicializadas segundo o grau de procura

(*InitialisePheromoneTrails*). Em seguida, cada formiga procura melhorar o seu deslocamento, isto é, a sua solução atual, uma parte seguindo as informações contidas nos feromônios e parte seguindo aleatoriamente (*ApplyAntAlgorithmMove*) (veja Figura 30). A quantidade de feromônios é atualizada, levando em conta a possibilidade de evaporação da substância (*UpdatePheromeTrails*) (veja Figura 31). O algoritmo finaliza quando o máximo número de iterações é atingido ou quando há a convergência para uma determinada solução, considerada quase-ótima.

```
AntAlgorithm(R,q,)
{    //Foreach ant generate a starting solution
     foreach ant i do
     {   x_i  = GenerateInitialStartPoint(); }
         //Intialise pheromone trails
         T = InitialisePheromoneTrails();
         do
         {  foreach ant i do
           { for r=1 to R do  x_i  = ApplyAntAlgorithmMove( x_i ,T);
            }
            UpdatePheromoneTrails(T,x);
         }while (not termination condition)
}
```

*Figura 29: Pseudo-Código básico de um ACO*

```
UpdatePheromeTrails(T, x')
{     //Simulate evaporation of the pheromone trails
      foreach (i,j) such that 1<=i,j<=N do
            T_ij = (1 − α_1) T_ij   ;
      //Reinforce pheromone trails using best solution found so far
      fori = 1 to N do
            T_ix'[i] = T_ix'[i] + α_2 / f(x'_i)
      return T
} //Ant i, pheromone trail T are parameters
```

*Figura 31: Procedimento UpdatePheromeTrails para ACO*

Uma vantagem das ACO's é o emprego em problemas estocásticos com número relativamente baixo de vizinhanças por estado [DOR99]. Por outro lado, problemas com uma vizinhança muito grande em cada estado (por exemplo, grafos totalmente conectados) não são eficientemente solucionados, apresentando tempo elevado de busca por soluções e/ou soluções não-competitivas.

### 3.3.4. Scatter Search -SS

O *Scatter Search* (SS) [CUN97] é um algoritmo evolucionário semelhante aos GA's e MA's. Gera novas soluções combinando as soluções encontradas a cada geração.

A idéia do SS é manter um conjunto de Q soluções elitizadas (*elite solutions*) e combinar os R indivíduos mais aptos da população atual com estas soluções elitizadas. R e Q são parâmetros definidos experimentalmente.

O pseudo-código do algoritmo é mostrado na Figura 32. Inicialmente, a nova população de soluções elitizadas é gerada (*GenerateInitialPopulationOfQSolutions*). A função aleatória utilizada para combinar vários indivíduos em um novo elemento (*GenerateNewSolFrom(sols)*). Se esta nova solução for mais apta que a pior solução elitizada, então esta será substituída. O algoritmo finaliza quando o máximo número de iterações é atingido ou quando há a convergência para uma determinada solução, considerada quase-ótima.

```
ScatterSearch(R,Q)
{       population = GenerateInitialPopulationOfQSolutions();
        do
        {       sols = SelectRBestSolsForCombining(population);
                x = GenerateNewSolFrom(sols);
                population = InsertSolIntoPopulation(population,x);
        }while not termination condition
}
```

*Figura 32: Pseudo-Código básico de um Scatter Search*

Uma vantagem do SS sobre GA's e MA's é o emprego mais controlado das combinações, por meio das soluções elitizadas, minimizando a aleatoriedade.

### 3.3.5. Path Relinking -PR

O *Path Relinking* (PR) [GLO96], [GLO00] é uma técnica que cria caminhos entre soluções e geram a população seguinte a partir das soluções que aparecem nessas posições.

A idéia do PR é procurar boas soluções entre duas boas soluções já encontradas, explorando o caminho entre elas. Parte do princípio que em muitos problemas, boas soluções estão localizadas próximas a outras, segundo o princípio da otimalidade aproximada [BOE94] ou do princípio da paisagem de um grande vale [GLO97].

O pseudo-código do algoritmo é mostrado na Figura 33. A distância de *Hamming* [FAU94] entre dois vetores v,v' de dimensão N é dada por

$$D_H(v, v') = \sum_{i=0}^{N-1} |v[i] - v'[i]|$$

*Equação 2: Distância de Hamming entre dois vetores v,v'*

```
PathRelinking( x1, x2 )
{       x = x1 ;
        x = ( f(x1) < f(x2) ) ? x1 : x2 ;
        while xa • xb and not termination condition do
        {       x = x' from N(x) such that f(x') is minimized and
                hamming distance(x', x2) < hamming_distance(x, x2)
                if (f(x) < f(x*))     x* = x;         }
        return x*;
}
```

*Figura 33: Pseudo-Código básico de um Path Relinking*

O PR está relacionado ao SS. Pode ser agregada à função *GenerateNewSolFrom(sols)*.

Uma vantagem do PR é a simplicidade [MIL04]. Pode ser usado como um procedimento para recombinação de soluções em SS e para aprimorar GA's, como pode ser encontrado em [REE98] e em [RES03b].

## 3.4. NEIGHBORHOOD BASED ALGORITHMS - NBA

NBA são metaheurísticas que restringem ou expandem vizinhanças de soluções ao processo de busca para escapar de ótimos locais. São exemplos desta categoria:

- *Variable Neighborhood Search (VNS),*
- Busca Tabu (BT).

### 3.4.1. Variable Neighborhood Search -VNS

O VNS [MLA97] é uma técnica que efetua uma busca local que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhanças. A idéia é testar os vizinhos até encontrar um ótimo local, a partir do qual cria uma vizinhança aumentada e reinicializa o processo. Focaliza a busca em torno de uma nova solução somente se um movimento de melhora é realizado.

O pseudo-código do algoritmo é mostrado na Figura 34. Uma solução x é gerada inicialmente (*GenerateInitialSolution*). Gera-se também $x_1 \in N_{open}(x_2)$ (*RandomSolutionPickedFromNx(N(x))*). Um melhoramento local é realizado (*LocalSearch*). Se $f(x_2) < f(x_1)$ , $x_2$ torna-se a melhor solução encontrada e a busca é iniciada em torno de $N_{open}(x)$ , senão k é incrementado. O algoritmo finaliza quando o máximo número de iterações (*KMAX*) é atingido ou quando há a convergência para uma determinada solução, considerada quase-ótima.

```
VariableNeighbourhoodSearch
{       x = GenerateInitialSolution();
        k = 1;
        do
        {   x1 = RandomSolutionPicked FromNx(N(x));
            x2 = LocalSearch(x1);
            if (f(x2) < f(x1))
            {   x = x2;
                k = 1;//improved solution => use smallest N(x)
            }
            else  if (k < KMAX)   //no improved solution found
                    k++;      //use next largest neighbourhood
        }while (not termination condition)
}
```

*Figura 34: Pseudo-Código básico de um VNS*

Por ser um NBA e ao contrário de outras MH baseadas em busca local, o VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno de uma nova solução, se um movimento de melhora é realizado.

### 3.4.2. Busca Tabu - BT

Busca Tabu (BT) [GLO89], [GLO90], [GLO97] é uma técnica de busca local que utiliza uma lista de movimentos proibidos para guiar a exploração do espaço de soluções, mesmo na ausência de movimentos de melhora, evitando que haja a formação de ciclos, isto é, o retorno a um ótimo local previamente visitado.

A idéia do BT é imitar o processo de memória dos seres humanos. Não somente a memória imediata (local), mas também a memória de mais longa temporalidade. Esta última está relacionada a uma parte da extensão do processo de exploração do algoritmo, enquanto aquela está relacionada ao valor da solução corrente. Desta forma, o algoritmo mantém o caminho percorrido até a presente solução. Esta informação é utilizada para guiar o algoritmo no movimento de uma solução para uma outra vizinhança no processo de busca por um resultado quase-ótimo. A memória na BT restringe algumas possíveis opções vizinhas, a fim de evitar movimentos cíclicos.

Os movimentos realizados recentemente são "memorizados" na lista de soluções proibidas, chamada de *lista tabu*. O uso de memória é a característica principal da BT. Porém, ao proibir movimentos para tentar evitar o retorno a uma solução já visitada, pode acontecer também desse mesmo movimento resultar numa solução nunca antes visitada. Nesses casos, é preciso a ativação de um módulo que fiscalize o papel das *listas tabu*. Este módulo é conhecido como critério de aspiração [GLO97].

Em sua forma clássica, gera-se uma solução inicial $s_0$ . A cada iteração seleciona-se o melhor vizinho de $s_0$ , dentro de um subconjunto V, $V \subseteq N_{open}(s_0)$ , $s'$ , com $s' \in V \subseteq N_{open}(s_0)$ , tal que $\forall s'' \in N_{open}(s_0), f(s') < f(s'')$ . A solução $s'$ torna-se, então, a melhor solução corrente mesmo que seja pior que $s_0$ $(f(s') > f(s_0))$ .

A escolha "para pior" de um vizinho maximal tem o objetivo de escapar de um

mínimo local indesejado. Há, entretanto, a possibilidade de formação de caminhos cíclicos, isto é, o retorno para uma solução já gerada anteriormente. Para isto, é utilizada a *lista tabu*. Ela contém uma lista em ordem reversa dos últimos $|T|$ movimentos realizados, considerados proibidos. O parâmetro $|T|$ é definido experimentalmente. Funciona como uma fila de tamanho fixo. Quando um novo movimento é adicionado à lista, o mais antigo sai. Assim, na exploração de um subconjunto $V \subseteq N_{open}(s)$ da solução corrente $s$, ficam excluídos da busca os vizinhos de $s$ que são obtidos por movimentos *m* contidos na *lista tabu* [GLO89].

A busca tabu apresenta um mecanismo que retira o *status tabu* de um movimento, sob determinadas circunstâncias, chamado de *Função de Aspiração* $A(v)$. Esta função é aplicada a cada vizinho da solução corrente. Para cada possível valor *v* da função objetivo existe um nível de aspiração $A(v)$. Na BT, *v* é representado por $s'$, vizinho da solução corrente $s_0$, onde $s' \in V \subseteq N_{open}(s_0)$. Uma solução $s'$ poderá ser escolhida, mesmo que $m$ esteja na lista tabu, se $f(s') \leqslant A(f(s*))$. Logo, a *Função de Aspiração* $A(v)$ representa o valor que o algoritmo aspira ao chegar em *v*. Como exemplo, considere $A(f(v)) = f(s_*) - 1$, onde $s_*$ é a melhor solução encontrada até agora. Como a solução conduz a uma melhor minimização, o *movimento tabu* é aceito.

O pseudo-código do algoritmo é mostrado na Figura 35 [GLO89], [MIL04]. *A*, *LT* e *ITER_MAX* são parâmetros de controle de BT. Uma solução $s_0$ é gerada (*GenerateInitialSolution*) e atribuída como melhor solução até o momento $(s^* \leftarrow s_0)$. A condição de parada *termination_condition* é dada pelo parâmetro *ITER_MAX*. A cada iteração, é escolhido o melhor vizinho $s'$ de $s_0$, conforme o seguinte critério:

- Melhor função de aptidão $f(s')$, sendo $\forall s'' \in V \subseteq N_{open}(s_0)$ $f(s') \leqslant f(s'')$, se o movimento *m*, $s' \leftarrow s_0 \circ m$, que está associado a $s'$ não estiver na *lista tabu*, evitando assim recorrer a movimentos cíclicos indesejados;
- Caso *m* esteja na *lista tabu*, analisar-se-á a *Função de Aspiração* (*A(v)*), $\{f(s') \leqslant A(f(s*))\}$, na tentativa de retirar o *status tabu* do movimento.

```
TabuSearch()
{
        s0  = GenerateInitialSolution(); //initial solution
        s* ← s0  ;                       //best solution
        Iter ← 0  ;                      //iteration counter
        BestIter ← 0  ;
        LT ← Ø  ;                        //tabu list is empty
       A = InitializeAspirationFunction(); //A(v)

      while (termination condition) do
      {   Iter++;
          Choose   s' ← s0 ∘ m | [∀ s'' ∈ V ⊆ N_open(s0), f(s') ≤ f(s'')]
                   and  [f(s') ≤ A(f(s*))]  ;
          UpdateTabuList(LT, s0, s');
            s0 ← s'  ;
          if   (f(s0) < f(s*))
          {      s* ← s0  ;
                 BestIter ← Iter  ;
          }
          UpdateAspirationFunction(A);
      }
      return   s*  ;
}
```

*Figura 35: Pseudo-Código básico da Busca Tabu [MIL04]*

Em seguida, a *lista tabu* é atualizada (*UpdateTabuList*). Como LT é uma fila de tamanho fixo, o novo movimento é adicionado à memória (LT) e a informação mais antiga é excluída. A solução $s'$ torna-se a solução atual $s_0$. Se a solução corrente $s_0$ for mais apta que a melhor solução encontrada $s^*$ , então $s_0$ para a ser a melhor solução e *BestIter* é atualizado. O algoritmo finaliza pelo parâmetro *ITER_MAX*. A melhor solução encontrada estará em $s^*$.

Uma estratégia para otimizar a estrutura de memória é baseada na freqüência dos movimentos [GLO97]. Este tipo de memória pode ser utilizado, por exemplo, para proibir movimentos que tenham sido muito freqüentes no histórico da busca, proporcionando, assim, maior diversificação.

Uma variante da busca tabu é incluir técnicas de *Intensificação* [HER92]. Objetiva-se concentrar a pesquisa em determinadas regiões consideradas promissoras. Pode ser empregado para explorar melhor a vizinhança de melhores soluções já visitadas $s^*$ . Outra opção de emprego é incorporar atributos das melhores soluções já encontradas durante o progresso da pesquisa e estimular componentes dessas soluções a tomarem parte

da solução corrente. Nesse caso, são consideradas livres no procedimento de busca lcoal apenas as componentes não associadas às boas soluções, permanecendo as demais componentes fixas.

Outra variante da busca tabu é aplicar estratégias de *Diversificação* [HER92]. Objetiva-se direcionar a pesquisa para regiões ainda não suficientemente exploradas em iterações anteriores, dentro do espaço de soluções. Ao contrário da Intensificação, gera soluções significativamente diferentes das atuais. É utilizada somente em determinadas situações. Por exemplo, quando obtém-se uma solução $s^*$ e o algoritmo já exauriu a sua busca, após MAX_ITER iterações para esta solução. Se quisermos executar uma procura em outra região, basta atribuir uma penalidade $w(s, m)$ em cada movimento *m*, associada a uma solução *s*.

Uma problema encontrado na BT é o gerenciamento da *lista tabu* [HER92]. Se a lista tabu for muito pequena, a probabilidade de minimização de caminhos cíclicos torna-se menor. Por outro lado, se for muito grande, pode se tornar computacionalmente inviável analisar todas as possibilidades da *lista tabu*. Uma opção intermediária é armazenar apenas as últimas $|T|$ soluções geradas, para evitar caminhos cíclicos em até $|T|$ iterações.

## 3.5. WEIGHTED AND PENALTY BASED ALGORITHMS - WPBA

WPBA são metaheurísticas que usam penalidades ou pesos para modificar a função objetivo do processo de busca para escapar de ótimos locais indesejados. Um exemplo desta categoria é o algoritmo *Guided Local Search (GLS)* [VOU97].

# 4-MOBILIDADE E MÉTRICAS DE DESEMPENHO EM REDES *AD HOC*

## 4.1. INTRODUÇÃO

Os modelos de mobilidade em redes *ad hoc* são utilizados para descrever o padrão de movimento de uma rede de nós em uma topologia $G$ , sua localização, velocidade e aceleração em função de uma unidade de tempo. É desejável que um modelo de mobilidade represente padrões de movimento próximos à realidade que será simulada. Particularmente, a adoção de modelos de mobilidade mais realísticos irá refletir no desempenho do algoritmo de particionamento em *clusters*. Estes modelos influenciam diretamente no desempenho do roteamento das informações que trafegam na rede. Neste sentido, diferentes trabalhos foram publicados apresentando conclusões semelhantes sobre o assunto [GER99b], [GER01], [DAV02],[BAI03b].

O estudo sobre modelos de mobilidade em redes *ad hoc* apresenta escopo "microscópico" [BAI03]. A individualidade dos nós é levada em consideração para a modelagem, análise, localização e velocidade relativa entre nós em $G$ . Isto ocorre porque estes fatores estão diretamente ligados à determinação de rotas e determinação de *clusters*.

Uma abordagem contrária pode ser encontrada em uma rede celular, onde adota-se uma visão "macroscópica" [MAR97], [LAM99], onde o parâmetro de referência não são os nós, mas sim, uma determinada área (ou célula). Neste novo nível, são levados em consideração, por exemplo, os movimentos de unidades móveis (UM) de uma célula para outra dentro de uma mesma MTSO (*handoff* ou *handover intra-switch*), movimentos de nós de uma célula para outra com mudança de MTSO (*handoff inter-swich ou roaming*), probabilidade de bloqueio de chamada, probabilidade de encerramento de chamada.

## 4.2. MOBILIDADE EM REDES AD HOC

Serão apresentados alguns modelos de mobilidade para redes *ad hoc*:

- *Random Walk Mobility Model,*
- *Random Waypoint Mobility Model,*
- *Boundless Simulation Area Mobility Model,*
- *Smooth Random Mobility Model,*
- *Reference Point Mobility Model*

Considerar-se-á que a mudança de direção $\theta$ ocorrem no intervalo $[0,2\pi]$ no sentido horário (círculo trigonométrico).

### 4.2.1. Random Walk Mobility Model - RWaMM

Este modelo é baseado no Movimento *Browniano* [SAN01], descrito matematicamente para modelar movimentos imprevisíveis de partículas na Física.

O objetivo é imitar os nós que se movem por caminhos imprevisíveis, ou seja, possuem movimentos instáveis e mudam sua velocidade e direção a cada intervalo de tempo $t$ , sendo distância e *t* constantes. Cada nó *i* escolhe aleatoriamente uma nova direção $\theta_i(t)$ ,no intervalo $\theta_i(t) \in [0,2\pi]$ e uma nova velocidade $v_i(t)$ com distribuição uniforme em $v_i(t) \in [0, v_{max}]$ . Desta forma, o nó se move com uma velocidade definida pelo vetor $v_i = (v(t)\cos\theta(t), v(t)sen\theta(t))$ [DAV00]. A Figura 36 mostra o deslocamento de um nó A em $t = [0..4]$ , observando-se que mudanças de direção e velocidade ocorrem aleatoriamente, isto é, os binômios $v_0, \theta_0$ , $v_1, \theta_1$ , $v_2, \theta_2$ e $v_3, \theta_3$ não têm relação entre si.

Trata-se de um modelo "sem memória", porque não leva em consideração nem armazena estados anteriores de localização e velocidade [HAA99]. As mudanças de $\theta_i(t)$ e $v_i(t)$ ocorrem independentemente dos resultados de $\theta_i(t-1)$ e $v_i(t-1)$ , causando movimentos e paradas bruscas.

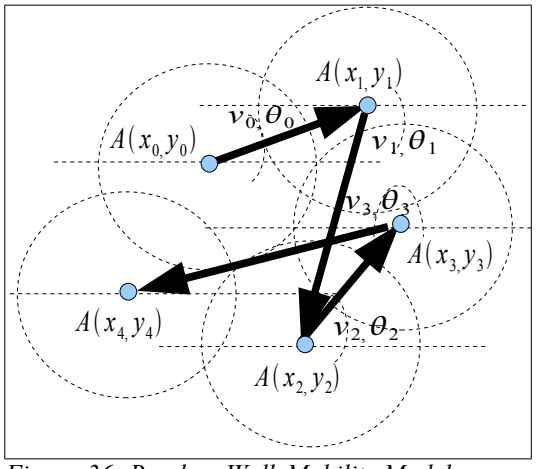


*Figura 36: Random Walk Mobility Model aplicado a um nó A*

Apresenta como vantagem a simplicidade na implementação e análise dos caminhos dos nós móveis. Por outro lado, tem com desvantagem não representar adequadamente cenários mais realísticos [DAV02].

### 4.2.2. Random Waypoint Mobility Model - RWyMM

Este modelo é semelhante ao *Random Walk Mobiliy Model*, porém com tempo de pausa [JOH96].

O objetivo é imitar os nós que se movem por caminhos imprevisíveis, ou seja, possuem movimentos instáveis e mudam $v$ e $\theta$ a cada intervalo $t$ , mas apresentando um tempo de pausa $t_{pause}$ . Em $t_0 = 0$ , um nó $i$ recebe $(x_0, y_0)$ e $t_{pausa} = T$ . Após $T$ ,gera-se de forma aleatória e distribuída uniformemente $\theta_i(t_1)$ , $\theta_i(t_1) \in [0, 2\pi]$ e $v_i(t_1)$ , $v_i(t_1) \in [v_{min}, v_{max}]$ . O nó $i$ desloca-se para o novo destino $(x_1, y_1)$ dentro da área de simulação e realizará novo movimento após permanecer lá por $t_{pausa} = T$ . A Figura 37 mostra o deslocamento de um nó A em $t = [0..4]$ . A permanência em uma posição apresenta um tempo de pausa $t_{i\,pausa}$ . As mudanças de direção e velocidade ocorrem aleatoriamente, isto é, os binômios $v_0, \theta_0$ , $v_1, \theta_1$ , $v_2, \theta_2$ e $v_3, \theta_3$ não têm relação entre si.

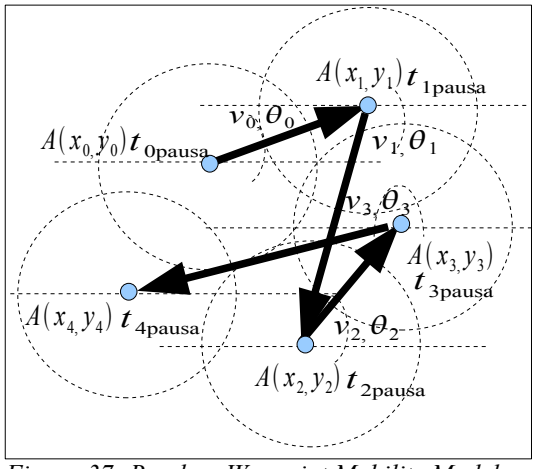


*Figura 37: Random Waypoint Mobility Model aplicado a um nó A*

O *Random Waypoint* torna-se igual ao *Random Walk Mobility Model* se $t_{pausa} = 0$ [JOH96], [BRO98].

Apresenta como vantagem a simplicidade na implementação. É amplamente usado para realizar comparações com outros modelos de mobilidade em redes *ad hoc* [BRO98], [GER98] , [GAR99], [JOH99].

### 4.2.3. Boundless Simulation Area Mobility Model - BSAMM

Este modelo apresenta uma relação entre o estado anterior e o atual, em termos de velocidade e direção [HAA97b].

A velocidade $v_i(t)$ , $v_i(t) \in [v_{min}, v_{max}]$ e direção $\theta_i(t)$ , $\theta_i(t) \in [0, 2\pi]$ de *i*, em $(x, y)$ são modificados a cada intervalo $\Delta(t)$ de acordo com os seguintes padrões:

$$v(t+\Delta t) = min[max[v(t)+\Delta v, 0], v_{max}] \text{ e}$$

$$\theta(t+\Delta t) = \theta(t)+\Delta\theta \ ,$$

onde $x(t+\Delta t) = x(t)+v(t)*\cos\theta(t)$ , $y(t+\Delta t) = y(t)+v(t)*sen\theta(t)$ , $v_{max}$ é a velocidade máxima definida na simulação, $\Delta v$ e $\Delta\theta$ são as variações de velocidade e direção uniformemente distribuídos, com $[-a_{max}*\Delta t, a_{max}*\Delta t]$ e $[-\alpha_{max}*\Delta t, \alpha_{max}*\Delta t]$ , respectivamente, onde $a_{max}$ e $\alpha_{max}$ são a aceleração máxima e aceleração angular máxima de *i* em $t$ .

A área de simulação possui o formato de um toróide, isto é, se um nó atingir o limite de borda, ele reaparecerá no lado oposto da área, segundo a regra de borda *Wrap Around* [BET01]. A Figura 38 mostra um exemplo do movimento de um nó *i* segundo o BSAMM, segundo uma representação plana da área de simulação. Uma representação do terreno em formato de toróide é visto na Figura 39. Nesta ilustração, uma seção transversal aparece em destaque para mostrar a trajetória do nó A em $t=[0..4]$ .

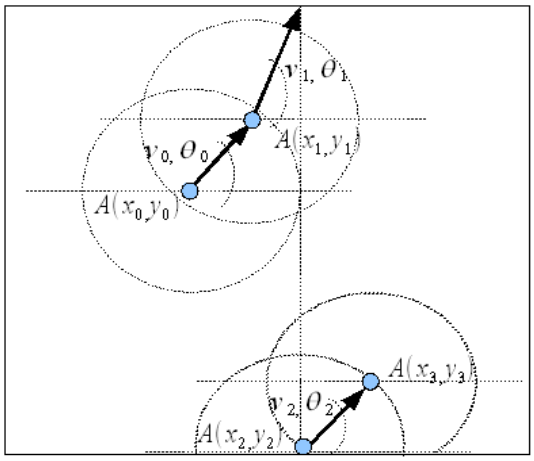


*Figura 38: Boundless Simulation Area Mobility Model aplicado a um nó A em área planar*

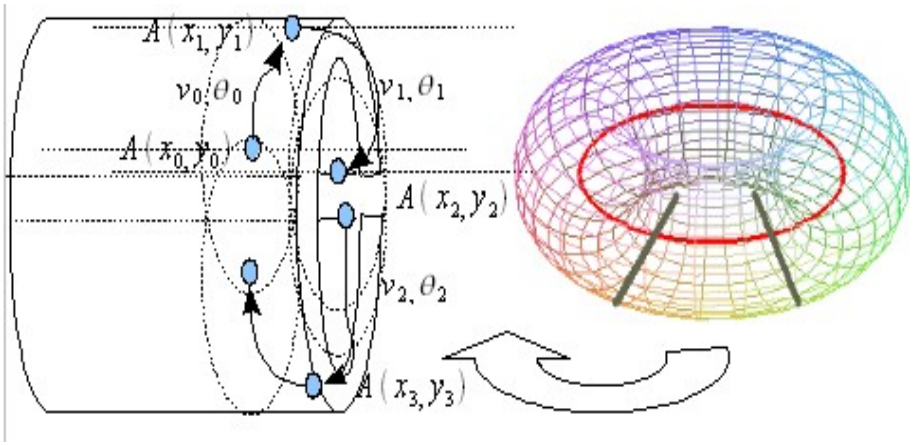


*Figura 39: Boundless Simulation Area Mobility Model aplicado a um nó A em um toróide*

### 4.2.4. Smooth Random Mobility Model - SRMM

Este modelo considera a dependência temporal de velocidade sobre a variação de tempo. As mudanças de velocidade e direção de um nó ocorrem de forma incremental e suave [BET01b].

O objetivo é simular um comportamento mais realístico, onde na vida real observa-se que os nós móveis tendem a se mover dentro de velocidades "preferidas" $\{v_0^{pref}, v_{1,}^{pref} \ldots, v_n^{pref}\}$ , ao contrário de outros modelos onde a velocidade é simplesmente distribuída de forma uniforme no intervalo $[0, v_{max}]$ .

A mudança de velocidade segue um processo de *Poisson* da seguinte forma :

$$p(v)=\begin{cases} p(v=v_0^{pref})\,\delta(v-v_0^{pref}) & v=v_0^{pref} \\ p(v=v_1^{pref})\,\delta(v-v_1^{pref}) & v=v_1^{pref} \\ p(v=v_2^{pref})\,\delta(v-v_2^{pref}) & v=v_2^{pref} \\ \dfrac{1-p(v=v_0^{pref})-p(v=v_1^{pref})\,p(v=v_2^{pref})}{v_{max}} & 0<v_{max}<1 \\ 0 & otherwise \end{cases}$$

,onde $p(v=v_0^{pref})+(v=v_1^{pref})+(v=v_2^{pref})<1$ e considerando três velocidades preferenciais $\{v_{0,}^{pref}\, v_{1,}^{pref}\, v_2^{pref}\}$ . A variação de aceleração segue uma distribuição uniformemente variada, no intervalo $[a_{min}, 0, a_{max}]$ . A velocidade em um tempo $t+\Delta t$ é dada então por $v(t+\Delta t)=v(t)+a(t+\Delta t)\Delta t$ .

A mudança de direção é dado simplesmente por uma distribuição uniformemente variada no intervalo $[0,2\pi]$ :

$$p(\theta)=\frac{1}{2\pi} \quad ,0\le\theta\le 2\pi$$

Uma vez escolhida a direção, a diferença de direção $\Delta\theta$ , entre a nova direção $\theta(t)$ e a direção anterior $\theta(t-1)$ é dada por:

$$\Delta\theta(t)=\begin{cases} \theta(t)-\theta(t-1) & ,-\pi<\theta(t)-\theta(t-1)\le\pi \\ \theta(t)-\theta(t-1)+2\pi & ,-2\pi<\theta(t)-\theta(t-1)\le-\pi \\ \theta(t)-\theta(t-1)-2\pi & ,\pi<\theta(t)-\theta(t-1)\le 2\pi \end{cases}$$

Como as mudanças de direção devem ocorrem de forma suave, como sugere o nome do modelo (*smooth*), $\Delta\theta$ não pode ser aplicado bruscamente em uma unidade de tempo. Para evitar que isso aconteça, a unidade de tempo unitária $\Delta(t)=t-(t-1)$ é dividida em unidades de tempo menores

$$\sum_{i=0}^{n}\varphi_i(t/n)=\{\varphi_0(t/n),\varphi_1(t/n),\ldots,\varphi_n(t/n)\} \quad ,$$

onde *n* é o número de frações sobre a unidade de tempo *t*. A cada $\varphi_i(t/n)$ é aplicado parte da variação de direção $\Delta\theta(t)$ e o nó irá mudar de direção de forma menos brusca e ao final de $\varphi_n(t/n)$ , sua mudança de direção estará completada.

A Figura 40 mostra o SRMM aplicado em três nós A, B, C. O nó $A(x_{0,}y_0)$ ocorre a mudança de direção para $A(x_{1,}y_1)$ em $\theta_0=+\pi/4$ . A variação $\Delta\theta_A=+\pi/4$ em uma unidade de tempo *t* é distribuída dentro de frações menores de tempo $\Delta\varphi_i(t/n) = \varphi_i(t/n)=\{\varphi_0(t/n),\varphi_1(t/n),...,\varphi_4(t/n)\}$ , com $n=5$ . Para B, temos $\theta_1=-\pi/4$ e $\Delta\theta_B=-\pi/4$ . Para C, temos $\theta_2=2\pi$ e $\Delta\theta_C=(2\pi)-2\pi=0$ , pois $\pi<\Delta\theta\leq 2\pi$ .

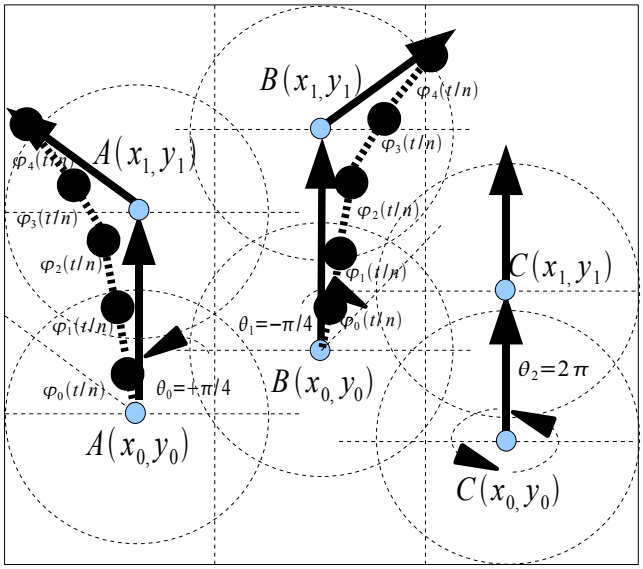

*Figura 40: Smooth Random Mobility Model aplicado a nós A,B e C*

Uma vantagem deste modelo é ser mais realístico por levar em consideração as leis clássicas da física para aceleração, velocidade e mudança de direção. Cada variação é dependente de um estado anterior em função do tempo. Este modelo apresenta dependência temporal [BET01b].

### 4.2.5. Reference Point Mobility Model - RPGM

Este modelo representa o movimento aleatório aleatório individual de cada nó móvel associado ao movimento aleatório de um grupo [GER99b]. Em alguns cenários realísticos, tais como em campos de batalha ou em equipes de resgate, temos a presença de equipes que seguem um determinado líder, cujo movimento influencia o deslocamento dos

seus seguidores.

O objetivo, como sugere o nome *reference point*, é criar em cada grupo um ponto de referência, podendo ser um nó líder de grupo ou simplesmente o centro geográfico. Cada referência contém seus membros. O movimento do líder $\overrightarrow{v_{group}(t)}$ determina a direção geral dos seus membros. A velocidade de um membro *i* em *t*, $\overrightarrow{v_i(t)}$ , é descrito da seguinte forma: $\overrightarrow{v_i(t)}=\overrightarrow{v_{group}(t)}+\overrightarrow{RM_i(t)}$ , onde $\overrightarrow{RM_i(t)}$ é um vetor distribuído uniformemente e representa o desvio de direção em relação ao líder no intervalo $[0,2\pi]$ .

A Figura 41 mostra o RPGM aplicado em três nós A, B e C. O nó $A(x_{0,}y_0)$ é líder de $B(x_{0,}y_0)$ e $C(x_{0,}y_0)$ no instante $t_0$ . O movimento para $A(x_{1,}y_1)$ apresenta velocidade $\overrightarrow{v_A(t)}$ , que é a velocidade de grupo $\overrightarrow{v_{group}(t)}=\overrightarrow{v_A(t)}$ . O deslocamento de B e C é composto por $\overrightarrow{v_{group}(t)}$ e uma componente aleatória uniformemente distribuída $\overrightarrow{RM_B(t)}$ e $\overrightarrow{RM_C(t)}$ para compor $\overrightarrow{v_B(t)}$ e $\overrightarrow{v_C(t)}$ , respectivamente.

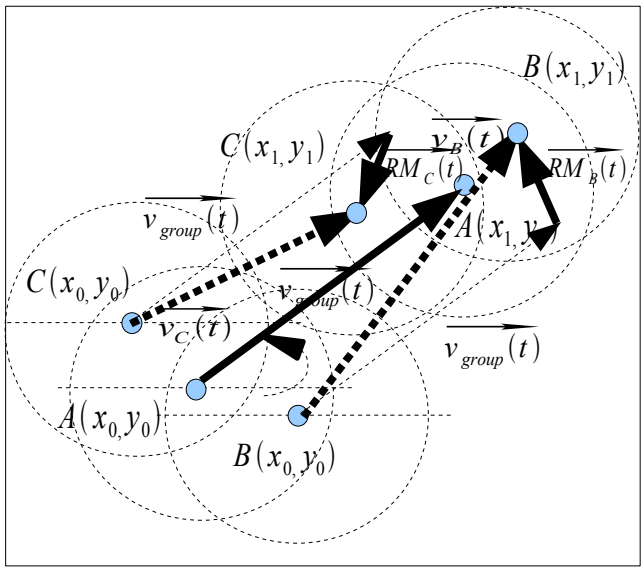


*Figura 41: Reference Point Mobility Model aplicado a nós A(líder), B e C (membros)*

Uma vantagem em relação ao RWaMM é o desempenho superior e a existência de uma correlação de velocidade e direção entre nós distintos é chamado de Dependência Espacial [GER99b].

## 4.3. REGRAS DE BORDA EM REDES AD HOC

O cenário de simulação utiliza uma área limitada para a representação da mobilidade de uma topologia G. Seus limites são chamados de bordas ou fronteiras. Regras de Borda são critérios para decidir o que fazer quando um nó atinge uma borda. Algumas regras de borda: *bounce* [HAA98],[BET01], *delete and replace* [BET01] e *wrap around* [BET01].

### 4.3.1. Bounce

A regra de borda *bounce* [HAA98],[BET01] é uma reflexão do movimento de um nó $n=(v,\theta)$ que atinge a borda do cenário de simulação. Se o nó atingir as bordas superior ou inferior, o novo valor da direção será $-\theta$ e se atingir as laterais, será $\Phi-\theta$ . A velocidade $v$ não é alterada, conforme Figura 42.

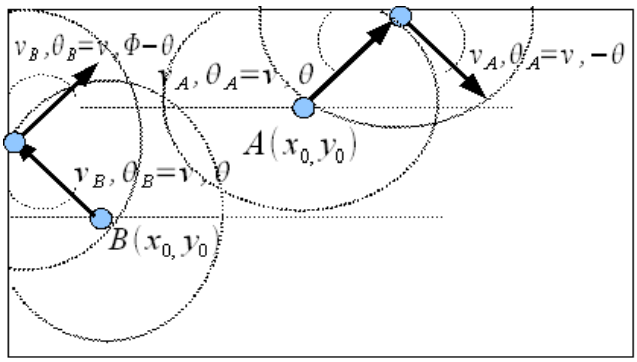


*Figura 42: Regra de borda bounce para nós A e B*

### 4.3.2. Delete and Replace

A regra de borda *delete and replace* [BET01] determina que quando um nó atinge a borda é retirado do cenário é substituído por um novo nó, que será inserido aleatoriamente dentro da área de simulação. Com isso, busca-se representar situações realísticas como entrada e saída de pessoas em uma sala, prédio ou *shopping* e movimentação de carros em rodovias, conforme Figura 43.

### 4.3.3. Wrap Around

A regra de borda *wrap around* [BET01] é um espelhamento do movimento de um nó $n=(v,\theta)$ que atinge a borda do cenário de simulação. Seu movimento e direção não são alterados. , conforme Figura 44.

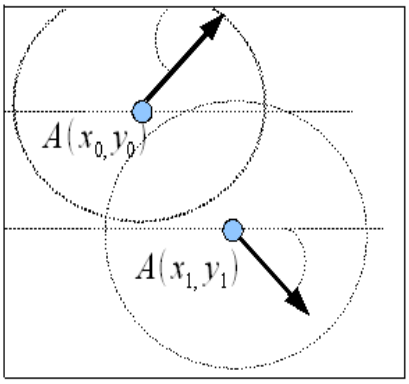


*Figura 43: Regra de borda delete and replace para nó A*

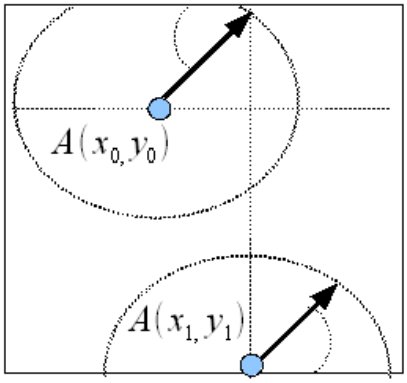


*Figura 44: Regra de borda wrap around para nó A*

## 4.4. MÉTRICAS DE DESEMPENHO

Modelos de mobilidade distintos apresentam comportamentos, algumas vezes, bem diferentes. O estabelecimento de métricas de desempenho mensura estas diferenças. Para avaliar um modelo é inadequado utilizar apenas um métrica de desempenho [ZHE04]. Algumas métricas são apresentadas nesta seção para medição de desempenho em uma rede *ad hoc*:

- Número de reafiliações,
- Fluxo de dados *inter-cluster*/*intra-cluster,* e
- Número de atualizações do *dominant set*.

Outras métricas de desempenho: velocidade média relativa (*average relative speed*) [JOH99b], taxa de mudança de link (*link change rate*) [GER99b] ,[GER01], duração média do link (*average link duration*) [BOL02],[BAI03b],[BAI03c] e duração média do caminho (*average path duration*) [BAI03c]. Sobre estes modelos, algumas correlações podem ser encontradas. [GER01] demonstrou que se a velocidade média

relativa aumenta, a taxa de mudança de link aumenta também. [BOL02] provou que se a duração média do link aumenta, tem-se um aumento do *throughput*, para RWyMM.

### 4.4.1. Número de Reafiliações

Uma reafiliação ocorre quando um *clusternode* se move e abandona o *cluster* atual e se torna membro de outro *cluster* dentro de DS(G) [CHA97].

### 4.4.2. Fluxo de Dados Intra/Inter-Cluster $\Psi$

É a diferença entre o fluxo de comunicação *intra* e *inter-cluster*, isto é, $\psi = \psi_{intra} - \psi_{inter}$ . O fluxo de dados *inter-cluster* ocorre entre *clusterheads*, graças à capacidade de transmissão *dual* dos nós, em baixa energia dentro do *cluster* e em alta entre *clusters*. Isto acontece quando há a minimização da comunicação *inter-cluster* e a maximização da comunicação *intra-cluster*, isto é, $\psi$ negativo. Essa é uma das metas do algoritmo proposto, isto é minimizar a *inter-cluster*.

### 4.4.3. Número de Atualizações do Dominant Set

Uma atualização do *dominant set* é realizada por meio de uma nova execução do algoritmo gerador da topologia. Isto acontece quando um nó se move, abandona o *cluster* atual e não ingressa em uma nova vizinhança de um *clusterhead* [CHA97].

## 4.5. DISPONIBILIDADE

Dada a natureza limitada de recursos de uma rede *ad hoc*, um modelo realístico deve considerar também a possibilidade de falhas dos elementos da rede. Cada nó de uma topologia G apresenta um grau de disponibilidade[HWA98], dado por

$$\zeta_i = \frac{mtbf_i}{mtbf_i + mttr_i} \quad ,$$

*Equação 3: Disponibilidade*

onde *mtbf_i* é o tempo médio entre falhas e *mttr_i* é o tempo médio de falha. A cada unidade de tempo é verificado se um nó *i* está disponível ou não. O mesmo teste é feito se um nó i indisponível na unidade de tempo anterior já retornou ao estado disponível.

# 5-MODELO PROPOSTO PARA OTIMIZAÇÃO DINÂMICA DE CLUSTERING EM REDES AD HOC

## 5.1. INTRODUÇÃO

Redes *ad hoc* não apresentam infra-estrutura fixa de comunicação. Uma das estratégias para a comunicação entre os nós e a manutenção de mudanças de conexão é a adoção de uma estrutura hierárquica baseada em *clusters* [EPH87], [GER95], [RAM98]. Qualquer nó pode se comunicar com qualquer outro nó seja dentro de um mesmo *cluster* ou fora do mesmo. Além disso, um bom critério de particionamento deve ter condições de preservar a comunicação entre os nós à medida que os mesmos realizam movimentações no terreno.

Um problema encontrado neste tipo de arquitetura é a impossibilidade de obter uma solução ótima em tempo polinomial [BAS97], [REE93].

O presente trabalho propõe os algoritmos *GAdHoc* [GAR05], *SAdHoc* e *TSAdHoc*, a partir de técnicas baseadas em metaheurísticas, respectivamente: algoritmos genéticos [HOL75], *simulated annealing* [KIR83] e busca tabu [GLO89], [GLO90], [GLO97], com o objetivo de realizar o particionamento de *clusters* em uma rede *ad hoc*, de forma dinâmica, em um determinado instante de tempo T. Estas técnicas foram escolhidas por serem as mais populares dentro de suas categorias (PBA, RMBA e NBA, respectivamente) e por estarem em classificações distintas.

Dentro do escopo do modelo proposto, leva-se em consideração a possibilidade de movimentação dos nós segundo modelos de mobilidade (RWaMM [SAN01], RWyMM [JOH96], BSAMM [HAA97b], RPGM [GER99b]), regras de borda (*bounce* [HAA98], [BET01]), possibilidade de *handoff inter-cluster* por perda de potência em relação ao *clusterhead*, *handoffs* por diferença de potência entre *clusterheads* interno e externo segundo um limiar – *threshold*, com agregação de histerese, exaustão de um nó por sobrecarga de transmissão e possibilidade de falha de nós da rede, consumo de energia, capacidade de transmissão e disponibilidade dos nós da topologia, com o objetivo de tornar

o ambiente de simulação mais realístico.

Por se tratar de um modelo dinâmico, avalia a possibilidade de rearranjos de topologia, segundo parâmetros de desempenho. A estratégia de minimização de fluxo *inter-cluster* é vista como o principal objetivo a ser alcançado, bem como, atualizações do DS(G) e número de reafiliações.

## 5.2. ARQUITETURA PROPOSTA

Seja uma rede constituída pelo grafo *G=(V,E)*, formado por nós de um conjunto *V*, onde cada nó possui um número de identificação distinto (ID), e de um conjunto de arestas *E*, formada pelas ligações entre nós quaisquer, desde que alcançáveis entre si (com distância menor que um raio de transmissão máximo RAIOMAX) e com fluxo de comunicação não-nulo.

A idéia básica é dispor nós em um cenário (classe *Cenário*) e determinar uma topologia para esta rede (métodos da classe *Cenário – cenario.determinaTopologia()*), isto é, realizar o particionamento em *clusters*, definindo o *dominant set, DS(G)*, e suas respectivas $N(v)_{open}$ de cada líder *v* (classes *Cluster, Clusternode e Clusterhead*), segundo algum método metaheurístico (classe-pai *Metaheuristica*), a citar: algoritmo genético (classe-filha *AlgoritmoGenetico*), *Simulated Annealing* (classe-filha *SimulatedAnnealing*) e Busca Tabu (classe-filha *TabuSearch*). A cada instante de tempo *t*, os nós se movem, dentro de um modelo de mobilidade de grupo ou não, perdem energia, podem falhar, transmitem dados, perdem, mantêm ou adquirem comunicação com outros nós, dentro de um raio de transmissão (RAIOMAX) (métodos da classe *Cenário – cenario.realizaDinamicaNos()*). Por isso, não necessariamente uma topologia gerada anteriormente será a mais adequada para o instante de tempo presente. Em cada *t,* é feita uma avaliação para verificar se é compensador determinar uma nova topologia ou não (métodos da classe *Cenário – cenario.analisaDinamicaNos()*), visto a formação de um novo arranjo de *DS(G)* e $N_{open}(clusterheads)$ também apresenta custo para a rede. O tempo de simulação é limitado a um tempo máximo TEMPOMAX. A simulação propriamente dita é desencadeada pela classe *ProgramaPrincipal*.

A Figura 45 representa o diagrama de classes básico do modelo proposto:

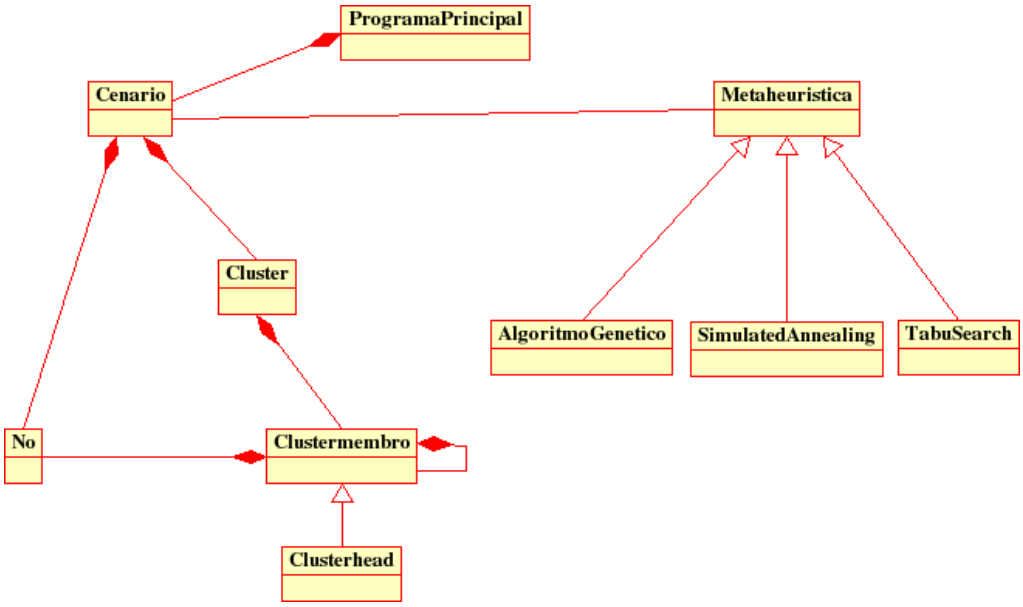


*Figura 45: Diagrama de Classes Básico do Modelo Proposto*

Os modelos de particionamento em *clusters*, propostos no presente estudo, são: *GAdHoc* [GAR05], *SAdHoc* e *TSAdHoc*, baseados, respectivamente, nas Metaheuristicas e classes de mesmo nome: algoritmo genético (classe *AlgoritmoGenetico*), *Simulated Annealing* (classe *SimulatedAnnealing*) e Busca Tabu (classe *TabuSearch*), cada qual como classe filha da super-classe *Metaheuristica.*

A seguir, serão descritos o programa principal, modelos propostos (*GAdHoc* , *SAdHoc* e *TSAdHoc*) e partes do ambiente de simulação.

## 5.3. PROGRAMA PRINCIPAL

O algoritmo básico para a simulação está situado dentro do escopo da classe *ProgramaPrincipal*, constitui os passos gerais para a execução do ambiente de simulação, como pode ser visto no método *executa* da classe *ProgramaPrincipal* na Figura 46.

A simulação é iniciada em $t=0$ com a disposição dos nós no terreno, de dimensão quadrada máxima DIMMAX, segundo parâmetros iniciais de configuração e condições restritivas de conectividade da rede (método *geraEstadoInicial* da classe *Cenario*). Ainda no instante inicial de tempo, temos a determinação do *dominant set*,

*DS(G)*, e $N_{open}(v)$ , $v \in DS(G)$ , para a determinação da topologia da rede (método *determinaTopologia* da classe *Cenario*). Sendo a rede dinâmica, temos a movimentação de nós no terreno, e suas implicações (consumo de energia, efeitos de borda, atualização de vizinhança, transmissão de dados) (método *realizaDinamicaNos* da classe *Cenario*) . A dinâmica de nós da rede inclui também a possibilidade de ocorrência de *handoffs* por perda de potência em relação ao *clusterhead*, *handoffs* por diferença de potência entre *clusterheads* interno e externo segundo um limiar – *threshold*, com agregação de histerese, exaustão de um nó por sobrecarga de transmissão e possibilidade de falha de nós da rede. Devido à esta dinâmica, uma topologia gerada anteriormente não necessariamente será a mais adequada para o próximo instante de tempo. Por isso, é feita uma avaliação para verificar se é compensador determinar uma nova topologia ou não (métodos da classe *Cenário – cenario.analisaDinamicaNos()*), visto que a formação de um novo arranjo de *DS(G)* e $N(v)_{open}$ , $v \in DS(G)$ , também apresenta custo para a rede. Se houver esta necessidade, uma nova topologia é gerada. O algoritmo é finalizado quando a condição de parada (*not termination condition*) é atingida.

```
Classe ProgramaPrincipal::void executa(void)
{       t=0;
        cenario.geraEstadoInicial();
        cenario.determinaTopologia();
        do
        {   cenario.realizaDinamicaNos();
            if (cenario.analisaDinamicaNos())
            {   cenario.determinaTopologia();
            }
            t++;
        }while (not termination condition);
}
```

*Figura 46: Método de execução do algoritmo básico da classe ProgramaPrincipal*

## 5.4. GADHOC (ALGORITMOS GENÉTICOS)

Uma das técnicas de otimização propostas no presente estudo é o *GAdHoc* [GAR05]. Trata-se de um algoritmo baseado em algoritmos genéticos (*genetic algorithms* - GA's) aplicada na realização do particionamento em *clusters*. GA's são metaheurísticas utilizadas para solucionar problemas de busca e otimização de forma adaptativa [HOL75]

É um algoritmo de pesquisa global que utiliza populações para encontrar um mínimo (máximo) local. [DOR95a] e [DOR95b] mostraram que obtêm-se soluções satisfatórias para funções não-lineares em espaços de soluções globais. A utilização de GA em funções não-lineares complexas também foi aplicado com êxito em [DOR95c] e [DOR94]. Uma descrição formal do algoritmo é provida em [REE93] e [GOL93].

### 5.4.1. Estrutura

O modelo de particionamento em *clusters GAdHoc* é formado por duas classes: *Gadhoc* e *Cromossomo*. Elas estão relacionadas por associação. A classe *Gadhoc* incorpora a base do GA. As fases do GA são métodos da classe *Gadhoc*: gerar população inicial (*Gadhoc::geraPopulacaoInicial()*), seleção (*Gadhoc::selecao()*), cruzamento (*Gadhoc::cruzamento()*), mutação (*Gadhoc::mutacao()*), nova população (*Gadhoc::calculaTamNovaPopulacao()*) e a condição de parada (*Gadhoc::condicaoDeParada()*). A população de cromossomos (*cromossomo[POPULACAO]*) e o tamanho da população (*populacao*) são definidos como atributos privados. Na classe cromossomo temos uma solução válida para o particionamento em *clusters* (*g[N][K]*) e a sua correspondente Função de Aptidão (*Fapt*) como atributos privados. A atribuição de uma solução inicial ocorre na inicialização (*Cromossomo::inicializa()*), o cálculo de *FApt* é de responsabilidade do método *Cromossomo::calculaFapt()* . Um método complementar à mutação (*Gadhoc::mutacao()*) é realizado na classe *Cromossomo* (*Cromossomo::fazMutacao()*). A Figura 47 mostra o Diagrama de Classes do modelo *GAdHoc*.

O modelo proposto segue o algoritmo básico do GA da Figura 27.

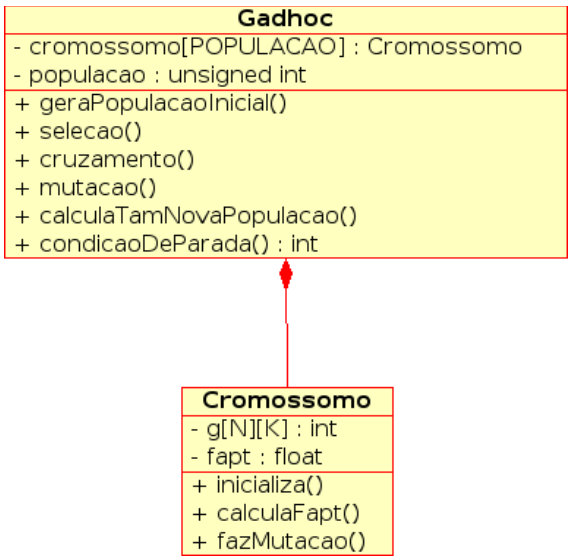


*Figura 47: Diagrama de Classes do GAdHoc*

### 5.4.2. Parâmetros do GAdHoc

Alguns parâmetros influem no comportamento do algoritmo, conforme as necessidades do problema e dos recursos disponíveis. A citar:

- Cromossomo
- População - $P$
- Função de Aptidão (*fitness*) - $FApt$
- Seleção
- Cruzamento (*crossover*)
- Mutação
- Taxa de Redução Populacional

### 5.4.3. Cromossomo

Trata-se de uma estrutura que apresenta uma solução válida dentro do espaço de soluções de um determinado problema. São integrantes da População do *GAdHoc*. São representados tipicamente em forma de cadeias binárias. Cada posição desta cadeia tem dois alelos possíveis: 0 e 1. Cada cromossomo apresenta uma Função de Aptidão que mensura a sua capacidade de resolver o problema dado.

No *GAdHoc,* temos o cromossomo $G_{N,K}$ [GAR05] definido por:

$$G_{N,K}=\begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & \cdots & g_{0,K-1} \\ g_{1,0} & g_{1,1} & \cdots & \cdots & g_{1,K-1} \\ \vdots & \cdots & g_{i,k} & \cdots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ g_{N-1,0} & g_{N-1,1} & \cdots & \cdots & g_{N-1,K-1} \end{bmatrix}$$

*Equação 4: Representação de um Cromossomo para G(N,K)*

, onde cada linha i representa um nó da topologia e cada coluna representa um *cluster* de G. Cada elemento $g_{ik}$ , $i,k \in \mathbb{N}, 0 \leqslant i < N, 0 \leqslant k < K$ , tem como propriedade:

- $\forall i, g_{i,0}, g_{i,1}, \ldots, g_{i,k-1}, g_{i,k}, g_{i,k+1}, \ldots, g_{N-1,K-1}$ ,sendo $g_{i,k}=1$ e $g_{i,0}$ , $g_{i,1}$ , $\ldots$ , $g_{i,k-1}$, $g_{i,k+1}, \cdots$ , $g_{N-1,K-1}$ .

### 5.4.4. População $P$

É o número de cromossomos usados na busca de uma solução quase-ótima. Cada cromossomo representa uma solução válida do problema a ser resolvido. $P$ influencia diretamente o desempenho global e a eficiência do *GAdHoc*. Se $P$ for pequeno, o desempenho pode cair e aumentar as chances de se encontrar mínimos locais indesejáveis, pois o domínio do espaço de busca é reduzido. Dentro dos limites dos recursos computacionais utilizados, portanto, é desejável que se utilize valores grandes para $P$ . Ter-se-á uma cobertura significativa do espaço de busca e previne convergências prematuras para soluções locais indesejadas.

### 5.4.5. Função de Aptidão (*fitness*) $F_{Apt}$

É uma função que calcula o valor de aptidão de cada indivíduo *i* de $P$ , $F_{Apt_i}$ ou $F_i$ . $F_{Apt}$ é o componente mais importante de qualquer algoritmo genético, pois é por meio desta função que se mede quão próximo um indivíduo está da solução desejada ou quão boa é esta solução. A estratégia do GA é minimizar $F_{Apt}$ .

Uma boa representação de $F_{Apt}$ garante uma convergência rápida e eficiente para uma solução quase-ótima. Se houver pouca precisão na avaliação, o algoritmo pode: perder desempenho, finalizar em mínimos locais indesejados, e até, em casos extremos,

deixar de lado uma solução quase-ótima.

No GAdHoc, temos $F_{Apt}$ ou *fitness* [GAR05] definida por:

$$F_{Apt} = \left\{ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{K-1} f_{ij} \cdot g_{ik} \cdot (1 - g_{kj}) \right\}$$

*Equação 5: Função de Aptidão ou fitness - GAdHoc*

, onde N é o número de nós da rede, K é o número máximo de clusters admitido, sendo $K \leqslant N$ , $i, j \in N, k \in K$ , $g_{ik}$ é variável de decisão, $f_{ij}$ representa o fluxo de comunicação passante entre os nós i,j da matriz de fluxos $F_{i;}$ $f_{ij}$ apresenta as seguintes propriedades:

- Se $i = j \rightarrow f_{ij} = 0$ ;
- $f_{ij} = f_{ji}$ ;
- Se i,j são alcançáveis, diretamente ou não, e houver fluxo de dados entre eles, então $f_{ij} > 0$ , caso contrário $f_{ij} = 0$ .

### 5.4.6. Seleção

É a função que seleciona os indivíduos para a reprodução. Baseia-se na Função de Aptidão $F_{Apt}$ de cada cromossomo. Indivíduos mais aptos têm maior probabilidade de serem escolhidos para a reprodução (*crossover*), permitindo que esses indivíduos passem as suas características para as próximas gerações, ou seja, para a próxima população $P'$ . Este processo funciona de forma análoga ao processo de seleção natural de Darwin, onde indivíduos altamente adaptados ao seu ambiente possuem naturalmente mais oportunidades para reproduzir do que aqueles considerados mais fracos.

Assim, se $Fi_{Apt}$ (ou simplesmente $F_i$ ) é a aptidão de um indivíduo i dentro de $P$ , a probabilidade de i ser selecionado $p(i)$ é $p(i) = \frac{F_i}{\sum_{i=1}^{N} F_i}$ .

Existem vários métodos para selecionar os indivíduos sobre os quais serão aplicados os operadores genéticos, entre eles:

- Método de seleção por Roleta (*roulette*); e

- Método de seleção por Torneio.

No método de seleção por Roleta, $select_{roulette}$ , cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, para indivíduos com alta aptidão $\uparrow F_i$ é dada uma porção maior da roleta, enquanto aos indivíduos de aptidão mais baixa, é dada uma porção relativamente menor $\downarrow F_i$ . Cada indivíduo é representado por uma fatia proporcional à sua aptidão relativa. Para visualizar este método, é comum usar um círculo dividido em regiões onde a área de cada região é proporcional à $F_i$ de *i*. Coloca-se sobre este círculo uma "roleta". Após um giro da roleta a posição dos cursores indicará o indivíduo selecionado. O número de rodadas $n_{roulette}$ deve determinar a nova população $P'$ . A seleção pode conter várias cópias de um mesmo cromossomo. Desta forma, os cromossomos que detiverem maior área terão maior probabilidade de serem selecionados várias vezes. Outros, por sua vez, podem simplesmente desaparecer.

```
S = sum of P(i);
Repeat N
    r = random[0..S];
    if(P(i) < r)
        select chromossome;
End repeat
```

*Figura 48: Método da Roleta*

No método de Seleção por Torneio, $select_{torneio}$ , cada indivíduo da população é escolhido aleatoriamente para formar uma sub-população temporária. Deste grupo, o indivíduo selecionado dependerá de uma probabilidade k definida previamente.

Um problema encontrado no Método da Roleta é o tempo de processamento, já que o método exige duas passagens por todos os indivíduos da população. Por outro lado, o método do Torneio apresenta melhor desempenho computacional, visto que não exige a comparação entre todos os indivíduos da população [BAN98].

Experimentalmente, verificou-se uma vantagem ligeiramente superior do método da Roleta sobre o Torneio sobre a convergência de $F_{apt}$ do Algoritmo Genético.

No *GAdHoc,* adotou-se o método da roleta.

```
k=0,90
Repeat N
    Choose 2 chromossomes
     r = random(0 and 1);
     if(r < k)
            select the best one;
      else
            select the worst one;
End repeat
```

*Figura 49: Método do Torneio*

### 5.4.7. Cruzamento (*crossover*)

No Cruzamento são criados novos indivíduos misturando características de dois indivíduos "pais". Esta mistura é feita tentando imitar (em um alto nível de abstração) a reprodução de genes em células. Trechos das características de um indivíduo são trocados pelo trecho equivalente do outro. O resultado desta operação é um indivíduo que potencialmente combine as melhores características dos indivíduos usados como base. Uma forma de cruzamento utiliza o ponto de corte $(c)$ .

O ponto de corte do cromossomo é escolhido de forma aleatória sobre o comprimento da cadeia de bits que o representa. A partir deste ponto é realizada a troca de material genético entre os dois indivíduos.

No *GAdHoc,* adotou-se pontos de corte aleatórios para cada par, distribuído uniformemente, com $c=[0..N-1]$ , onde c é aplicado ao $k*c-ésimo$ bit, conforme é descrito na Tabela 1. Considere K=3, c =4 e os grupos de bits entre [0..N-1].

| ***Par(casal)*** | ***i*** | ***Cromossomo(bits)*** |
|---|---|---|
| 0 | 0 | **001001001010010** --- **100010010010** |
| | 1 | 010010100001001 --- 010100100100 |
| ***Filhos do casal*** | ***i'*** | ***Cromossomo*(bits)** |
| 0 | 0 | **001001001010010** --- 010100100100 |
| | 1 | 010010100001001 --- **100010010010** |

*Tabela 1: Cruzamento de um par de indivíduos (ponto de corte: c=4)*

### 5.4.8. Mutação

A operação de Mutação é efetuada alterando-se o valor de um gene de um indivíduo escolhido aleatoriamente com uma determinada probabilidade de ocorrência em $P$, chamada de taxa de mutação $(\theta)$. É utilizada para garantir maior varredura do domínio de soluções e evitar que o algoritmo genético permaneça em mínimos indesejados. É responsável pela introdução da diversidade genética na população.

O valor de $\theta$ precisa ser grande o suficiente para causar diversidade genética. Por outro lado, se for demasiadamente grande, acaba dificultando a convergência para uma solução quase-ótima. Taxas comumente usadas estão entre $\theta$ =[0,01..1], sendo $\theta$ uma porcentagem [HOL75].

No *GAdHoc*, devido às propriedades de $G_{ik}$ da Equação 4, a mutação ocorre em um conjuntos de k-bits, sendo $k$ o número de *clusters*, conforme é apresentado na Tabela 2. No *GAdHoc,* adotou-se $\theta = 0{,}5\%$.

| ***Cromossomo(bits)*** |
|---|
| 100100100010---**010**---001100100100 |
| 100100100010---**001**---001100100100 |

*Tabela 2: Mutação no quinto conjunto de k-bits (k=3) de um cromossomo*

### 5.4.9. Taxa de Redução Populacional

Determina a redução populacional a cada ciclo do GA. Se a taxa de redução for alta, a convergência será rápida. Há o risco de se atingir mínimos locais indesejados, mas com tempo de execução menor. Por outro lado, se a taxa de redução for pequena, a convergência da solução é lenta. Isto aumenta a probabilidade de se atingir um ótimo local, mas com um tempo de execução maior, podendo em alguns casos causar exaustão de recursos computacionais e a conseqüente saída anormal da execução do programa. No modelo proposto, optou-se por uma convergência lenta.

### 5.4.10. Funcionamento do GAdHoc

Descrever-se-á o funcionamento do *GAdHoc* por meio de um exemplo. Seja uma rede constituída pelo grafo $G=(V,E)$ ,com |V|=9, formado por nós de um conjunto *V*, onde cada nó possui um número de identificação distinto (ID), e de um conjunto de arestas *E*, formada pelas ligações entre nós quaisquer, entre nós alcançáveis entre si (com distância menor que um raio de transmissão máximo RAIOMAX) e com fluxo de comunicação não-nulo ,F , representado pelas matrizes de fluxo F e distância D. Para F, temos que $f_{ij}$ representa o fluxo de comunicação passante, sendo $f_{ij}=0$ , se a aresta $\overline{ij}>RAIOMAX$ ou $\overline{ij}\leq RAIOMAX$ desde que $f_{ij}$ seja zero, valores uniformemente distribuídos para $f_{ij}\neq 0$ e, por fim, $f_{ij}=f_{ji}$ . Para D, temos que $d_{ij}=0$ se $i=j$ e $d_{ij}=\infty$ se $\overline{ij}>RAIOMAX$ .

$$F=\begin{bmatrix} 0 & f_{01} & f_{02} & 0 & 0 & 0 & 0 & 0 & 0 \\ f_{10} & 0 & f_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ f_{20} & f_{21} & 0 & f_{23} & 0 & 0 & f_{26} & 0 & 0 \\ 0 & 0 & f_{32} & 0 & f_{34} & f_{35} & f_{36} & 0 & 0 \\ 0 & 0 & 0 & f_{43} & 0 & f_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & f_{53} & f_{54} & 0 & 0 & 0 & 0 \\ 0 & 0 & f_{62} & f_{63} & 0 & 0 & 0 & f_{67} & f_{68} \\ 0 & 0 & 0 & 0 & 0 & 0 & f_{76} & 0 & f_{78} \\ 0 & 0 & 0 & 0 & 0 & 0 & f_{86} & f_{87} & 0 \end{bmatrix}$$

*Equação 6: Matriz de Fluxos(F)para uma topologia G(V,K)=(9,3), com |V|=9 e K=3.*

$$D=\begin{bmatrix} 0 & 8.246 & 10.296 & \infty & \infty & \infty & \infty & \infty & \infty \\ 8.246 & 0 & 7.616 & \infty & \infty & \infty & \infty & \infty & \infty \\ 10.296 & 7.616 & 0 & 16.2789 & \infty & \infty & \infty & \infty & 17.0300 \\ \infty & \infty & 16.279 & 0 & 9.434 & 10.198 & \infty & \infty & 16.763 \\ \infty & \infty & \infty & 9.434 & 0 & 7.810 & \infty & \infty & \infty \\ \infty & \infty & \infty & 10.198 & 7.810 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 & 8.544 & 7.280 \\ \infty & \infty & \infty & \infty & \infty & \infty & 8.544 & 0 & 10.770 \\ \infty & \infty & 17.0300 & 16.763 & \infty & \infty & 7.280 & 10.770 & 0 \end{bmatrix}$$

*Equação 7: Matriz de Distâncias (D) para uma topologia G(V,K)=(9,3)*

Deseja-se obter um valor mínimo para $F_{Apt}$ para o *GAdHoc*, definido na Equação 5, para um particionamento em *k-clusters*. Neste exemplo, com K=3.

Inicia-se o algoritmo com a Geração da População Inicial, suponha a população inicial $P$ =6, determinada aleatoriamente:

| ***i*** | ***Cromossomo(bits)*** | ***F_i*** |
|---|---|---|
| 0 | 10010010001001001001001001 | 100,000 |
| 1 | 10001000100100100100100100 1 | 790,000 |
| 2 | 00100100101010010000101001 0 | 1076,000 |
| 3 | 10010010001001000110010010 0 | 552,000 |
| 4 | 00100100101001010001001001 0 | 552,000 |
| 5 | 01001010000100101010010010 0 | 1058,000 |

*Tabela 3: Exemplo: População de indivíduos com P=6*

A próxima fase é a Seleção. Baseou-se no método da roleta. Cada indivíduo da população é representado na roleta proporcionalmente à sua $F_i$ . Como deseja-se um valor mínimo, optou-se pelo parâmetro inversamente proporcional $1/F_i$ . Assim, para indivíduos com alta aptidão (menor valor) é dada uma porção maior da roleta, enquanto aos indivíduos de aptidão mais baixa, é dada uma porção relativamente menor. Assim, usando a formulação

$$p(i)=\frac{1/F_i}{\sum_{i=0}^{N=5} 1/F_i}=\frac{1/F_i}{0{,}01676356}$$

, temos:

| **i** | **F_i** | **1/F_i** | **p(i)** | **p(i)** |
|---|---|---|---|---|
| 0 | 100,000 | 0,01000000 | 0,5965 | 59,65% |
| 1 | 790,000 | 0,00126582 | 0,0755 | 7,555 |
| 2 | 1076,000 | 0,00092937 | 0,0554 | 5,54% |
| 3 | 552,000 | 0,00181159 | 0,1081 | 10,81% |
| 4 | 552,000 | 0,00181159 | 0,1081 | 10,81% |
| 5 | 1058,000 | 0,00094518 | 0,0564 | 5,64% |
| soma | | 0,01676356 | 1,00 | 100% |

*Tabela 4: GAdHoc: Método da Roleta: cálculo de p(i)(%)*

Neste contexto, foram selecionados os pares $Select_{roulette} = \{0,3\},\{2,3\},\{4,5\}$ . É importante salientar que no método da roleta, um mesmo cromossomo pode ser selecionado mais de uma vez. Temos os seguintes pares de cromossomos:

| *Par(casal)* | *i* | *Cromossomo(bits)* | *F_i* |
|---|---|---|---|
| 0 | 3 | 10010010001001000110010010 0 | 552,000 |
| | 2 | 00100100101010010000101001 0 | 1076,000 |
| 1 | 4 | 00100100101001010001001001 0 | 552,000 |
| | 5 | 01001010000100101010010010 0 | 1058,000 |
| 2 | 0 | 10010010001001001000100 1001 | 100,000 |
| | 3 | 10010010001001000110010010 0 | 552,000 |

*Tabela 5: GAdHoc: Método da Roleta: seleção de pares de cromossomos*

Na fase do Cruzamento (*crossover*), cada par recebe um ponto de corte aleatório, distribuído uniformemente $c = [0..N-1]$ . Desta forma, temos:

| *par* | $i_1$ *x* $i_2$ | *c* |
|---|---|---|
| 0 | 3 x 2 | 4 |
| 1 | 4 x 5 | 4 |
| 2 | 0 x 3 | 2 |

*Tabela 6: GAdHoc: determinação do ponto de corte para cada par (casal)*

| *Filhos do casal* | *i'* | *Cromossomo(bits)* | *F'_i* |
|---|---|---|---|
| 0 | 0 | 10010010001001010000101001 0 | 1252,000 |
| | 1 | 00100100101010000110010010 0 | 774,000 |
| 1 | 2 | 00100100101001001010010010 0 | 100,000 |
| | 3 | 01001010000100110001001001 0 | 1090,000 |
| 2 | 4 | 10010010001001000110010010 0 | 552,000 |
| | 5 | 10010010001001001000100 1001 | 100,000 |

*Tabela 7: GAdHoc: filhos dos respectivos casais após o crossover*

Na mutação, temos $\theta$ =0,5%, i.e, $\theta = 0{,}00030$ . Como se trata de um exemplo bem simples, temos $P = 6$ . A taxa de mutação não se torna significativa, visto que não é suficiente grande para provocar a mutação de pelo menos um indivíduo. Mas suponha que um indivíduo aleatório apresente mutação. Suponha que o quinto conjunto de k-bits (k=3) do cromossomo 4 sofra mutação.

| *i* | *Cromossomo(bits)* |
|---|---|
| 4 | 100100100010---**010**---001100100100 |
| 4' | 100100100010---**001**---001100100100 |

*Tabela 8: GAdHoc: cromossomo 4 sofrendo mutação*

## 5.5. SADHOC (SIMULATED ANNEALING)

Outra técnica de particionamento em *clusters* proposta no presente estudo é o *SAdHoc*.Trata-se de um algoritmo baseado na técnica de metaheurística *Simulated Annealing* (SA) [KIR83]. SA é um método de busca local probabilístico, usada para percorrer um espaço de soluções contendo vários mínimos locais. Trata-se de uma variante do Método de Metrópolis [MET53], que simula o comportamento de uma coleção de átomos em equilíbrio numa dada temperatura por meio da geração de variáveis aleatórias distribuídas de acordo com uma dada distribuição multivariada de probabilidade. Admite soluções de piora do tipo $T \to 0 \Rightarrow e^{-\Delta / k_b T} \to 0$ para escapar de ótimos locais, onde $k_b$ é a constante de *Boltzman*, $k_b$=*1,3807e-23 J.K*$^{-1}$. Foi derivado de simulações em termodinâmica e por esta razão o parâmetro *T* é referenciado como temperatura e a maneira pela qual ela é reduzida é chamada de processo de resfriamento.

### 5.5.1. Estrutura

O modelo de particionamento em *clusters SAdHoc* é formado por duas classes: *Sadhoc* e *Solucao*. Elas estão relacionadas por associação. A classe *Sadhoc* incorpora a base do SA. As fases do SA são métodos da classe *Sadhoc*: gera solução inicial (*Sadhoc::geraSolucaoInicial()*), gera temperatura inicial (*Sadhoc::geraTemperaturaInicial()*), seleciona vizinho da solução atual $s_0$ (*Sadhoc::selecionaVizinhoS()*). É proposto uma variante para a seleção de s, $s \in N(s_0)_{open}$ . O modelo clássico [KIR83], apresenta uma seleção aleatória de s vizinho de $s_0$. Experimentalmente, verificou-se melhor convergência para resultados quase-ótimos e válidos para o problema do particionamento em *clusters* se selecionarmos um vizinho aleatório a partir de uma Extremidade de Partida ($OE(k_i)$) de um *cluster* ($k_i$) selecionado aleatoriamente. Esta solução está implementada por método protegido, que é chamado pelo método anterior (*Sadhoc::selecionaVizinhoS()*), e é chamado de

*Sadhoc::geraUmVizinhoAleatorioAPartirDeExtremidadeDePartida()*. O método *Sadhoc::aplicaProbabilidadeDeAceitacaoDeMovimentoDePiora()*, é chamado quando $\Delta(f)>0$ , e avalia se haverá a aceitação de uma solução pior que a solução atual. O *SAdHoc* emprega a condição $r<e^{-\Delta(f)/T}$ , onde r é gerado aleatoriamente em distribuição uniforme, $r=[0..1]$ . Embora a condição original [KIR83] apresente a constante de Boltzman ($k_b$), uma variante da formulação sem a mesma é sugerida em [REE93].

O método *Sadhoc::aplicaRazaoDeResfriamento()*, realiza a redução de temperatura do sistema em função de uma Razão de Resfriamento - $\alpha$ . O método *Sadhoc::condicaoDeParada()* determina a finalização do algoritmo, que ocorrerá quando houver congelamento do sistema (*Sadhoc::verificaCongelamento()*) ou quando houver cristalização do sistema, isto é, sem melhora do sistema após um determinado número de resfriamentos sucessivos (*Sadhoc::verificaCristalizacao()*). O número de resfriamentos é obtido por um Coeficiente de Cristalização - $\delta$ . A melhor solução obtida, isto é, o estado com energia mínima, estará em $s_0$, na finalização do algoritmo. A Figura 50 mostra os Diagramas de Classes dos modelos *Sadhoc e solucao.*

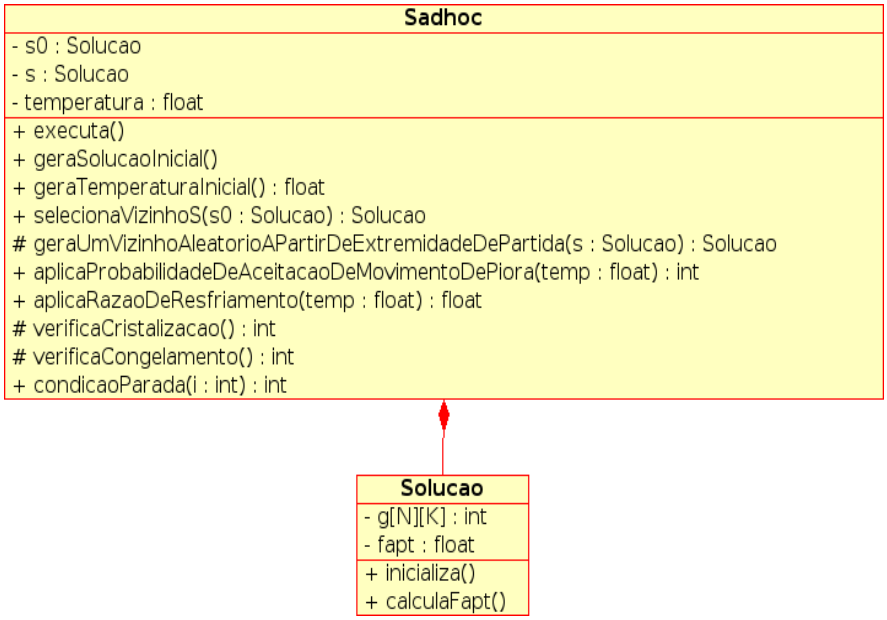


*Figura 50: Diagrama de Classes do SAdHoc*

O modelo proposto segue o algoritmo básico do SA da Figura 25.

### 5.5.2. Parâmetros do SAdHoc

Alguns parâmetros influem no comportamento do algoritmo, conforme as

necessidades do problema e dos recursos disponíveis e foram obtidos por experimentação. A citar:

- Temperatura Inicial – $T_{0,}$
- Razão de Resfriamento - $\alpha$ ,
- Coeficiente de Cristalização - $\delta$ ,
- Temperatura de Congelamento – $T_{c,}$
- Probabilidade de Aceitação de um Movimento de Piora-($p_{aceitação}$),
- Número Máximo de Iterações (*SAMAX*).

### 5.5.3. Temperatura Inicial - $T_0$

A Temperatura Inicial ($T_0$) representa o estado inicial do sistema. Seu valor é obtido experimentalmente. Não deve ser tão baixa para que o sistema não fique aquecido. Por outro lado, se for demasiadamente alta, o sistema permanecerá em estados de alta energia, aceitará um grande número de soluções vizinhas, dificultando a convergência de $F_o$ durante as temperaturas mais elevadas.

Como o valor ideal para $T_0$ depende de cada problema, [SOU06] sugere uma etapa preliminar para determinar experimentalmente a temperatura inicial do sistema.

A idéia é determinar a temperatura inicial do sistema ($T_0$) em função do percentual de vizinhos aceitos, *va(%)*, (*T(va(%))*). A partir de uma temperatura inicial baixa ( $T_{baixa}$ ) o sistema sofrerá aquecimento gradativo, em razão de uma taxa de aquecimento ( $\lambda$ ) até atingir um estado de aquecimento, tal que a Probabilidade de Aceitação de Soluções Vizinhas seja de $p_{aceitacao}$ .

No *SAdHoc*, os valores experimentais são: $T_{baixa}$ =100, $\lambda$ =0.10 [SOU06], $p_{aceitacao}$=95% [SOU06].

### 5.5.4. Razão de Resfriamento $\alpha$

A Razão de Resfriamento- $\alpha$ , $\alpha$=[0..1] realiza a diminuição da temperatura do sistema em cada iteração. Seu valor é obtido experimentalmente. Se

$\alpha \ll 1$ , a convergência é rápida para *T*. Conduz a produtos meta-estáveis de maior energia interna. Apresenta maior possibilidade de atingir estados indesejáveis de cristalização por resfriamento brusco do sistema, isto é, sem movimentos de melhora após um determinado número de iterações. Por outro lado, se $\alpha \approx 1$ , temos uma convergência lenta para *T*. O resfriamento lento conduz a produtos mais estáveis, estruturalmente fortes e de menor energia, aqui representado por $F_0$. No modelo proposto, optou-se com um resfriamento lento.

### 5.5.5. Coeficiente de Cristalização $\delta$

A Cristalização ocorre quando o sistema atinge um arranjo estável de suas moléculas (componentes) durante o processo de resfriamento. O Coeficiente de Cristalização determina o número de estados necessários para se tipificar um estado estável cristalizado, isto é, sem movimentos de melhora após um determinado número de iterações. É obtido experimentalmente. Seu valor não deve ser muito pequeno para não tipificar estados estáveis por curto intervalo de iterações. Nem sempre um estado de cristalização é desejável, principalmente quando ocorrer em alguma etapa intermediária da execução do algoritmo. Estados indesejáveis de cristalização podem ocorrer quando há resfriamento brusco do sistema. No modelo proposto, optou-se por $\delta - 20$ .

### 5.5.6. Temperatura de Congelamento - $T_c$

A Temperatura de Congelamento determina o estado de entropia mínima do sistema, após resfriamentos sucessivos. Depende do problema proposto e da sua complexidade, em particular do número de nós do sistema e sua disposição no terreno. Outra dependência é com relação a temperatura inicial do sistema ($T_0$) que é variável mesmo em diferentes execuções de um mesmo cenário. Se $T_c$ for alto, há o risco de se atingir o congelamento precipitado do sistema, podendo atingir estados intermediários ou mínimos locais indesejados. Como o congelamento do sistema é apenas uma das condições de parada do SA e varia conforme o problema. No SAdHoc, optou-se então por *Tc* extremamente baixo..

### 5.5.7. Probabilidade de Aceitação de um Movimento de Piora-(P(aceitação))

*P(aceitação)* ocorre caso $\Delta(f)>0$ . Verifica se haverá a aceitação de uma solução pior que a solução atual. Emprega a condição $r<e^{-\Delta(f)/T}$ , onde $r$ é gerado aleatoriamente em distribuição uniforme, $r=[0..1]$ . A probabilidade de aceitação é maior em temperaturas mais elevadas. Por outro lado, a aceitação diminui em temperaturas mais baixas. No *SAdHoc*, optou-se pela variante $r<e^{-\Delta(f)/T}$ [REE93], sem a constante de *Boltzman* ($k_b$), para escapar de mínimos locais, ao contrário da formulação original [KIR83], apenas para simplificar a formulação do problema.

### 5.5.8. Número Máximo de Iterações (SAMAX)

O parâmetro *SAMAX* simplesmente determina o estado de equilíbrio térmico do sistema [MET53]. É uma das condições de parada do sistema. Outra condição é a obtenção de estado de cristalização.

### 5.5.9. Funcionamento do SAdHoc

Descrever-se-á o funcionamento do *SAdHoc* por meio de um exemplo. Seja a rede descrita na seção anterior. Deseja-se obter um valor mínimo para $F_o$ para o *SAdHoc* para um particionamento em *k-clusters*. Neste exemplo, com K=3. Inicia-se o algoritmo com a determinação da temperatura inicial do sistema ($T_0$) em função do percentual de vizinhos aceitos (*va*), (*t(va)*). A partir de uma temperatura inicial baixa $(T_{baixa})$ o sistema sofrerá aquecimento gradativo até atingir um estado de aquecimento, tal que a Probabilidade de Aceitação de Soluções vizinhas seja de $p_{aceitação}$ .Considere os valores experimentais $T_{baixa}=100, p_{aceitação}=95\%$ , conforme Figura 51.

Obtemos $T_0=t(p_{aceitacao}\geqslant 0.95)=895{,}430$ .

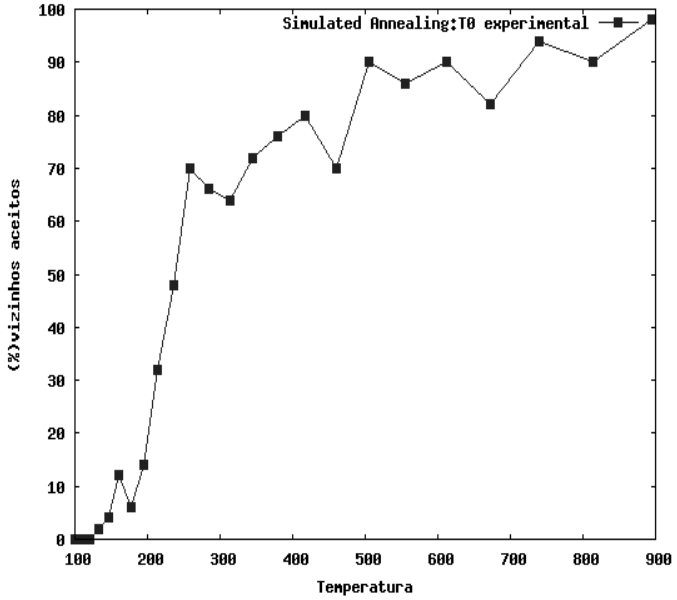


*Figura 51: SAdHoc: gerando temperatura inicial do sistema*

A próxima fase é a geração de uma solução inicial, determinada aleatoriamente, no instante *T=0*, conforme Tabela 9:

| ***T*** | $s_0$ | $F_{so}$ |
|---|---|---|
| 0 | 10010010010001001000100110 0 | 878,000 |

*Tabela 9: SAdHoc: gerando solução inicial aleatória, em T=0*

O algoritmo prossegue gerando, em cada laço, uma solução *s* vizinha de $s_0$, $s \in N_{open}(s_0)$ . Para simplificar este exemplo, suponha que *SAMAX*=4. Para *T=1*:

| ***T*** | $s_0$ | $F_{so}$ | ***s*** | $F_s$ |
|---|---|---|---|---|
| 1 | 10010010010001001000100110 0 | 878,000 | 10010010010001001000100100**1** | 404,000 |
| -->novo otimo local s encontrado $\Delta(f) = f_s - f_{s_0} < 0 \quad \Rightarrow s_0 \leftarrow s$ | | | | |

*Tabela 10: SAdHoc: execução do algoritmo, em T=1*

Temos a seleção de um vizinho *s* de $s_0$, obtido a partir do movimento do r-ésimo conjunto de k-bits da seqüência, sendo *k* o número de *clusters*, e *r* é o r-ésimo cluster, obtido de forma aleatória. Na seqüência de bits de *s*, temos em destaque o conjunto de k-bits movimentado em relação a $s_0$.

A solução *s* é melhor que a solução atual $s_0$, $s \in N_{open}(s_0)$ . Logo $s_0 \leftarrow s$ . O algoritmo prossegue. Para *T=2*:

| **T** | $s_0$ | $F_{so}$ | **s** | $F_s$ |
|---|---|---|---|---|
| 2 | 10010010010001001000100 1001 | 404,000 | 10010010001001001000100 1001 | 100,000 |
| -->novo otimo local s encontrado $\Delta(f)=f_s-f_{s_0}<0 \quad \Rightarrow s_0 \leftarrow s$ | | | | |

*Tabela 11: SAdHoc: execução do algoritmo, em T=2*

Tal como no instante anterior, temos que *s* é melhor que a solução atual $s_0$, $s \in N_{open}(s_0)$ . Logo $s_0 \leftarrow s$ . O algoritmo prossegue. Para *T=3*:

| **T** | $s_0$ | $F_{so}$ | **s** | $F_s$ |
|---|---|---|---|---|
| 3 | 10010010001001001000100 1001 | 100,000 | 10010010001001001000100 1001 | 404,000 |
| -->executando annealing (escapa de otimo local s0) $\Delta(f)=f_s-f_{s_0}>0 \quad \Rightarrow s_0 \leftarrow s$ | | | | |

*Tabela 12: SAdHoc: execução do algoritmo, em T=3*

Um movimento de piora é aceito neste instante. Como $\Delta(f)=f_s-f_{s_0}>0$ , temos a aplicação da probabilidade de aceitação de um movimento de piora dado por $r<e^{-\Delta(f)/T}$ . Como o resultado foi verdadeiro, temos a aceitação de um movimento de piora ($F_s$=404,000), característica do SA. O algoritmo prossegue. Para *T=4*:

| **T** | $s_0$ | $F_{so}$ | **s** | $F_s$ |
|---|---|---|---|---|
| 4 | 10010010010001001000100 1001 | 404,000 | 10010010001001001000100 1001 | 100,000 |
| -->novo otimo local s encontrado $\Delta(f)=f_s-f_{s_0}<0 \quad \Rightarrow s_0 \leftarrow s$ | | | | |

*Tabela 13: SAdHoc: execução do algoritmo, em T=4*

Temos que s é melhor que a solução atual $s_0$. Logo $s_0 \leftarrow s$ . Como *SAMAX*=4, o algoritmo finaliza com $F_{s0}$=100,000.

## 5.6. TSADHOC (BUSCA TABU)

Outra técnica de particionamento em *clusters* proposta no presente estudo é o *TSAdHoc*. Trata-se de um algoritmo baseado na técnica de metaheurística Busca Tabu (BT) [GLO89], [GLO90], [GLO97]. A BT apresenta memória e utiliza uma lista de movimentos proibidos para guiar a exploração do espaço de soluções, mesmo na ausência de movimentos de melhora, evitando que haja a formação de ciclos, isto é, o retorno a um ótimo local previamente visitado. Procura-se, desta forma, imitar o processo de memória dos

seres humanos, a fim de escapar de caminhos cíclicos indesejados.

### 5.6.1. Estrutura

O modelo de particionamento em *clusters TSAdHoc* é formado por cinco classes: *Tsadhoc, Solucao*., *FuncaoAspiracao*, *ListaTabu* e *ItemTabu*. Elas estão relacionadas por associação. A classe *Tsadhoc* incorpora a base do TS. Há quatro atributos privados nesta classe: o objeto *sm* da classe *Solucao* representa a melhor solução encontrada até o momento. Pode ser representada também por *s**. O objeto $s_o$ é a solução na presente iteração. O objeto *s* é uma solução, tal que $s \in N_{open}(s_0)$ . Por fim, o atributo *funcaoaspiracao* representa $A(f(s_m))$ .

As fases do TS são métodos das classes citadas acima: gera solução inicial (*Tsadhoc::geraSolucaoInicial()*), inicializa $A(f(s_m))$ (*FuncaoAspiracao::inicializa()*), inicializa Lista Tabu (LT) (*ListaTabu::inicializa()*), seleciona vizinho tabu da solução atual $s_0$ (*Tsadhoc::selecionaVizinhoTabuS()*). Um vizinho *s*, $s \in N_{open}(s_0)$ é selecionado se $\forall s'' \in N(s_0)_{open} \Rightarrow f(s) \leqslant f(s'')$ e desde que o movimento *m*, $s \leftarrow s_0 \circ m$ , atenda:

- $s \leftarrow s_0 \circ m \Rightarrow s \notin \mathrm{LT}$ ou
- $s \leftarrow s_0 \circ m, s \in \mathrm{LT} \Rightarrow f(s) \leqslant A(f(s_m))$

A relação de pertinência é verificada pelo *Status Tabu* (*ST*), implementado no método (*ListaTabu::statusTabu()*). É proposta uma variante para a seleção de *s*, $s \in N_{open}(s_0)$ . O algoritmo TS apresenta uma seleção aleatória de *s* vizinho de $s_0$ [GLO89]. Experimentalmente, verificou-se melhor convergência para resultados quase-ótimos e válidos para o problema do particionamento em *clusters* se selecionarmos um vizinho aleatório a partir de uma Extremidade de Partida ($OE(k_i)$) de um *cluster* ($k_i$) selecionado aleatoriamente. Esta variante está implementada em um outro método, que é chamado pelo método *Tsadhoc::selecionaVizinhoTabuS()*, e é chamado de *Tsadhoc::geraUmVizinhoAleatorioAPartirDeExtremidadeDePartida()*. Após a seleção de *s*, temos que o movimento *m* passa a se tornar um *Movimento Tabu* (*MT*) e é inserido em *LT* (*ListaTabu::atualiza()*). Se a solução *s* for menor que a menor solução encontrada ($s_m$) (*Tsadhoc::comparaSolucaoSComMelhorSolucaoEncontradaSm()*), então $s_m$ recebe *s* e

$A(f(s_m))$ é atualizada (*FuncaoAspiracao::atualiza()*). O algoritmo finaliza quando a condição de parada for atingida (*Tsadhoc::condicaoDeParada()*), alcançado se o número de iterações atingir o número máximo de iterações *TSMAX*. A Figura 52 mostra a estrutura do diagrama de classes do *TSAdHoc*.

O modelo proposto segue o algoritmo básico do TS da Figura 35.

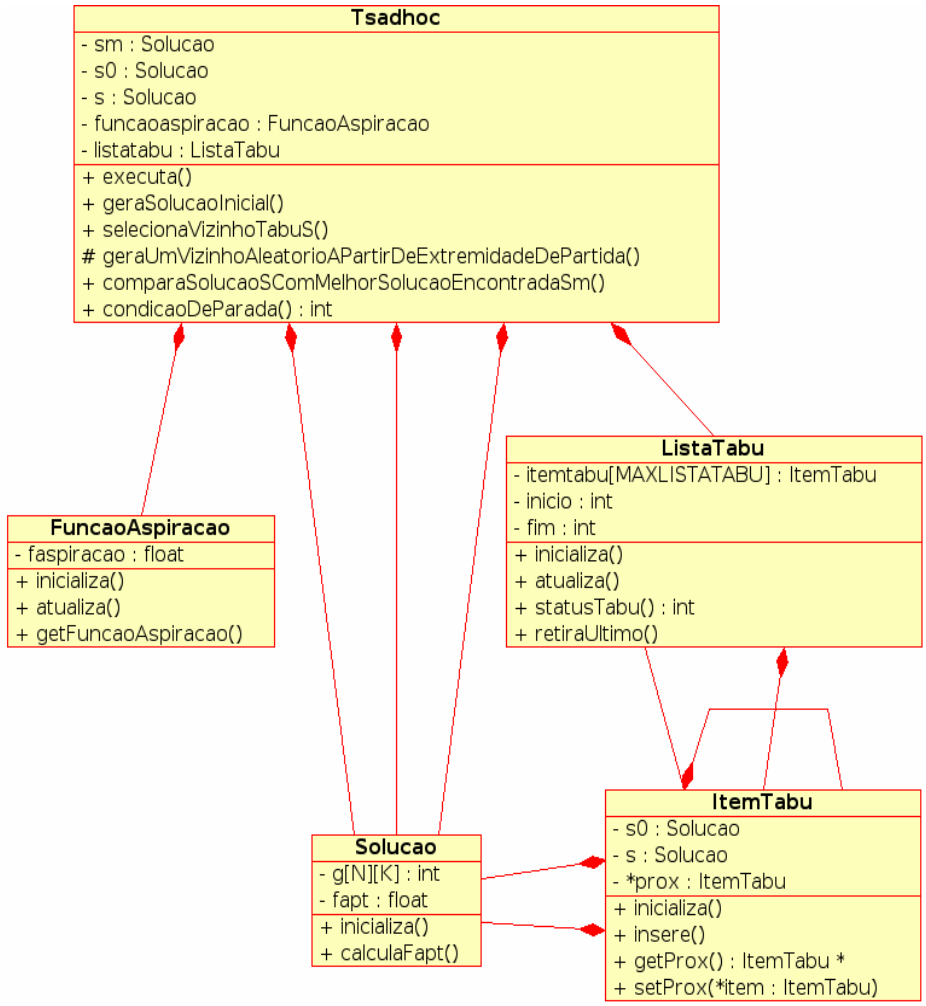


*Figura 52: Diagrama de Classes do TSAdHoc*

### 5.6.2. Parâmetros do TSAdHoc

Alguns parâmetros influem no comportamento do algoritmo, conforme as necessidades do problema e dos recursos disponíveis e obtidos por experimentação. A citar:

- Tamanho de LT - $|T|$ ,

- Cardinalidade de Busca em $N_{open}(s_0)$ ,
- Função de Aspiração - $A(f(s_m))$ ,
- Número Máximo de Iterações (*TSMAX*).

### 5.6.3. Tamanho da LT $|T|$

A *Lista Tabu* "memoriza" movimentos realizados na lista de soluções proibidas. Ela contém uma lista reversa dos últimos $|T|$ movimentos realizados, considerados proibidos. Funciona como uma fila de tamanho fixo. Quando um novo movimento *m* é adicionado à *LT*, o mais antigo sai. Assim, na exploração de um subconjunto $V \subseteq N_{open}(s_0)$ da solução corrente $s_0$, ficam excluídos da busca os vizinhos de $s_0$ que são obtidos por movimentos *m* contidos em *LT*, a menos que $f(s) \leqslant A(f(s_m))$ . O tamanho de *LT* é definido experimentalmente.

O tamanho da *LT* é um problema encontrado na BT [HER92]. Se a *LT* for muito pequena, a probabilidade de surgimento de caminhos cíclicos indesejados torna-se maior, a partir de $T+1$ iterações. Por outro lado, se for muito grande, tem a desvantagem de impedir o retorno a uma solução gerada anteriormente e que dependem de movimentos armazenados nas últimas $|T|$ posições da fila [GLO89]. Adicionalmente, se $|T| \to \infty$ temos a inviabilidade computacional de analisar todas as possibilidades da *LT*. Uma opção intermediária é armazenar apenas as últimas $|T|$ soluções geradas, para evitar caminhos cíclicos em até $|T|$ iterações.

### 5.6.4. Cardinalidade de Busca- $N_{open}(s_0)$

A Cardinalidade de Busca em $N_{open}(s_0)$ determina subconjunto *V* da vizinhança de $s_0$ que será analisada para se obter *s*, onde $\forall s'' \in V, s \in V, V \subseteq N(s_0)_{open} \Rightarrow f(s) \leqslant f(s'')$ .

Este parâmetro não pode ser muito grande, por se tornar computacionalmente inviável analisar todas as soluções possíveis vizinhas de $s_0$. Por outro lado, não pode ser muito pequena a ponto de não permitir a escolha de uma boa solução *s*.

### 5.6.5. Função de Aspiração- $A(f(s_m))$

A Função de Aspiração tem como objetivo retirar o *ST* de um movimento. É aplicado a uma solução *s*, $s \in N(s_0)_{open}$ tal que $s \leftarrow s_0 \circ m, s \in \mathrm{LT} \Rightarrow f(s) \leqslant A(f(s_m))$ . Aceita-se o MT *m* somente se conduzir a um vizinho melhor que a melhor solução encontrada $s_m$, isto é, representa o valor que o algoritmo aspira ao chegar em $s_m$.

Se $A(f(s_m))$ apresentar valor pequeno, a aceitação de MT é maior. Isto aumenta a possibilidade de formação de caminhos cíclicos, isto é, o retorno para uma solução já visitada anteriormente. Por outro lado, pode proibir movimentos para soluções que ainda não foram visitadas e que dependem de movimentos armazenados nas últimas $|T|$ posições da fila [GLO89].

No *TSAdHoc* priorizou-se evitar a formação de caminhos cíclicos para $A(f(s_m))$ .

### 5.6.6. Número Máximo de Iterações (TSMAX)

O parâmetro *TSMAX* simplesmente determina a condição de parada do algoritmo, pelo número máximo de iterações.

### 5.6.7. Funcionamento do TSAdHoc

Descrever-se-á o funcionamento do *TSAdHoc* por meio de um exemplo. Seja a rede descrita no item *5.4.9. Funcionamento do GAdHoc*. Deseja-se obter um valor mínimo para $F_o$ para o *TSAdHoc* para o particionamento em *k-clusters*. Neste exemplo, com K=3. Para simplificar este exemplo, suponha que *TSMAX*=4 e $|T|=3$ .

Inicia-se o algoritmo com a geração de uma solução inicial, determinada aleatoriamente. Em seguida, tem-se a inicialização de $\mathrm{LT} \leftarrow \emptyset$ e $A(f(s_m)) = A$ , no instante *T=0*, conforme Tabela 14:

| **T** | $s_m$ | $F_{sm}$ | $s_0$ | $F_{so}$ |
|---|---|---|---|---|
| 0 | 00100100101010010001001 0010 | 408,000 | 00100100101010010001001 0010 | 408,000 |
| | $LT \leftarrow \varnothing$ | | | |

*Tabela 14: TSAdHoc: gerando solução inicial s0,sm=s0, em T=0*

O algoritmo prossegue gerando, ainda no primeiro laço, uma solução *s* vizinha de $s_0$, $VT(s_0): s \leftarrow m \circ s_0$ , $F_{VT(s_0)} = 100{,}000$ , sendo $s \in N(s_0)_{open}$ . $VT(s_0)$ é obtido a partir do movimento *m* do r-ésimo conjunto de k-bits da seqüência, sendo *k* o número de *clusters*, e *r* é o r-ésimo cluster, obtido de forma aleatória, segundo a condição: $s \leftarrow m \circ s_0 | (m \notin LT) \| (m \in LT \wedge f(s) < A(f(s_m)))$ , conforme Tabela 15 (movimento *m* em destaque):

| **T** | $s_m$ | $F_{sm}$ | $s_0$ | $F_{so}$ |
|---|---|---|---|---|
| 0 | 00100100101010010001001 0010 | 408,000 | 00100100101010010001001 0010 | 408,000 |
| | *VT(s0):* $s \leftarrow m \circ s_0 \| (m \notin LT)$<br>*s0 :* $s_0 \leftarrow s$<br>*sm :* $f(s) < f(s_m) \Rightarrow (s_m \leftarrow s), A(f(s_m)) \leftarrow A(f(s))$<br>*LT :* $LT \leftarrow \varnothing$ | | | |
| | | | **s** | $F_s$ |
| | | | 00100100**100**100100010010010 | 100,000 |
| | $s_m$ | $F_{sm}$ | $s_0$ | $F_{s0}$ |
| | 00100100110010010001001 0010 | 100,000 | 00100100110010010001001 0010 | 100,000 |
| | *LT :* $(f(s_0), f(s)) \notin LT \Rightarrow LT \leftarrow LT \cup (f(s_0), f(s))$ | | | |

*Tabela 15: TSAdHoc: gerando solução inicial s0,sm=s0, em T=0*

Temos que *LT* é atualizada com novo *MT*, $LT = \{(408{,}000\,;100{,}000)\}$ . Como $f(s) < f(s_m)$ , a melhor solução encontrada ( $f(s_m)$ ) é atualizada também, juntamente com $A(f(s_m))$ . A solução $s$ passa a ser $s_0$ .

Em *T=1*, suponha que $VT(s_0): s \leftarrow m \circ s_0$ , $F_{VT(s_0)} = 404{,}000$ , conforme Tabela 16 (movimento *m* em destaque):

<table>
<tr><th>T</th><th>$s_m$</th><th>$F_{sm}$</th><th>$s_0$</th><th>$F_{so}$</th></tr>
<tr><td rowspan="7">1</td><td>00100100110010010001001 0010</td><td>100,000</td><td>00100100110010010001001 0010</td><td>100,000</td></tr>
<tr><td colspan="4">$VT(s_0)$: $s \leftarrow m \circ s_0 | (m \notin \mathrm{LT})$<br>$s0$ : $s_0 \leftarrow s$<br>$sm$ : $f(s) > f(s_m)$<br>$LT$ : $\{(408{,}000\,;\,100{,}000)\}$</td></tr>
<tr><td colspan="2" rowspan="2"></td><th>$s$</th><th>$F_s$</th></tr>
<tr><td>00100100100110010001001 0010</td><td>404,000</td></tr>
<tr><th>$s_m$</th><th>$F_{sm}$</th><th>$s_0$</th><th>$F_{s0}$</th></tr>
<tr><td>00100100110010010001001 0010</td><td>100,000</td><td>00100100100110010001001 0010</td><td>404,000</td></tr>
<tr><td colspan="4">$LT$ : $(f(s_0), f(s)) \notin \mathrm{LT} \Rightarrow \mathrm{LT} \leftarrow \mathrm{LT} \cup (f(s_0), f(s))$</td></tr>
</table>

*Tabela 16: TSAdHoc: execução do algoritmo, em T=1*

Temos que *LT* é atualizada com novo *MT*, $\mathrm{LT} = \{(100{,}000\,;404{,}000)\}$ . Como $f(s) > f(s_m)$ , $f(s_m)$ é mantida, juntamente com $A(f(s_m))$ . A solução $s$ passa a ser $s_0$ . Em *T=2*, suponha que $VT(s_0): s \leftarrow m \circ s_0$ , $F_{VT(s_0)} = 100{,}000$ , conforme Tabela 17 (movimento *m* em destaque):

<table>
<tr><th>T</th><th>$s_m$</th><th>$F_{sm}$</th><th>$s_0$</th><th>$F_{so}$</th></tr>
<tr><td rowspan="7">2</td><td>00100100110010010001001 0010</td><td>100,000</td><td>00100100100110010001001 0010</td><td>404,000</td></tr>
<tr><td colspan="4">$VT(s_0)$: $s \leftarrow m \circ s_0 | (m \notin \mathrm{LT})$<br>$s0$ : $s_0 \leftarrow s$<br>$sm$ : $f(s) < f(s_m) \Rightarrow (s_m \leftarrow s), A(f(s_m)) \leftarrow A(f(s))$<br>$LT$ : $\{(408{,}000\,;\,100{,}000), (100{,}000\,;\,404{,}000)\}$</td></tr>
<tr><td colspan="2" rowspan="2"></td><th>$s$</th><th>$F_s$</th></tr>
<tr><td>00100100100100100010010 0010</td><td>100,000</td></tr>
<tr><th>$s_m$</th><th>$F_{sm}$</th><th>$s_0$</th><th>$F_{s0}$</th></tr>
<tr><td>00100100110010010001001 0010</td><td>100,000</td><td>00100100110010010001001 0010</td><td>100,000</td></tr>
<tr><td colspan="4">$LT$ : $(f(s_0), f(s)) \notin \mathrm{LT} \Rightarrow \mathrm{LT} \leftarrow \mathrm{LT} \cup (f(s_0), f(s))$</td></tr>
</table>

*Tabela 17: TSAdHoc: execução do algoritmo, em T=2*

Temos que *LT* é atualizada com novo *MT*, $\mathrm{LT} = \{(404{,}000\,;100{,}000)\}$ . Como $f(s) < f(s_m)$ , a melhor solução encontrada ( $f(s_m)$ ) é atualizada também, juntamente com $A(f(s_m))$ . A solução $s$ passa a ser $s_0$ .

Em *T=3*, suponha que $VT(s_0): s \leftarrow m \circ s_0$ , $F_{VT(s_0)} = 404{,}000$ , conforme Tabela 18 (movimento *m* em destaque e solução *s*, cujo movimento é *MT*, tachado):

| **T** | $s_m$ | $F_{sm}$ | $s_0$ | $F_{so}$ |
|---|---|---|---|---|
| 3 | 00100100110010010001001 0010 | 100,000 | 00100100110010010001001 0010 | 100,000 |
| | *VT(s₀):* $s \leftarrow m \in \mathrm{LT} \wedge f(s) > A(f(s_m))$ , $F(s)=440{,}000$<br>*VT'(s₀):* $s \leftarrow m \notin \mathrm{LT}$ , $F(s)=570{,}000$<br>*s0 :* $s_0 \leftarrow s$<br>*sm :* $f(s) > f(s_m)$<br>*LT :* $\{(408{,}000\,;100{,}000),(100{,}000\,;404{,}000),(404{,}000\,;100{,}000)\}$ | | | |
| | | | **s** | **$F_s$** |
| | | | ~~00100100100110010001001 0010~~<br>00100110010010010001001 0010 | ~~404,000~~<br>570,000 |
| | $s_m$ | $F_{sm}$ | $s_0$ | $F_{s0}$ |
| | 00100100110010010001001 0010 | 100,000 | 00100110010010010001001 0010 | 570,000 |
| | *LT:* $(f(s_0), f(s)) \notin \mathrm{LT}, \lvert\mathrm{LT}\rvert = \lvert\mathrm{T}\rvert \Rightarrow \mathrm{LT} - \{m_0\}, \mathrm{LT} \leftarrow \mathrm{LT} \cup (f(s_0), f(s))$ | | | |

*Tabela 18: TSAdHoc: execução do algoritmo, em T=3*

Dois eventos importantes ocorrem em *T=3*. Primeiramente, o vizinho $VT(s_0): s \leftarrow m \circ s_0$ , $F_{VT(s_0)}=404{,}000$ , não pode ser selecionado como *s*, pois o movimento *m* é tabu, i.e., $s \leftarrow m \circ s_0 | \mathrm{LT}=\{m_0, m_1, \ldots, m, \ldots, m_{|T-1|}\}$ . Além disso, $s$ poderia ser selecionado mesmo com $m \in \mathrm{LT}$ , desde que $f(s) < A(f(s_m))$ , mas isso não acontece. Como $s \leftarrow m \in \mathrm{LT} \wedge f(s) > A(f(s_m))$ , este movimento *m* é descartado e uma nova solução $VT'(s_0)$ é gerada. Suponha que $VT'(s_0): s \leftarrow m \circ s_0$ , $F_{VT'(s_0)}=570{,}000$ . Como $s' \leftarrow m \notin \mathrm{LT}$ , a solução é aceita como vizinha tabu de $s_0$. Como $f(s) > f(s_m)$ , $f(s_m)$ é mantida, juntamente com $A(f(s_m))$ . A solução $s$ passa a ser $s_0$ . Outro evento importante é o limite do tamanho de LT ter sido atingido. Neste caso, antes de inserirmos o novo *MT*, $\mathrm{LT}=\{(100{,}000\,;570{,}000)\}$ , devemos retirar o *MT* que está a mais tempo na lista, isto é, o primeiro elemento $m_0$ , para a inserção de $(f(s_o), f(s))$ .

Por fim, para *T=4*, suponha que $VT(s_0): s \leftarrow m \circ s_0$ , $F_{VT(s_0)}=100{,}000$ , conforme Tabela 19 (movimento *m* em destaque e solução *s*, cujo movimento é *MT*, tachado):

| **T** | $s_m$ | $F_{sm}$ | $s_0$ | $F_{so}$ |
|---|---|---|---|---|
| 4 | 00100100110010010001001 0010 | 100,000 | 00100110010010010001001 0010 | 570,000 |
| | *VT(s₀):* $s \leftarrow m \notin \mathrm{LT}$<br>*s0 :* $s_0 \leftarrow s$<br>*sm :* $f(s) > f(s_m)$<br>*LT :* $\{(100{,}000\,;404{,}000),(404{,}000\,;100{,}000),(100{,}000\,;570{,}000)\}$ | | | |
| | | | **s** | $F_s$ |
| | | | 00100100110010010001001 0010 | 100,000 |
| | $s_m$ | $F_{sm}$ | $s_0$ | $F_{s0}$ |
| | **00100100110010010001001 0010** | **100,000** | 00100100110010010001001 0010 | 100,000 |
| | *LT:* $(f(s_0), f(s)) \notin \mathrm{LT}, \lvert\mathrm{LT}\rvert = \lvert\mathrm{T}\rvert \Rightarrow \mathrm{LT} - \{m_0\}, \mathrm{LT} \leftarrow \mathrm{LT} \cup (f(s_0), f(s))$ | | | |

*Tabela 19: TSAdHoc: execução do algoritmo, em T=4*

O limite do tamanho de LT foi novamente atingido. Antes de inserirmos o novo *MT*, $m = (570{,}000\,;100{,}000)$, devemos retirar o *MT* que está a mais tempo na lista, i.e., o primeiro elemento $m_0 = (408{,}000\,;100{,}000)$, para a inserção de $(f(s_o), f(s))$. Assim: $\mathrm{LT} = \{(100{,}000\,;404{,}000),(404{,}000\,;100{,}000),(100{,}000\,;570{,}000)\}$. Como $f(s) > f(s_m)$, $f(s_m)$ é mantida, juntamente com $A(f(s_m))$. A solução $s$ passa a ser $s_0$. O algoritmo finaliza, pois $T = TSMAX$. A solução está em $f(s_m) = 100{,}000$ (em negrito, na Tabela 19).

## 5.7. DETERMINAÇÃO DE CLUSTERHEADS

Nesta segunda fase, determinam-se os *clusterheads*, a partir dos *clusters* já formados. O objetivo é determinar o nó de menor peso $W_i$. O nó selecionado será considerado líder e seus vizinhos serão considerados *clusternodes*, que não serão mais candidatos a se tornarem *clusterheads*. O processo continua com a escolha do nó de menor peso, dentro do conjunto de nós que ainda não foram considerados líderes ou membros de *clusters* já formados.

Foram propostos melhoramentos no modelo baseado em pesos [CHA00], onde cada nó possui um peso $W_{v,}$, obtido a partir do número de vizinhos (tratado no fator $\Delta_v$), do grau de mobilidade (dado por $M_v$), da energia ( fator $P_v$ ) e da distância de cada vizinho (denominado fator $D_v$).

Dada a natureza limitada de recursos de uma rede *ad hoc*, um modelo realístico deve considerar também a possibilidade de falhas dos elementos da rede. Por isso, o modelo [CHA00] foi adaptado, acrescentado-se um quinto elemento $\zeta$ - *disponibilidade*. Cada nó *v*, $v \in N$ , apresenta um grau de disponibilidade $\zeta_v$ [HWA98]:

$$\zeta = \frac{mtbf}{mtbf + mttr}$$

*Equação 8: Disponibilidade*

, onde *mtbf* é o tempo médio entre falhas e *mttr* é o tempo médio de falha. A cada unidade de tempo é verificado se um nó *i* está disponível ou não. O mesmo teste é feito se um nó *i* indisponível na unidade de tempo anterior já retornou ao estado disponível. Desta forma, o peso de cada nó $W_v$ é dado por:

$$W_v = w_1 . \Delta_v + w_2 . D_v + w_3 . M_v + w_4 . P_v + w_5 . \zeta_v$$

*Equação 9: Modelo proposto baseado em pesos*

, onde $w_i \in \mathbb{R}$ , $0 \leqslant w_i \leqslant 1$ , $\sum_{i=0}^{5} w_i = 1$ , são os pesos atribuídos a cada $W_v$. $\Delta_v$ é definido por $\Delta = |d_v - \delta|$ , onde $d_v$ é o número de vizinhos de *v*, isto é, o grau de *v*, dado por $d_v = |N(v)| = \sum_{v' \in V, v' \neq v} \{dist(v, v') < tx_{range}\}$ e $\delta$ é o número ideal de membros para *v*. $D_v$ é definido pela soma das distâncias de todos os vizinhos de *v*, $D_v = \sum_{v' \in N(v)} \{dist(v, v')\}$ . $M_v$ é o grau de mobilidade do nó *v*, expresso pela média da velocidade de *v* até o tempo *T*. É definido por $M_v = \frac{1}{T} \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}$ , onde $(x_t, y_t)$ e $(x_{t-1}, y_{t-1})$ são as coordenadas de *v* no tempo *t* e *t-1* respectivamente. $P_v$ é definido pelo tempo cumulativo que *v* permaneceu como *clusterhead*.

## 5.8. DETERMINAÇÃO DO *CONNECTED DOMINANT SET* (CDS(G))

Por fim, com a topologia gerada, é formado o *Connected Dominant Set* (CDS(G)) para a comunicação entre os *clusters* da rede, a fim de evitar o problema do raio infinito dos *clusterheads* determinados em uma topologia genérica.

Adaptou-se o modelo de Wu e Li [WUL99] para a construção do CDS(G). Considerou-se os *clusterheads* encontrados como sendo *dominators* da rede e não serão

descartados na segunda fase do algoritmo (eliminação de redundâncias). Este algoritmo apresenta duas fases: formar o CDS(G) e eliminar *dominators* redundantes.

Na primeira fase, inicialmente, todos os vértices não estão marcados, exceto os *clusterheds,* que já estão selecionados. Os nós trocam informações entre suas vizinhanças. Um vértice $u$ é marcado quando $v, w \in N_{open}(u) | não\ existir\ \overline{v, w}$ . Neste caso, $u$ é denominado *dominator*. Ao final, o conjunto de *dominators* formará o CDS(G).

Na segunda fase, é necessário eliminar o *dominator* redundantes. Um *dominator u* é eliminado do CDS(G) e desmarcado se $\exists v \in CDS(G) | N_{closed}(v) \supseteq N_{closed}(u)$ . Os *dominators* que permanecerem marcados, formarão o CDS(G) definitivo. Salienta-se mais uma vez que os *clusterheads* obtidos na fase de Determinação de *Clusterheads* já estão marcados na primeira etapa e não serão desmarcados na etapa seguinte. Exemplos da determinação de CDS(G) com visualização da rede ao fim da primeira fase, bem como, na etapa final são mostrados no Apêndice C – Visualização de Topologias.

## 5.9. PARÂMETROS DO MODELO PROPOSTO

O modelo proposto apresenta parâmetros visando torná-lo realístico:

- Reafiliação por Diferença de Potência (RDP),
- Reafiliação por Queda de Sinal (RQS),
- Disponibilidade,
- Ciclo de Vida de *v,*
- Consumo de Energia,
- Mobilidade,
- Regras de Borda,
- Vizinhança Aberta de *v,*
- Rearranjo de Topologia.

### 5.9.1. Reafiliação por Diferença de Potência (RDP)

Uma reafiliação (*handoff*) ocorre quando um nó *v* abandona o seu *cluster* atual

$C_i(v)$ , cujo *clusterhead* é $h_i(v)$ , e passa a fazer parte de um *cluster* $C_j(v)$ , $i \neq j$ . O nó $v$ passa a receber também o sinal do *clusterhead* $h_j(v)$ , $DS(G)=\{h_0, h_1, \ldots, h_i, \ldots, h_j, \ldots, h_{K-1}\}$

Uma Reafiliação por Diferença de Potência (RDP) ocorre quando um nó *v* recebe o sinal com maior potência, $\rho$ , de um *clusterhead* vizinho $\rho(h_j(v))$ , em relação ao *clusterhead* atual $\rho(h_i(v))$ . Acrescentou-se um valor limiar (*threshold*)- $\tau$ , com agregação de histerese (*relative signal strength with hysteresis and threshold*). Esta adaptação evita problemas relacionados ao efeito *ping-pong* em uma reafiliação [STA01]. Sendo assim, uma RDP ocorre quando $C_i(v), \exists h_j(v) \in DS(G) | \rho(h_i(v)) + \tau \leqslant \rho(h_j(v))$ ou $C_i, \exists h_j(v), h_m(v) \in DS(G) | \rho(h_i(v)) + \tau \leqslant \rho(h_m(v)) \leqslant \rho(h_j(v))$ .

### 5.9.2. Reafiliação por Queda de Sinal (RQS)

Outro parâmetro presente no modelo proposto é a Reafiliação por Queda de Sinal (RQS). Ocorre em *v*, quando há falha do seu *clusterhead* ou simplesmente por movimentação abrupta, por $\rho(h(v)) \rightarrow 0$ . Para o caso de movimentação abrupta, o nó *v* é reafiliado para outro *cluster* $C_j$ , cujo *clusterhead* é $h_j$ , da seguinte forma: $\rho(h_i(v)) \rightarrow 0, \forall \rho(h'_j(v)) \geqslant 0, \exists h_j(v) | \rho(h_j(v)) \geqslant \rho(h'_j(v)) \geqslant \rho(h_i(v))$ .

### 5.9.3. Disponibilidade

A disponibilidade $\zeta$ é um melhoramento em relação ao modelo proposto por [CHA00] para tornar o cenário de simulação mais realístico. Cada nó *v* da rede apresenta um coeficiente de disponibilidade $\zeta_v \in \mathbb{R}$ , $\zeta_v = [0..1]$ . Este fator apresenta duas componentes $\zeta_v = \zeta_{BASE} + \zeta_{RAND}$ , onde $\zeta_{BASE}$ é um valor base para toda a rede e $\zeta_{RAND} = [0..(1 - \zeta_{BASE})]$ é um valor aleatório para cada *v*. Durante a simulação, os nós são verificados se estão disponíveis ou não, por um atributo de controle *falha* $(\pi)$ . Se houver falha, $\pi_v = 0$ , se estiver disponível, $\pi_v = 1$ .

### 5.9.4. Ciclo de Vida de *v*

O Ciclo de Vida de um nó *v* mapeia todos os estados $\Gamma$ possíveis dentro do modelo proposto. São eles: ativo, ativo com economia de energia e falho.

Inicialmente, todos os nós estão ativos, $\Gamma_v = ativo$ , ou seja, com energia $P_v$ maior que um patamar de energia mínima $P_{min}$ (na verdade, iniciam com energia máxima $P_{max}$ ), disponíveis e transmitindo informação. Então, $\Gamma_v = ativo \Rightarrow P(v) > P_{min}, \pi_v = 1$ . Neste estado, *v* poderá se movimentar e fazer parte da organização da topologia, seja como *clusterhead* ou como *clusternode*.

O estado ativo com economia de energia, $\Gamma_v = ativo\text{-}$ , temos $\pi_v = 1$ , sendo, $P_{min} \geqslant P_v \geqslant 0$ ou $P_v \geqslant P_{min}$ se não está transmitindo(*standby*). O objetivo deste estado intermediário é prolongar a vida de nós com baixa energia ou que não estão transmitindo. Caso $P_{min} \geqslant P_v \geqslant 0$ , o nó *v* ainda participa da organização da topologia, com a restrição de não poder se tornar *clusterhead*, para não ficar exaurido mais rapidamente e não comprometer a rede.

O estado falho, $\Gamma_v = falho$ , ocorre quando $\Gamma_v(t-1) = \{ativo, ativo\text{-}\}$ ou simplesmente, $\Gamma_v(t-1) = \{ativo*\}$ , com os seguintes eventos: $\pi_v = 0 \vee P_v \geqslant 0$ . Se $\pi_v = 0$ , *v* é penalizado, não participa da topologia em *t*, não se move e o $\Gamma_v(t) = falho$ . Quando $\pi_v = 1$ , *v* retorna ao estado anterior, ativo ou ativo com economia de energia. Mas, se $P_v = 0$ , temos a exaustão completa de *v,* quando está completamente sem energia. Neste caso, o nó é considerado morto e descartado definitivamente da rede.

A Figura 53 mostra o diagrama de estados do Ciclo de Vida de *v*:

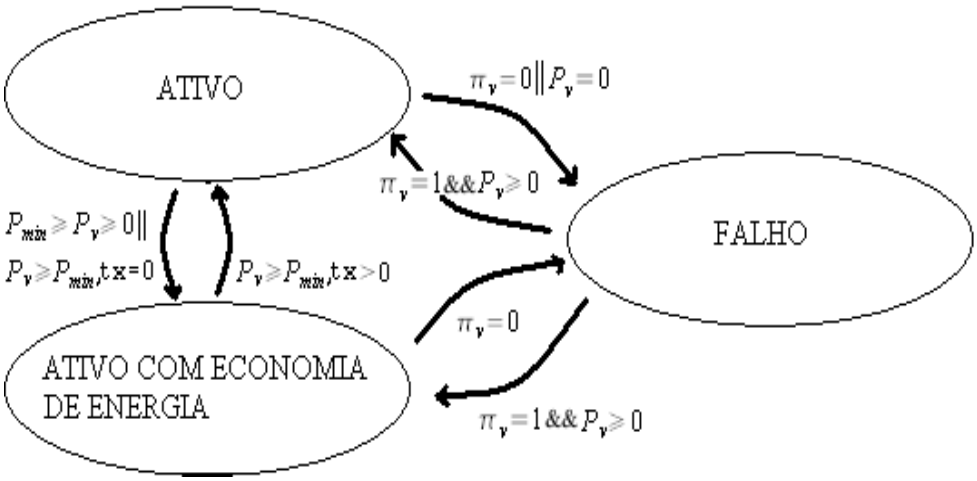


*Figura 53: Ciclo de Vida dos nós da rede*

### 5.9.5. Consumo de Energia

O modelo proposto implementa consumo de energia na rede $P$, de acordo com as diferentes atividades executadas pelo nós durante a simulação. Inicialmente, $\forall v \in G, P(v)_{max} = 1$ e sofrem degradação à medida que ser tornam *clusterheads* $P(v) = P(v) - P(v)_{clusterhead}$, *clusternode* $P(v) = P(v) - P(v)_{clusternode}$ ou se movem $P(v) = P(v) - P(v)_{mobilidade}$, onde $P(v)_{clusterhead}$, $P(v)_{clusternode}$, $P(v)_{mobilidade}$ são os custos das respectivas atividades. A degradação de $P(v)$ ao longo da simulação exercerá influência direta em $W_v$ e $\Gamma_v$.

### 5.9.6. Mobilidade

Adotou-se no presente as regras de mobilidade, SRMM [BET01b] e RPGM [GER99b], em conjunto com BSAMM [HAA97b], com o objetivo de tornar o movimento da rede mais realístico e minimizar rearranjos indesejáveis da topologia por movimentações bruscas ou não-realísticas, procurando a manutenção ao longo do tempo da relação $\psi_{intra} / \psi_{inter}$, do número de reafiliações e atualizações de $DS(G)$.

### 5.9.7. Regras de Borda

O presente estudo considera que a progressão da rede não é realizada em um terreno infinito. Logo, quando um nó atinge algum limite dimensional ou borda, precisa ser deslocado de forma a permanecer dentro do espaço de progressão válido. O modelo

proposto adota a regra de borda *bounce* [HAA98] por tornar mais realístico o cenário de simulação, dentro destas condições.

### 5.9.8. Vizinhança Aberta de *v*

No modelo proposto, a vizinhança aberta de um nó ativo*, $N(v)_{open}$ , é dado por $N(v)_{open}=\{\forall\, w\in G|\overline{vw}=\overline{wv}\leqslant tx_{range}, \Gamma(v), \Gamma(w)=\{ativo*\}, v\neq w\}$ , onde $tx_{range}$ é o raio de transmissão máxima dos nós de *G*.

### 5.9.9. Rearranjo de Topologia

Um rearranjo é uma organização da topologia no instante *t'*, com *t'>t*, e desde que em t, *t>0*, um arranjo tiver acontecido. Isto significa que uma nova execução do algoritmo proposto é realizada. Seja $h_i$, $h_j$ *clusterheads* de $C_i$, $C_j$, respectivamente, $C_i, C_j\in C$ , $i, j\in[0, K-1]$ e $v\in C_i, v\neq h_i$ . Isto acontece quando:

i. Há falta de adjacência , i.e., se *v* abandona $C_i$, em um instante *t*, $t=[1, T_{exec}]$ e não consegue reafiliar-se a outro, em *t+1*. Em uma tentativa de manter a rede interligada, um novo arranjo de topologia é executado;

ii. Ocorre falha de um *clusterhead*, i.e., se $\Gamma_{hi}(t)=\{falho\}$ ,t>0. Em conseqüência $\rho(h_i(v))\rightarrow 0$ , $\forall\, v\in C_i, v\neq h_i$ . Um novo rearranjo é realizado para o estabelecimento da comunicação destes nós, considerando uma abordagem mais geral;

iii. $\sum_{k=0}^{K-1}(\psi_{C_k}(t-\Delta t))<0$ , onde $C_k$ é o k-ésimo *cluster*, $\psi_{C_k}(t-\Delta t)$ é

$\psi_{C_k}(t-\Delta t)=(\sum_{i=0}^{nodes_k}\psi(v_i)_{intra}(t-\Delta t)-\sum_{i=0}^{nodes_k}\psi(v_i)_{inter}(t-\Delta t))$ onde i é o i-ésimo nó do k-ésimo *cluster* . Esta informação é armazenada em cada $h_k(v)$ , no instante anterior $t-\Delta t$ . O valor de $\Delta t$ se torna ótimo quando $\Delta t\rightarrow 0$ . Para a implementação em máquinas portáteis, isto se torna um desafio. Para efeito de simulação, considerar-se-á o tempo discreto $\Delta t=1$ .

## 5.10. PASSAGEM DE MENSAGENS

### 5.10.1. Introdução

O objetivo desta etapa é definir procedimentos básicos, com base em troca de mensagens entre elementos da rede, para a determinação de *clusters*, identificação de *clusterheads*, encaminhamento de mensagens e identificação de um nó reafiliado ao novo *clusterhead.* Também é realizada uma análise da complexidade das mensagens em cada procedimento. Trabalhos anteriores da literatura mostram diferentes algoritmos para satisfazer a estas necessidades básicas: [CHA97],[BAS99a], [BAS99b], [CHA00]. Estas abordagens não são generalistas e cada uma se restringe a um determinado domínio de problema.

Uma abordagem para a determinação de *clusters* apresenta complexidade de mensagens *O(n)*, onde *n* é o número de nós [BAS99a]. Este trabalho se restringe a uma rede sem mobilidade (quase estática) e não considera reafiliações de nós entre *clusters*, durante o processo de *clustering*. Por outro lado, o DMAC, *distributed mobility-adaptive clustering*, é um algoritmo distribuído que aceita modificações da rede e a complexidade de comunicação entre um *clusternode* e o seu *clusterhead* é $\Theta(1)$ [BAS99b]. É restritivo quanto a estratégia de minimização de fluxo inter-clusters. Outro trabalho relacionado utiliza CDS(G) agregado à capacidade dual de transmissão dos *clusterheads*, em baixa dentro do *cluster* e em alta fora do *cluster*, este com maior raio de transmissão. O encaminhamento de mensagens entre dois nós é realizado com $O(|clusterheads|)$ , onde $|clusterheads|$ é a cardinalidade de DS(G) [CHA00]. Este modelo se restringe somente a *dominant sets* conexos. Por fim, é apresentado em [CHA97] um modelo que utiliza uma lista para identificar nós e seus respectivos *clusters*, chamada de *Clust_List*. Realiza a troca de mensagens em $O(B+\Delta_{max})$ , onde B é o número de nós de borda. Este trabalho se restringe a aceitar apenas *clusters* sobrepostos.

A idéia básica é adaptar aspectos de cada modelo julgados vantajosos e adequados ao modelo proposto, dentro do contexto de satisfazer os procedimentos básicos mencionados.

Neste trabalho, não abordaremos a análise de algoritmos de roteamento para redes *ad hoc*. Esperamos, no entanto, que o modelo proposto contribua para um melhor

roteamento de mensagens na rede. São algumas técnicas de roteamento: *Dynamic Source Routing* [JOH96], *Ad Hoc On-demand Distance Vector Routing* [PER99] e FSR [GER99a].

### 5.10.2. Mensagens para a identificação de particionamento em clusters

É enviada em *broadcast* uma mensagem *Clust_List*. Trata-se de uma lista contendo a identificação dos nós e seus respectivos *clusters*, determinados pelo MHAdHoc. Para efeito de simplificação do problema, vamos supor que o MHAdHoc é executado em um ponto da rede e o *broadcast* será inicializado a partir dele. Cada nó recebe a mensagem e verifica em *Clust_List* qual é o *cluster* associado ao seu ID. Um exemplo é mostrado na Tabela 20, supondo N=9 e K=3.

A complexidade de mensagens é dada por $O(n^2)$ .

| *ClusterId* | *Nodes* |
|---|---|
| 0 | 0,1,2 |
| 1 | 3,4,5 |
| 2 | 6,7,8 |

*Tabela 20: Exemplo de lista Clust_List*

### 5.10.3. Mensagens para a identificação de clusterheads

A determinação de *clusterheads* ocorre a partir dos *clusters* já formados. O algoritmo para notificar aos *clusternodes* sobre quem é o *clusterhead*, no âmbito de cada *cluster*, é realizada com base no procedimento *On receiving Ch(u)* [BAS99b], onde *Ch(u)* é uma mensagem informando que o nó *u* é o *clusterhead* [BAS99a]. Como este modelo considera a distância de um hop de *v*, $v \in N_{open}(u)$ , acrescentou-se um novo parâmetro ao procedimento, $C_k$, que informa o ID do cluster.

Quando *v* recebe a mensagem, se $v \in C_k$ , armazenará a identificação do seu *clusterhead,* senão, simplesmente, descarta a mensagem. A garantia de que todos os nós receberão as mensagens dos seus respectivos *clusterheads* é dada pela conectivadade de cada *cluster*. Para dar conhecimento ao *clusterheads* sobre quem são os seus *clusternodes*, é enviada uma mensagem *JOIN(v,u)* [BAS99a], indicando que *v* deseja se tornar

*clusternode* de *u*. O procedimento *On receiving Ch(u,$C_k$)* é descrito na Figura 54.

```
On receiving CH(u,Ck)
{
        if(Ck == C(v))
        {
            send JOIN(v,u);
            Clusterhead:=u;
            send to other neighborhors message CH(u);
        }
        else
        {   discard message;
        }
}
```

*Figura 54: Procedimento On receiving CH(u,Ck)*

A complexidade de mensagens, no pior caso, ocorre quando há apenas um *clusterhead* em uma rede com *n* nós, com *n-1* mensagens. Logo, temos $O(n)$ .

### 5.10.4. Mensagens para dar conhecimento de um novo nó ao clusterhead

Outro procedimento básico é como dar conhecimento de um novo nó a um *clusterhead*. Adaptou-se ao modelo proposto, o procedimento *On receiving JOIN(u,v)* [BAS99b], solicitando que o nó *u* seja membro do *clusterhead v*.

Se $h(u)_{t-1}=v'$, $v' \in DS(G)$ , então $h(u)_t=v$ e *C(v)*. Senão, para $h(u)=v$, em t-1, então C(v) é mantido, conforme Figura 55.

```
On receiving JOIN(u,v)
{
        if(h(u)t-1!=v)
        {
            Ci=Ci∪{u}   ,v is Ci clusterhead;
            h(u)t=v;
        }
        else
        {   just a confirmation that u is in Ci, v is Ci clusterhead;
        }
}
```

*Figura 55: Procedimento On receiving JOIN(u,v)*

A complexidade de mensagens para o pior caso é dada por $O(|clusternodes|)$ , onde $|clusternodes|$ é a cardinalidade do número máximo de

elementos permitido por *cluster*.

### 5.10.5. Mensagens entre nós da rede

O *dominant set* do modelo proposto é do tipo CDS(G), i.e., estabelece um *backbone* virtual [CHE03] na rede. [CHA00] descreve um modelo para encaminhamento de mensagens dentro e fora de cada *clusterhead* para CDS(G). Este método foi adaptado ao presente estudo.

Se um nó $v_a$ deseja estabelecer conexão com um nó destino $v_b$, então $v_b$ necessita descobrir a rota até $v_b$. O nó $v_b$ envia uma mensagem *route discovery request* contendo o $ID_{vb}$ para o seu *clusterhead* $h(v_a)$. Se $v_b$ não estiver no mesmo cluster de $v_a$, então o $h(v_a)$ retransmite a requisição para os *clusterheads* vizinhos. Ao receber as mensagens, cada *clusterhead* verifica os nós afiliados em uma lista (constituída pela composição de requisições JOIN). Se $v_b$ for encontrado, então uma mensagem *reply* é enviada por $v_b$. Percorre o *backbone* virtual até chegar ao no $v_a$.

A complexidade de pior caso ocorre quando o DS(G) está conectado por um grafo em linha, i.e., $O(|clusterheads|)$ , onde $|clusterheads|$ é a cardinalidade do DS(G).

### 5.10.6. Complexidade de Mensagens

A partir do exposto, a complexidade de mensagens é dada por:

- Complexidade de mensagens para Identificação de Particionamento em Clusters: $O(n^2)$ ;
- Complexidade de mensagens para a identificação de clusterheads, dados os clusters já formados: $O(n)$ ;
- Complexidade de mensagens para dar conhecimento de um novo nó ao clusterhead: $O(|clusternodes|)$
- Complexidade de mensagens entre nós da rede: $O(|clusterheads|)$ .

## 5.11. ANÁLISE DE COMPLEXIDADE

Uma característica muito importante de qualquer algoritmo é o seu tempo de execução. É possível obter uma boa aproximação teórica de tempo de execução real de um algoritmo por meio de métodos analíticos. O objetivo é determinar uma função matemática que traduza o comportamento de tempo de um algoritmo, de forma independente do computador utilizado, da linguagem e compiladores empregados e das condições locais de processamento As notações O e $\Omega$ descrevem, respectivamente, o limite superior e inferior assintóticos. A notação $\Theta$ descreve o limite superior justo (O e $\Omega$ ) [SWA94].

Esta análise limitar-se-á ao pior caso de tempo e espaço da Busca Tabu, por ter obtido os melhores resultados experimentais, em relação ao SA e ao GA.

### 5.11.1. Complexidade de Tempo

A complexidade de tempo do algoritmo básico da Busca Tabu é dada por $O(\text{MAX\_ITER}\, n^2 k)$ , sendo $n=|V|$ e $k$ igual ao número de *clusters*.

Detalhes de cálculo, algoritmo básico e procedimentos adicionais são apresentados no Apêndice VI.

Além da implementação do algoritmo básico, alguns procedimentos adicionais foram agregados para adequar o modelo TS ao problema em estudo e compor o modelo proposto TSAdHoc.

Houve a necessidade de desenvolver um procedimento de validação do movimento $m$, $s' \leftarrow s_0 \circ m$ , para a obtenção de um vizinho $s'$ de $s_0$. Observou-se experimentalmente que a escolha aleatória de uma linha de $s_0$ para realizar o *shift* para a direita ou para esquerda pode gerar uma solução absurda em dois casos: pelo ingresso de um nó não adjacente ao *cluster* de destino ou quando o nó movimentado torna o *cluster* de origem desconexo. A solução foi implementar o método *isValidMovement* na classe *TabuSearch*. A complexidade de tempo deste método é $O(n^2)$ . Detalhes de cálculo também são apresentados no Apêndice VI.

Outra necessidade foi desenvolver um mecanismo para validar a solução $s_0$, no

procedimento de geração de uma solução inicial (*GenerateInitialSolution()*). Observou-se experimentalmente que a geração aleatória da matriz $G_{ik}(n,k)$, constituinte de $s_0$, pode conduzir a *clusters* desconexos, não correspondendo a uma solução realística. A solução foi implementar o método *isValidSolution* na classe *Solution*. A complexidade de tempo deste método é $O(n^2 k)$ .

### 5.11.2. Complexidade de Espaço

A complexidade de espaço do algoritmo básico da Busca Tabu é dada por $O(nk)+O(|T|)=O(nk+|T|)$ . Detalhes acrescentados na implementação não influenciaram na complexidade de espaço computada teoricamente.

### 5.11.3. Conclusão

O TSAdHoc foi implementado exatamente conforme o algoritmo TS. Os detalhes acrescentados na implementação não influenciaram na complexidade de tempo e espaço computada teoricamente para o método. A análise teórica realizada se limita ao escopo do problema e não se trata de uma análise genérica.

Como o particionamento de uma rede em *clusters* é um problema NP-completo, possivelmente não há uma solução polinomial ótima. Portanto, este trabalho contribui para tentar resolver este problema.

# 6-ESTUDOS DE CASO

## 6.1. INTRODUÇÃO

Este capítulo apresenta e analisa resultados de simulações de desempenho dos algoritmos propostos *GAdHoc* [GAR05], *SAdHoc* e *TSAdHoc*, a fim de analisar vantagens e desvantagens de cada um e concluir, de forma geral, o melhor. Estes modelos também são comparados com outras técnicas já existentes para comparação de resultados e validação do presente estudo, a citar: *Weighted Clustering Algorithm* (WCA) e *Highest-Degree Algorithm* (HDA).

Nestas análises, utiliza-se como parâmetros de desempenho o resultado final de cada técnica, isto é, o valor da Função Objetivo $F_o$ alcançado, para medir a qualidade da técnica, e no tempo necessário para se obtê-la, ou seja, no Tempo de Execução $T_{exec}$ . O objetivo dos modelos propostos é realizar o particionamento *k-clustering* da rede e determinar DS(G), dados os *clusters* já formados em um instante de tempo *t*.

Apresentamos também uma análise do comportamento da topologia de *clusters* já formada, à medida que os nós se movimentam, realizam RDP e RQS, consomem energia, transmitem dados, falham, mudam de estados, conforme o seu ciclo de vida, e participam da organização da rede, se ativos, como *clusterhead* ou *clusternode*. Avalia-se a necessidade de rearranjo de topologia conforme critérios de desempenho em um determinado instante de tempo. É desejável que um bom modelo de particionamento em *clusters* minimize o número de rearranjos de topologia. Outro fator importante ligado à topologia já formada ao longo do tempo *t* é avaliar o comportamento da rede em função das reafiliações (*handoffs*) de nós da rede, não somente ao seu quantitativo, mas particularmente aos nós reafiliados que forçam rearranjos de topologia. Uma boa topologia deve ser menos suscetível à rearranjos de topologia por nós reafiliados. Por fim, outro fator importante para uma boa topologia é o número de rearranjos da rede ao longo de *t*. Deseja-se que seja pequeno.

Para tanto, os algoritmos propostos *GAdHoc* [GAR05], *SAdHoc* e *TSAdHoc*

foram analisados entre si e com as técnicas WCA e HDA, a fim de analisar vantagens e desvantagens de cada um e concluir, de forma geral, o melhor.

Outro estudo apresentado está relacionado a determinados parâmetros das MH em estudo. Torna-se importante uma análise à parte de alguns parâmetros por serem obtidos experimentalmente e variam não só conforme o problema e sua complexidade, mas sim, a cada execução do algoritmo. Analisa-se a obtenção da Temperatura Inicial ($T_0$) no SA e o ciclo de aquecimento e resfriamento do sistema no SA.

Por fim, um melhoramento no processo de seleção de vizinhos $s$ de $s_0$ no capítulo 05 é proposto baseado em Extremidades de Partida (EP), chamado de Seleção por Extremidades de Partida (SLEP). Experimentalmente, foram obtidos melhores resultados e com menor tempo de execução $T_{exec}$ , em relação a outros critérios de seleção tradicionais, a citar: Seleção Aleatória (SA) e Busca Local (BL).

## 6.2. SIMULAÇÃO

Seja uma topologia $G$ constituída por $N$ nós dispostos em um *grid* 100x100. Na posição inicial, os nós estão posicionados de forma uniformemente distribuída. Os nós podem se mover em todas as direções aleatoriamente, conforme regra de mobilidade e de borda, dentro do espaço simulado. O número de nós $N$ varia de 20 a 50. O raio de transmissão ($rx$) apresenta valor variável, dentro de cada simulação, com $rx=[rx_{baixo}; rx_{alto}]$ ou simplesmente $rx=[r,R]$. De acordo com o ambiente de simulação, verificou-se experimentalmente que $rx_{baixo} = r = 20$, $rx_{alto} = R = 50$.

Toda a rede inicia no estado ativo $\Gamma(v)=ativo, \forall v \in G$ , com energia máxima $P(v)_{max}=1$ e $\pi(v)=1$ . Apresenta, também, disponibilidade $\zeta(v) \in \mathbb{R}$ , $\zeta(v)=\zeta_{BASE}+\zeta(v)_{RAND}$ , $\zeta(v)=[0..1]$ , onde $\zeta_{BASE}=0{,}96$ , $\forall v \in G$ , valor obtido experimentalmente, e $\zeta(v)_{RAND}=[0..(1-\zeta_{BASE})]$ é um valor aleatório uniformemente distribuído. A disponibilidade de cada nó é verificada a cada unidade de tempo (*ut*), podendo passar para os estados descritos na Figura 53. O mesmo teste é feito se um nó $v$ , $\Gamma(v)_{t-1}=falho$ , indisponível na unidade de tempo anterior já retornou ao estado disponível, se $\pi(v)_t=1$ , obtido aleatoriamente. Uma RDP ocorre quando $C_i(v), \exists h_j(v) \in DS(G) | \rho(h_i(v))+\tau \leqslant \rho(h_j(v))$ ou quando existir mais de um

*clusterhead* alcançável pelo nó reafiliado *v*, $v \in G$, isto é, $C_i, \exists h_j(v), h_m(v) \in DS(G) | \rho(h_i(v)) + \tau \leqslant \rho(h_m(v)) \leqslant \rho(h_j(v))$. Uma RQS ocorre quando a potência do sinal do *clusterhead* $h(v)$ cai a zero por afastamento abrupto entre o nó *v*, $v \in N_{open}(h)$ ,e o seu *clusterhead* ou simplesmente por falha de $h(v)$, isto é, $\rho(h_i(v)) \to 0$ .O nó *v* é reafiliado para outro *cluster* $C(j)$, cujo *clusterhead* é $h(j)$ ,da seguinte forma: $\rho(h_i(v)) \to 0, \forall \rho(h'_j(v)) \geqslant 0, \exists h_j(v) | \rho(h_j(v)) \geqslant \rho(h'_j(v)) \geqslant \rho(h_i(v))$ . A vizinhança de *v* é $N_{open}(v) = \{ \forall w \in G | \overline{vw} = \overline{wv} \leqslant tx_{range}, \Gamma(v), \Gamma(w) = \{ativo*\}, v \neq w\}$

A mobilidade adotada nos modelos *GAdHoc* [GAR05]*, SAdHoc e TSAdHoc* foi baseado nos modelos SRMM [BET01b], BSAMM [HAA97b] e *RPGM* [GER99b], este último nos casos de mobilidade em grupo. Isto significa que, para *t>0*, *v(t)* e *a(t)* não variam de forma aleatória. Utilizou-se como base o modelo BSAMM [HAA97b] para criar um raio de curvatura mínimo $\theta$ , obtido experimentalmente, com $\theta(t+1) = \theta(t) + \Delta\theta$ , $\Delta\theta = [0..\pi/4]$ e velocidade *v* definida por $v(t+1) = min[max[0, v(t) + \Delta v], v_{max}]$ , posicionados em um terreno bidimencional, onde $x(t+1) = x(t) + v(t) * \cos\theta(t)$ e $y(t+1) = y(t) + v(t) * sen\theta(t)$ .

As mudanças de direção, $\Delta\theta$ , ocorrem de forma suavizada, com base em uma adaptação simplificada do modelo SRMM [BET01b]. Esta adaptação visa minimizar a possibilidade de mudanças bruscas de direção. Isto foi feito com a finalidade de tornar o modelo mais realístico. Divide-se a variação de direção, $\Delta\theta$ , ocorrida em uma unidade de tempo em frações menores de tempo $\sum_{i=0}^{n} \varphi_i(t/n) = \varphi_0(t/n) + \varphi_1(t/n) + ... + \varphi_n(t/n)$ , onde *n* é o número de frações sobre uma unidade de tempo. Sendo *n=2*, então temos: $\sum_{i=0}^{2} \varphi_i(t/n) = \varphi_0(0) + \varphi_1(1/2) + \varphi_2(1)$ .

Para os casos de mobilidade em grupo, foi adotado o modelo *RPGM* [GER99b]. A mobilidade de um grupo é representada por um vetor resultante $\vec{R}_j$ , onde *j* é o j-ésimo grupo. Cada nó *i* pertencente à área geográfica deste grupo apresentará como velocidade a resultante entre a própria velocidade $\vec{v}_i$ e $\vec{R}_j$ do seu *cluster* correspondente. Desta forma $\vec{v}_i = \sum_{x=1}^{X} (\vec{v}_{i_x} + \vec{R}_j)$ , onde $i \in G, X \in C$ . Sendo o terreno

bidimencional, *X=2*. Para cada *cluster*, temos $\vec{v}_i = \sum_{x=1}^{2} (\vec{v}_{i_x} + \vec{R}_j), \forall i \in G, \forall j \in C$ . Por fim, todas as simulações foram realizadas atribuindo a cada nó velocidade *v* e aceleração *a*, distribuídos uniformemente no início da simulação.

A regra de borda adotada foi o modelo *bounce* [HAA98], [BET01]. Caso algum nó atinja o limite do terreno, será refletido. Para as bordas superior e inferior $\theta(t-1) = \theta \Rightarrow \theta(t) = -\theta$ e para as bordas laterais $\theta(t-1) = \theta \Rightarrow \theta(t) = \Phi - \theta$ , onde $\Phi = \pi$ . Se um nó atingir outro limite do terreno após uma reflexão (se estiver no "canto" do terreno), então $\theta(t-1) = \theta \Rightarrow \theta(t) = -\theta \Rightarrow \theta(t') = \theta$ caso o primeiro limite seja a borda inferior ou superior e $\theta(t-1) = \theta \Rightarrow \theta(t) = \Phi - \theta \Rightarrow \theta(t') = \theta$ , caso o primeiro limite seja uma borda lateral. Este critério foi escolhido por ser mais realístico.

O particionamento é empregado para um *k cluster*, irá variar entre os seguintes valores: a partir de 2, já que 1 é uma solução trivial, até o limite inferior de $\lfloor N/2 \rfloor$ .

O desenvolvimento da simulação foi feito em linguagem C++, *Shell Script/Gnuplot* [GNU06], para automatização de tarefas e plotagem de gráficos e foi utilizada a biblioteca de funções para algoritmos genéticos, chamada *libGA* [COR05]. Parte do ambiente foi desenvolvido em linguagem JAVA (JAVA2D). A modelagem do sistema foi feita utilizando o Umbrello UML Modeller, versão KDE Linux [UMB06].

### 6.2.1. Parâmetros de simulação do GA

Para o GA, foram utilizados os seguintes parâmetros: População Inicial $(P)$ - A população inicial de cromossomos foi gerada aleatoriamente. O tamanho desta população foi obtida experimentalmente, isto é, a partir de um tamanho mínimo $(P_{min})$ até o máximo $(P_{max})$ . Aumentou-se $P_{min}$ até a obtenção de resultados convergentes, mas sem que haja exaustão de recursos computacionais. Neste caso, $P_{min} = 35$ até $P_{max}$ . Para um PC Pentium M Centrino com 1.73MHz e 1GB RAM, obteve o valor de $P = 300$ . Seleção - Foi utilizado o método da roleta (*roulette*) para a seleção dos cromossomos mais aptos. Durante o processo de simulação, verificou-se que a formação dos pares de cromossomos mais aptos feita de forma aleatória convergia lentamente para um mínimo local. Um comportamento similar foi observado para o método do torneio. *Crossover* - Foi escolhido um ponto de corte aleatório *c*, $c = [0; N-1]$ . Durante o

*crossover*, um novo valor *c* foi gerado aleatoriamente com distribuição uniforme para cada par de cromossomos selecionado. Mutação (θ) - Verificou-se experimentalmente que se a taxa de mutação escolhida for alta, os resultados convergem lentamente para uma solução ótima local. Os mesmos resultados são obtidos se a taxa de mutação for nula. Utilizou-se, então, um valor intermediário $\theta = 0{,}5\%$ . Taxa de Redução Populacional – determina a redução populacional a cada ciclo do GA. Verificou-se experimentalmente que se a taxa de redução for rápida, resultados em mínimos locais indesejados, mas com tempo de execução menor. Por outro lado, se a taxa de redução for lenta, a tendência de convergir para um ótimo local é maior, mas com um tempo de execução maior, podendo em alguns casos causar exaustão de recursos computacionais e a conseqüente saída anormal da execução do programa. Utilizou-se então a taxa de 5%.

### 6.2.2. Parâmetros de simulação do SA

Para o SA, foram utilizados os seguintes parâmetros: Coeficiente de Cristalização $\delta$ - determina a parada na execução do algoritmo após $\delta$ ciclos sem que haja mudança do arranjo do sistema. Dizemos então que o sistema atingiu um estado de cristalização. Supor que o estado de cristalização é atingido após um número pequeno de ciclos $\delta_{min}$ é um grande risco, pois o sistema pode ser encerrado com um suposto estado estável que poderia ser modificado após $\delta_{min} + \epsilon$ , onde $\epsilon$ é um número pequeno de ciclos. Experimentalmente, caracterizou-se a cristalização com $\delta = 20$ . Razão de Resfriamento $\alpha$ - determina a diminuição da temperatura em cada iteração até atingir o congelamento do sistema. Verificou-se experimentalmente que $\alpha <<< 1$ , a convergência é rápida, podendo atingir mínimos locais indesejados, mas com tempo de execução menor. Adotou-se uma estratégia de convergência bem lenta para o sistema para evitar estados de cristalização indesejados, com $\alpha = 0{,}99$ . Observou-se que o tempo de execução foi maior, mas não houve caso de exaustão de recursos computacionais durante as simulações. Temperatura inicial ($T_0$) – determina o estado de aquecimento antes da execução propriamente dita do SA. Deve ser alta o suficiente para proporcionar um alto grau de agitação de moléculas do sistema. No SA, esta grande entropia se traduz na possibilidade de aceitação de vizinhos. Quanto mais "quente" uma solução $s_0$, maior a possibilidade de aceitação de um vizinho *s*. Um problema encontrado na determinação de $T_0$ é a sua

variação conforme o problema e a complexidade do problema, dado pelo número de nós e a sua disposição no terreno. Verificou-se que seria necessário obter $T_0$ de forma experimental em cada execução, para que o estado de aquecimento do sistema seja ideal, quente o suficiente para proporcionar alto grau de agitação de moléculas, mas não em demasia, que dificulte a convergência para uma solução quase ótima. Para a obtenção de $T_0$, inicia-se com $T_{baixa}$. Em cada ciclo, verifica-se o grau de aceitação de vizinhos $s$ de $s_0$. O processo termina quando atinge $p_{aceitacao}(T_0)=95\%$. Considerou-se $T_{baixa}=100$. Temperatura de congelamento ($T_c$) – depende do problema proposto e da sua complexidade, em particular do número de nós do sistema e sua disposição no terreno. Outra dependência é com relação a temperatura inicial do sistema ($T_0$) que é variável, mesmo em diferentes execuções de um mesmo cenário. Como o congelamento do sistema é apenas uma das condições de parada do SA, optou-se então por $Tc$ extremamente baixo, $T_c$=0. Probabilidade de Aceitação ($p_{aceitacao}$), com $p_{aceitacao} = r < \mathrm{e}^{(-\Delta(f)/T)}$, onde $r = [0,1]$, obtido aleatoriamente em distribuição uniforme. O número máximo de iterações (*SAMAX*), considerou-se $SAMAX=50$.

### 6.2.3. Parâmetros de simulação do TS

Para o TS, foram utilizados os seguintes parâmetros: Tamanho da LT ($|\mathrm{LT}|$) - verificou-se experimentalmente que $|\mathrm{LT}|$ deve ser grande o suficiente para impedir movimentos cíclicos na execução do TS. Por outro lado, não deve ser muito pequeno, o que poderá dificultar a convergência da solução para um resultado quase ótimo. Considerou-se, então, $|\mathrm{LT}|=20$. Cardinalidade de busca ($|N(s)_{open}|$), para a obtenção de $s$, $s \in V$, $V \subseteq N(s_o)_{open}$, com $f(s) \leqslant f(s''), \forall s'' \in V$. A partir de um cenário de teste, iniciou-se simulações com uma cardinalidade de busca baixa, $|N(s)_{open}|_{baixa}$, elevando-se gradativamente até se atingir resultados satisfatórios. Obteve-se resultados satisfatórios com $|N(s)_{open}|=50$. Função de Aspiração ($A(f(s_m))$) - priorizou-se evitar a formação de caminhos cíclicos, mas com alguma aceitação de movimentos $m$, $s \leftarrow s_m \circ m$, $s \in \mathrm{LT} \Rightarrow f(s) \leqslant A(f(s_m))$. Experimentalmente obteve-se a Função de Aptidão $A(f(s_m)) = f(s_m) - 1$. O número máximo de iterações (*TSMAX*), considerou-se $TSMAX=50$.

### 6.2.4. Parâmetros de Desempenho

Para a medição de desempenho, foram analisados os seguintes critérios: (i) número de reafiliações [CHA02], (ii) fluxo de dados *inter-cluster* $\psi_{inter}$ *e intra-cluster* $\psi_{intra}$ , (iii) número de atualizações do *dominant set*, *DS(G),* (iv) Função Objetivo, $F_o$ e (v) Tempo de execução $T_{exec}$ .

Uma reafiliação ocorre quando um *clusternode* se move e abandona o *cluster* atual e se torna membro de outro *cluster*. O fluxo de dados *inter-cluster* ocorre entre *clusterheads*, graças à capacidade de transmissão *dual* dos nós, em baixa energia dentro do *cluster* e em alta entre *clusters*. Essa é uma das metas do algoritmo proposto, isto é, minimizar a comunicação *inter-cluster*. $\psi$ é a diferença entre o fluxo de comunicação *intra* e *inter-cluster*, isto é, $\psi$ . Uma nova execução do algoritmo gerador da topologia provoca atualização do *dominant set*. Isto acontece quando: (i) um nó abandona o seu *cluster* atual e não consegue reafiliar-se a outro (falta de adjacência), (ii) ocorre falha de um *clusterhead*. (iii) $\sum_{k=0}^{k-1}(\sum_{i=0}^{node_k}\psi(i)_{intra}(t-\Delta t)-\sum_{i=0}^{node_k}\psi(i)_{inter}(t-\Delta t))<0$ , onde i é o i-ésimo nó do k-ésimo *cluster* . O valor de $\Delta t$ é ótimo quando $\Delta t \to 0$ . Considerar-se-á $\Delta t = 1$ , para tempo de simulação discreto. Esta é uma abordagem mais geral. O valor minimizado da Função Objetivo $F_{min}$ é o resultado obtido por cada modelo após uma execução e pode ser utilizada para medir a qualidade da técnica. O Tempo de Execução $T_{exec}$ , por sua vez, mede o tempo necessário para se obtê-la, dentro de vários cenários de simulação, desde uma topologia mais simples até uma rede mais complexa. A complexidade aqui é traduzida pelo número de nós da rede (*N*). Considerou-se experimentalmente *N=20*, disposta de forma aleatória uniformemente distribuída. Incrementou-se o número de nós em dez unidades até um valor máximo viável computacionalmente para todos os experimentos apresentados e dentro dos recursos disponíveis. Obteve-se então *N=50*.

## 6.3. TEMPERATURA INICIAL EXPERIMENTAL NO SA

Experimentos foram realizados para a obtenção de uma Temperatura Inicial

($T_0$) que permita aquecimento suficiente do sistema e o seqüente resfriamento para melhora da têmpera do material por ocasião do processo de diminuição da entropia entre as moléculas. Não deve estar pouco aquecido a ponto de não permitir agitação suficiente das moléculas, particularmente, no caso do SA, é traduzido pela possibilidade de aceitação de soluções vizinhas. Por outro lado, não deve estar muito aquecido a ponto de dificultar a convergência para estados estáveis de baixa energia. O emprego de $T_0$ aleatório apresentada o risco de se ter uma temperatura inadequada, por estar muito acima ou abaixo do desejável. Obter de forma controlada o valor de $T_0$ torna-se então uma etapa vantajosa, mesmo que haja algum custo computacional adicional. Para a obtenção de $T_0$, inicia-se com $T_{baixa}$=100. Em cada ciclo, verifica-se o grau de aceitação de vizinhos $s$ de $s_0$. Considerou-se que o processo termina quando atinge $p_{aceitacao}(T_0)=95\%$.

Um exemplo de obtenção de $T_0$ experimental é mostrado na Figura 56 e outra na Figura 57.

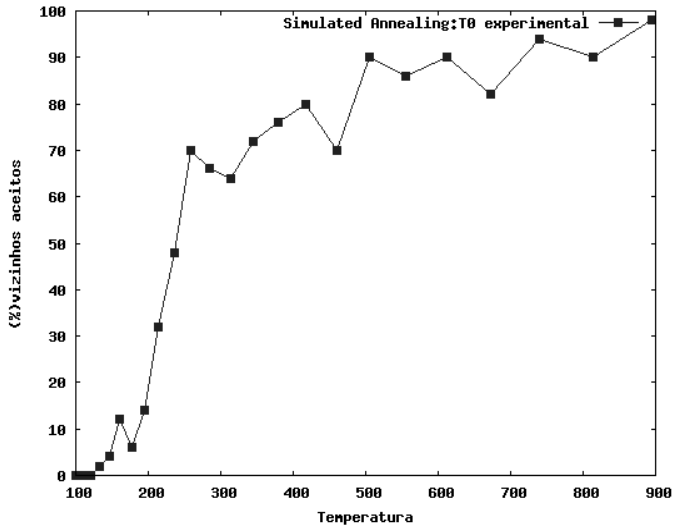


*Figura 56: Obtendo Temperatura Inicial Experimental, com T0=900, para p(aceitacao)>95%*

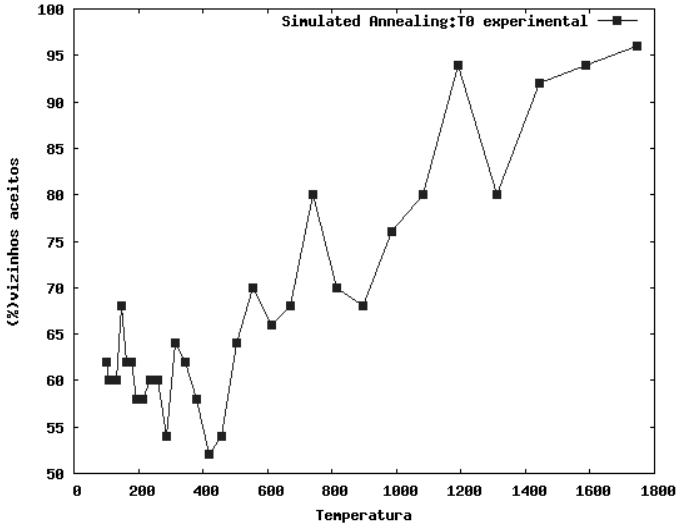

*Figura 57: Obtendo Temperatura Inicial Experimental, com T0=1800, para p(aceitacao)>95%*

Observamos na Figura 56 e na Figura 57 estados irregulares de aumento e diminuição do percentual de aceitação à medida que a temperatura aumenta. Isto ocorre porque $p_{aceitacao}(s_0)$ depende do arranjo de moléculas do sistema em cada estado, podendo existir arranjos mais, menos propícios à mudanças de estado, ou mais instáveis, mais sujeitos à mudanças.

Observa-se também que houve resultados distintos para $T_0$, mesmo com as duas simulações sendo realizadas sobre um mesmo cenário, com a mesma quantidade e disposição da rede. Isto acontece, mesmo para duas execuções distintas de um mesmo cenário porque o processo de obtenção de $T_0$ sempre parte de uma solução inicial aleatória $s_0$. Considere a solução inicial $s_0$ para o experimento da Figura 56 e $s_0'$ para a Figura 57. Considere também a sucessão de soluções aceitas em cada experimento $s_0, s_0(1), s_0(2), ..., s_0(max-1)$ e $s_0', s_0'(1), s_0'(2), ..., s_0'(max'-1)$, com onde *max-1* e *max'-1* é a última iteração de cada SA até atingir $T_0$. A solução $s_0(1)$ é a próxima solução aceita no primeiro experimento após $s_0$, onde $s_0(1) \in N(s_0)_{open}$, independente de critério $f(s_0(1)) \leqslant f(s_0) \| r \leqslant e^{-\Delta(f(s_0))/T}$, com $r=[0,1]$, e $s'_0(1)$ é a próxima solução aceita no segundo experimento após $s'_0(1)$, onde $s_0'(1) \in N(s_0')_{open}$, independente de critério $f(s_0'(1)) \leqslant f(s_0') \| r' \leqslant e^{-\Delta(f(s_0'))/T}$, com $r'=[0,1]$. Como o domínio de soluções é complexo, acreditamos ser muito pouco provável a obtenção de $s_0$ e $s_0'$, com $s_0 = s_0'$, muito menos a permanência de igualdade na seqüência $s_0, s_0(1), s_0(2), ..., s_0(max-1)$ e

$s_0^{'}, s_0^{'}(1), s_0^{'}(2), ..., s_0^{'}(max^{'}-1)$ de cada execução, mesmo para execuções distintas de um mesmo cenário.

Conclui-se que a definição de um sistema suficientemente aquecido depende das condições do próprio sistema. Para cada caso, podermos ter um valor diferente para $T_0$. Em alguns casos, podemos afirmar que $T_0$ pode sofrer variações em execuções diferentes com mesmo cenário. A Figura 56 e a Figura 57 mostram esta situação. Nos dois experimentos tem-se o mesmo cenário. Para a Figura 56, o sistema é considerado aquecido, com $p(aceitacao) \geqslant 95\%$ para $T_0=900$. Por outro lado, na Figura 57 o mesmo sistema é considerado aquecido com $T_0=1800$.

A grande vantagem deste processo é a obtenção de uma temperatura inicial $T_0$ ideal personalizada para cada problema. Isto evita a aleatoriedade de $T_0$ passível de ser mais baixa ou mais alta que o necessário. Uma desvantagem é o custo computacional acrescido a $T_{exec.}$

## 6.4. CICLO DE AQUECIMENTO E RESFRIAMENTO NO SA

Experimentos foram realizados sobre o comportamento do ciclo de aquecimento e resfriamento do SA para observar que a obtenção de $T_0$ em cada execução apresenta resultados satisfatórios. O aquecimento ocorre na fase inicial do sistema. O objetivo é obter $T_0$ que proporcione um alto grau de entropia no sistema e torná-lo "quente" o suficiente para facilitar mudanças de estado. No caso do SA, mudança de estado significa a possibilidade de aceitação de soluções vizinhas $s$ de $s_0$, onde $s \in N(s_0)_{open}$, chamado de $p_{aceitacao}$. O processo de aquecimento é iniciado a partir de uma temperatura considerada baixa para a média de experimentos realizados $T_{baixa}$. Uma solução $s_0$ é selecionada aleatoriamente. Aquece-se gradativamente o sistema sob ação do coeficiente de aquecimento $\lambda$ e verifica-se $p_{aceitacao}(s_0)$, em cada $T_{i+1} = \lambda T_i, i = [0, max-1]$, onde *max-1* é a iteração onde $T_{max-1} = T_0$. O processo termina quando atingimos um estado de alta energia, isto é, se $T_{atual}$ permitir um grande percentual aceitação de vizinhos $s$, isto é, $p_{aceitacao}(s_0) \geqslant p_{aceitacao}$. Considerou-se o valor experimental de $p_{aceitacao} = 95\%$. O sistema é considerado aquecido e $T_{atual}$ torna-se $T_0$ e o sistema está pronto para o processo de têmpera do material. O resfriamento é iniciado a partir de $T_0$. A razão de resfriamento

$\alpha$ atua em cada laço diminuindo a temperatura do sistema, com $T_{i+1}=\alpha T_i, i=[0,max-1]$ . Constitui uma das condições de parada do SA, $T_{atual}=T_c$ , juntamente com $iter \leqslant SAMAX$ e $iter \leqslant \delta$ .

Um exemplo de ciclo de aquecimento/resfriamento é mostrado na Figura 58 e outro na Figura 59. É importante observar que o aquecimento de cada ciclo corresponde ao processo de obtenção de $T_0$ descrito na Figura 56 e na Figura 57, respectivamente.

Observamos durante a fase de aquecimento das simulações que $T_{i+1} \geqslant T_i$ , $\forall i, i \leqslant k, T_k = T_0$ . Isto acontece porque durante a obtenção de $T_0$ experimental, que o sistema está em aquecimento gradativo sob ação de $\lambda$ , com $T=T(\lambda)$ . A partir de uma temperatura considerada baixa $T_{baixa}=100$ o sistema é aquecido, independentemente das oscilações de $p_{aceitacao}(s_0)$ durante o processo, enquanto não atinge $p_{aceitacao}(s_0) \leqslant p_{aceitacao}$ , onde $s_0, s_0(1), s_0(2), ..., s_0(max-1)$ é a sucessão de soluções aceitas de cada SA e *max-1* é a última iteração até atingir $T_0$.

Observamos durante a fase de resfriamento das simulações que $T_{i+1} \leqslant T_i$ , $\forall i, i \geqslant k, T_k = T_0$ . Isto acontece porque a partir dos respectivos T0, inicia-se o processo de têmpera do material, por ação do coeficiente de resfriamento $\alpha$ , com $T=T(\alpha)$ . A entropia do sistema é gradativamente diminuída e constitui-se em uma das condições de parada do SA, quando $T_{atual}=T_c$ .

Observamos também para as duas simulações o número de iterações, *iter*, *iter* e *iter'* e a temperatura final $T_{final}$ e $T'_{final}$ .Verifica-se que $iter \neq iter'$ e $T_{final} \neq T'_{final}$ , mesmo com a igualdade de cenário de simulação para execuções distintas. Isto ocorre porque, mesmo para duas execuções distintas de um mesmo cenário, o SA sempre parte de uma solução inicial aleatória $s_0$. Considere a solução inicial $s_0$ para o experimento da Figura 58 e $s_0'$ para a Figura 59. Considere também a sucessão de soluções aceitas em cada experimento após $T_0$, $s_0, s_0(1), s_0(2), ..., s_0(max-1)$ para o primeiro experimento e $s_0', s_0'(1), s_0'(2), ..., s_0'(max'-1)$ , para o segundo, onde *max-1* e *max'-1* é a última iteração de cada SA até atingir a condição de parada. A solução $s_0(1)$ é a próxima solução aceita no primeiro experimento após $s_0$, onde $s_0(1) \in N(s_0)_{open}$ , independente de critério $f(s_0(1)) \leqslant f(s_0) || r \leqslant e^{-\Delta(f(s_0))/T}$ , com $r=[0,1]$ , e $s'_0(1)$ é a próxima solução aceita no

segundo experimento após *s'0(1)*, onde $s_0^{'}(1) \in N(s_0^{'})_{open}$ , independente de critério $f(s_0^{'}(1)) \leqslant f(s_0^{'}) \| r^{'} \leqslant e^{-\Delta(f(s_0^{'}))/T}$ , com $r^{'} = [0,1]$ . Como o domínio de soluções é complexo, acreditamos ser muito pouco provável a igualdade da seqüência $s_0, s_0(1), s_0(2), ..., s_0(max-1)$ e $s_0^{'}, s_0^{'}(1), s_0^{'}(2), ..., s_0^{'}(max^{'}-1)$ de cada execução do SA, mesmo supondo que $T_0 = T'_0$ e cenários de simulação iguais para execuções distintas do SA. Logo, nestas condições, não podemos prever que $iter = iter'$ , muito menos que $T_{final} = T'_{final}$ . No presente caso, temos $iter = 75$ , $T_{final} = 900$ , $iter' = 82$ , $T'_{final} = 1800$ .

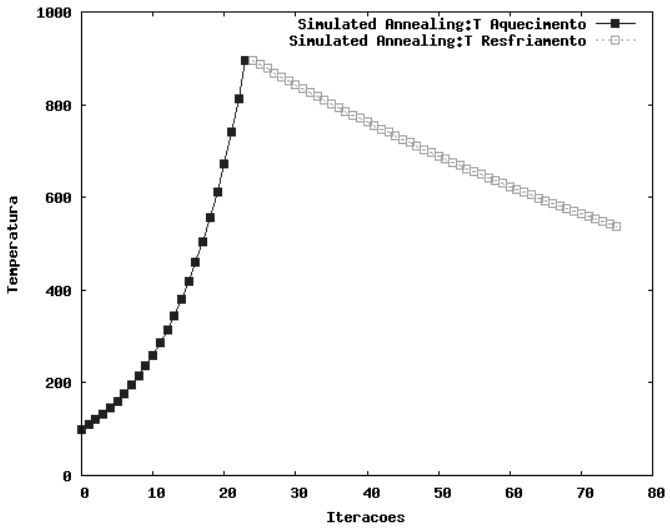


*Figura 58: Um Ciclo de Aquecimento e Resfriamento do SA, para T0=900*

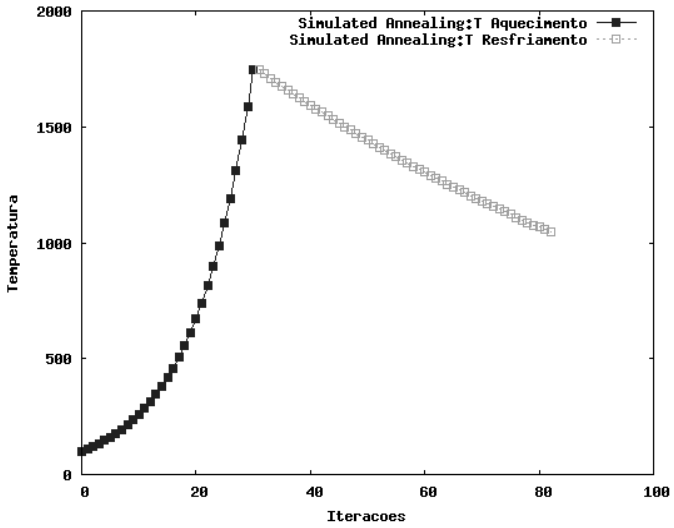


*Figura 59: Um Ciclo de Aquecimento e Resfriamento do SA, para T0=1800*

## 6.5. ANÁLISE DE DESEMPENHO EM RELAÇÃO À FUNÇÃO OBJETIVO

Experimentos foram realizados para verificar qual MH do presente estudo apresenta melhores resultados em relação à Função Objetivo ($F_0$):*GAdHoc* [GAR05], *SAdHoc* e *TSAdHoc*. Verificou-se também o desempenho dessas técnicas com outras técnicas já existentes: *Weighted Clustering Algorithm* (WCA) [CHA00] e *Highest-Degree Algorithm*(HDA) [GER95]. O objetivo dos modelos propostos é realizar o particionamento da rede em *clusters* e determinar *clusterheads*, dados os *clusters* já formados em um instante de tempo *t*. Para efeito de simplificação de análise de resultados considere o conjunto {*GAdHoc*, *SAdHoc* e *TSAdHoc}* chamado de *MHAdHoc*.

Para as simulações, utilizou-se como parâmetro de desempenho o resultado final de cada técnica, isto é, o valor da Função Objetivo $F_o$ alcançado, para medir a qualidade da técnica, desde uma topologia mais simples até uma rede mais complexa. A complexidade aqui é traduzida pelo número de nós da rede (*N*). Considerou-se experimentalmente *N=20*, disposta de forma aleatória uniformemente distribuída. Incrementou-se o número de nós em dez unidades até um valor máximo viável computacionalmente para todos os experimentos apresentados. Obteve-se *N=50*.

Para o tratamento estatístico de dados, foi calculada a Média $\mu$ , Desvio Padrão $\sigma$ , Coeficiente de Variação (*CV*) para cada informação apresentada, a partir de amostragem de experimentos de cada cenário de simulação, *N=[20..50],* aplicada em cada técnica, com $CV \leqslant 5\%$ . O Intervalo de Confiança é de 95%, para o intervalo $[\bar{X} - SE\,,\, \bar{X} + SE]$ ,com $\bar{X} = \mu$ , se a amostra for suficientemente grande, $n \geqslant 30$ , e $SE = 1.96 * \sigma / \sqrt{n}$ , onde SE é o Erro Padrão da Média [MON94], [TRI01].

A Figura 60 mostra o desempenho segundo o resultado da Função Objetivo ($F_0$) para diferentes topologias, com *N=[20,50]*, envolvendo as técnicas propostas *GAdHoc, SAdHoc* e *TSAdHoc*, bem como as técnicas já existentes WCA e HDA. Quanto menor o valor de $F_0$ melhor será o resultado obtido.

Analisamos inicialmente os algoritmos propostos *GAdHoc, SAdHoc e TSAdHoc* em relação aos modelos WCA e HDA. Observamos que todos modelos propostos apresentam menores valores para $F_0$ em relação aos modelos já existentes em

todos os cenários de simulação. Isto ocorre devido aos melhoramentos implementados nos modelos propostos, com base na minimização de $\psi_{inter}$ . Em conseqüência, temos resultados positivos para os algoritmos em estudo.

Observamos, agora, entre os algoritmos propostos que os modelos *SAdHoc* e *TSAdHoc* apresentam menores valores de $F_0$ em relação ao *GAdHoc*. Esta diferença, de modo geral, se acentua à medida que a topologia aumenta. Isto ocorre possivelmente devido a ajustes particulares ao SA e TS no processo de seleção de um vizinho $s$ de $s_0$, durante a execução de cada algoritmo. Este melhoramento também desenvolvido no presente estudo e será analisado ainda neste capítulo em um tópico à parte. Isto poderia proporcionar uma certa vantagem na convergência para soluções quase ótimas em cada execução, com melhora de desempenho como podemos constatar. Por sua vez, este método não foi implementado no GA, acoplado ao *GAdHoc*, por se tratar de um algoritmo tipicamente populacional. A busca para uma solução final se dá essencialmente pela troca de material genético entre cromossomos de $P_i$ para uma nova geração $P_{i+1}$ , onde $i=[0, max-2]$ , para *max-1* como sendo a última geração do GA e não por seleção de um vizinho $s$ de $s_0$, como basicamente ocorre no SA e no TS, acoplados nos modelos propostos *SAdHoc* e *TSAdHoc*, cada qual dentro do seu método.

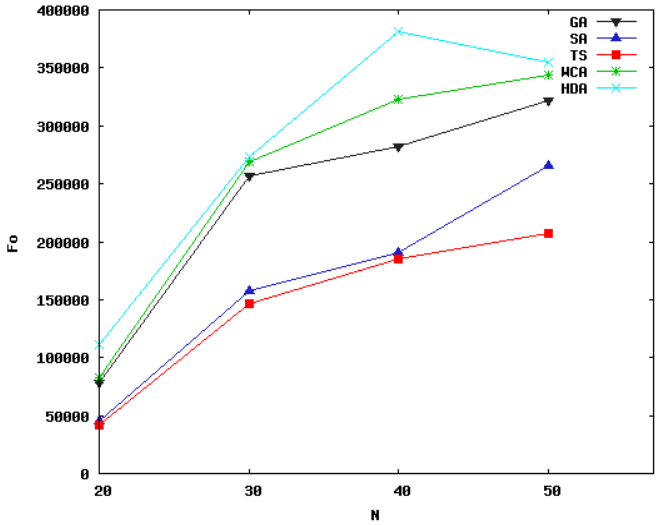


*Figura 60: Análise de Desempenho segundo o resultado da Função Objetivo para diferentes topologias*

Concluímos que os modelos propostos apresentam melhor desempenho em relação a alguns modelos já existentes:

- O *TSAdHoc* e o *SAdHoc* apresentaram melhores desempenhos em relação aos

demais, com uma ligeira vantagem do *TSAdHoc*. Por ser um *NBA*, tal como o *VNS*, eles não necessitam gerar uma população de soluções, agregada às condições restritivas de validade de cada cromossomo em cada ciclo de execução, com possível perda de desempenho, como ocorre no *GAdHoc*. O *TSAdHoc* apresenta uma ligeira vantagem sobre o *SAdHoc*, provavelmente por não possuir uma fase preliminar para estabelecer condições iniciais de execução como ocorre no *SAdHoc*. Além disso, a implementação do *SLEP* no processo de seleção de um vizinho *s* de $s_0$ no *TSAdHoc* contribui para os bons resultados. Uma discussão sobre o *SLEP* como método de seleção de vizinho *s* de $s_0$ será apresentado em um tópico à parte pela sua importância.

## 6.6. ANÁLISE DE DESEMPENHO EM RELAÇÃO AO TEMPO DE EXECUÇÃO

Experimentos foram realizados para verificar qual MH do presente estudo apresenta melhores resultados em relação ao Tempo de Execução($T_{exec}$):*GAdHoc* [GAR05], *SAdHoc* e *TSAdHoc*. Verificou-se também o desempenho dessas técnicas com outras técnicas já existentes:*Weighted Clustering Algorithm* (WCA) [CHA00] e *Highest-Degree Algorithm* (HDA) [GER95]

Para as simulações, utilizou-se como parâmetro de desempenho o tempo necessário para se obter $F_0$, isto é, o tempo de execução entre diferentes cenários de simulação, desde uma topologia mais simples até uma rede mais complexa. A complexidade aqui é traduzida pelo número de nós da rede(*N*). Considerou-se experimentalmente *N=20*, disposta de forma aleatória uniformemente distribuída. Incrementou-se o número de nós em dez unidades até um valor máximo viável computacionalmente para todos os experimentos apresentados. Obteve-se *N=50*.

Para o tratamento estatístico de dados, foi calculada a Média $\mu$ , Desvio Padrão $\sigma$ , Coeficiente de Variação (*CV*) para cada informação apresentada, a partir de amostragem de experimentos de cada cenário de simulação, *N=[20..50],* aplicada em cada técnica, com $CV \leqslant 5\%$ . O Intervalo de Confiança é de 95%, para o intervalo $[\bar{X} - SE, \bar{X} + SE]$ ,com $\bar{X} = \mu$ , se a amostra for suficientemente grande, $n \geqslant 30$ , e $SE = 1.96 * \sigma / \sqrt{n}$ , onde SE é o Erro Padrão da Média [MON94], [TRI01].

A Figura 61 e a Figura 62 mostram o desempenho segundo o Tempo de Execução ($T_{exec}$) para diferentes topologias, com *N=[20,50]*, envolvendo as técnicas propostas *GAdHoc, SAdHoc* e *TSAdHoc*, bem como as técnicas já existentes WCA e HDA (exceto *GAdHoc* na Figura 62).

Analisamos os algoritmos propostos *SAdHoc e TSAdHoc* e os modelos WCA e HDA na Figura 62. O *GAdHoc* é visto somente na Figura 61, por apresentar ordem de grandeza de resultados diferente em relação aos demais. Na Figura 62, observamos que $T_{exec}$ nos modelos *TSAdHoc*, WCA e HDA não sofrem alterações significativas ao longo dos diferentes cenários de simulação. Por outro lado, o *SAdHoc* sofre uma sensível piora no tempo de execução à medida que a topologia aumenta, mas dentro da mesma ordem de grandeza dos demais anteriormente citados, o que não acontece com o *GAdHoc*.

Concluímos que os modelos propostos apresentam comportamentos bem distintos à medida que o número de nós da rede aumenta:

- O *TSAdHoc* apresenta maior robustez, não sofrendo grandes mudanças ao longo dos diferentes cenários de simulação. O desempenho um pouco inferior em relação aos modelos existentes, é dado pela implementação mais complexa. Mesmo que $T_{exec}(TSAdHoc) > T_{exec}(WCA, HDA)$ ,compensa-se pela obtenção de melhores resultados em $F_0(TSAdHoc) \ll F_0(WCA, HDA)$ , $F_0(TSAdHoc) \ll F_0(SAdHoc, GAdHoc)$ , $\forall N \in \{20,...,50\}$
- O *SAdHoc*, por sua vez, apresenta $T_{exec}$ mais suscetível a mudanças de topologia. Temos que $T_{exec}(N=20) < ... < T_{exec}(N=50)$ . Isto ocorre porque na implementação do SA, acoplado ao *SAdHoc*, optou-se por obter experimentalmente a Temperatura Inicial ($T_0$) e não de forma simplesmente aleatória. Como o valor ideal de $T_0$ depende de cada problema, selecionar aleatoriamente $T_0$ há a possibilidade de surgimento de condições não-ideais de aquecimento do sistema, seja para o excesso ou para a falta. Durante esta fase, temos a verificação de $p_{aceitacao}(s_0)$ , como referência de aquecimento inicial ideal. Quanto maior o número de nós, este processo consome mais tempo de processamento. Em suma, trata-se de uma troca, visando maximizar melhores resultados de $F_0$ para o *SAdHoc*, sem que haja grandes comprometimentos de $T_{exec}$.

– Por fim, o *GAdHoc* é o algoritmo que apresentou piores resultados em $T_{exec}$. Isto ocorre porque tem-se a necessidade de geração populacional, em geral com um número significativo de cromossomos para maximizar a diversidade de indivíduos, dentro de um domínio de soluções. Mas isto tem um preço, todos os indivíduos passam por verificações de validade para não se tornarem soluções impossíveis ou não realísticas, acaba consumindo grande para do tempo total de execução do algoritmo. Isto ocorre também nas demais MH propostas, mas por se tratarem de algoritmos não-populacionais, esta verificação se restringe à ordem de grandeza de indivíduos e não de populações, como ocorre no GA. Em conseqüência, colabora também para o baixo desempenho, a aplicação destes mesmos mecanismos de validação para as gerações seguintes. Mesmo que possam herdar melhor aptidão para sobreviver em relação aos seus cromossomos pais, verificou-se experimentalmente que alguns cromossomos filhos podem não corresponder à soluções realísticas da rede. A melhoria no processo de *crossover*, onde temos a escolha de um ponto de corte aleatório distribuído uniformemente, como referência de divisão de material genético à próxima geração de cada casal de indivíduos que passam pelo processo de *crossover*, pode ser proposta como trabalhos futuros, numa possível tentativa de melhora de desempenho do GA acoplado ao *GAdHoc*.

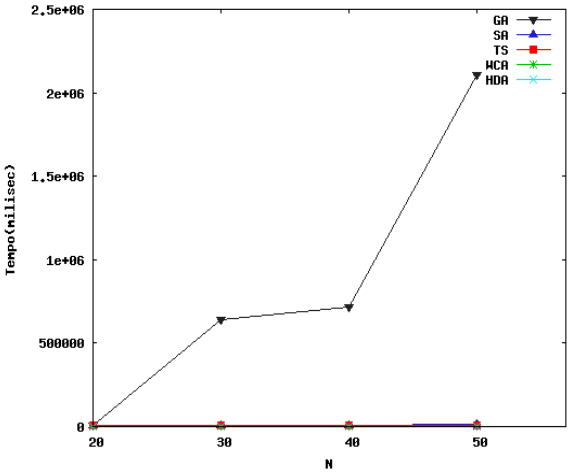


*Figura 61: Análise de Desempenho segundo o Tempo de Execução (Texec) para diferentes topologias*

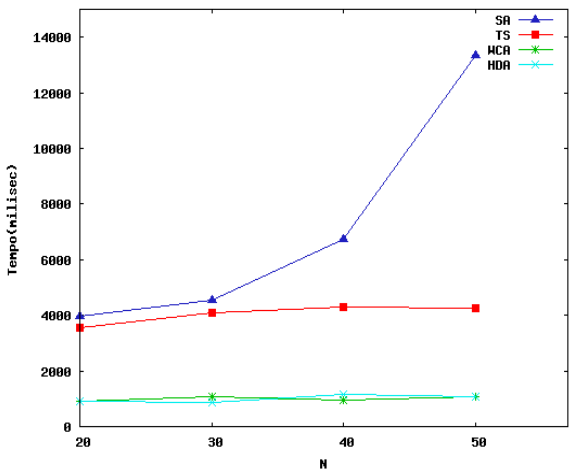


*Figura 62: Análise de Desempenho segundo o Tempo de Execução (Texec) para diferentes topologias*

## 6.7. ANÁLISE DE DESEMPENHO EM RELAÇÃO AO FLUXO INTRA/INTER CLUSTERS

Experimentos foram realizados para verificar o parâmetro de desempenho fluxo *intra-cluster,* $\psi_{intra}$ , e fluxo *inter-cluster*, $\psi_{inter}$ , entre o modelo proposto, *MHAdHoc,* e outras técnicas já existentes: *Weighted Clustering Algorithm* (WCA) [CHA00] e *Highest-Degree Algorithm* (HDA) [GER95]. Para efeito de simplificação na coleta e análise de dados, o *MHAdHoc* foi empregada apenas com a metaheurística Busca Tabu.

Para as simulações, utilizou-se como parâmetro de desempenho a relação entre $\psi_{intra}$ e $\psi_{inter}$ observado em cada técnica em cenários de diferentes complexidades em relação a, $T_{exec}$ e em cada intervalo $T_g$. $T_{exec}$ é o tempo total de simulação. $T_g$ é a granulosidade, isto é, o intervalo de tempo entre cada observação do comportamento da rede.

A duração de cada $T_{exec}$ deve ser suficientemente grande para que se possa observar o comportamento da rede, movimentações, rearranjos de topologia, consumo de energia, *handoffs inter-cluster* por perda de potência em relação ao *clusterhead*, *handoffs* por diferença de potência entre *clusterheads* interno e externo, possibilidade de falha de nós da rede, consumo de energia, capacidade de transmissão e disponibilidade dos nós da topologia, tudo com o objetivo de tornar o ambiente de simulação mais realístico. Obteve-se $T_{exec}=20$, para todas as simulações. A granulosidade observada é de $T_g=1$. Em cada intervalo de $T_g$, foi registrado o valor do parâmetro em observação e fatores relevantes para a análise. A complexidade dos cenários varia conforme o número de nós da rede ($N$). Considerou-se experimentalmente $N=20$, disposta de forma aleatória uniformemente distribuída. Incrementou-se o número de nós em dez unidades até um valor máximo viável computacionalmente para todas as simulações. Obteve-se experimentalmente $N=50$.

As ilustrações 63, 64 e 65 mostram o comportamento de $\psi_{intra}$ e $\psi_{inter}$ em execuções distintas das técnicas MHAdHoc, WCA e HDA , respectivamente ao longo de $T_{exec}$ com granulosidade $T_g$ em um cenário com N=20, com os nós distribuídos uniformemente de forma aleatória. A diferença de fluxo $\psi=\psi_{intra}-\psi_{inter}$ de cada técnica

ao longo do tempo de execução é mostrada na Figura 66.

Na Figura 63 observa-se que as maiores diferenças entre $\psi_{intra}(MHAdHoc)$ e $\psi_{inter}(MHAdHoc)$ ocorrem geralmente logo após os instantes em que $\psi < 0$ . Isto acontece devido aos eventos dinâmicos da rede ao longo da simulação. Por exemplo, nós da rede se movimentam provocam diferenças na topologia. *Clusternodes* podem sair da área de seus *clusters* e provocarem alterações em $\psi$ . Outra possibilidade é a ocorrência de falha em um *clusterhead*, provocando novo particionamento em *clusters*. Além disso, é possível que tenha sido causado, simplesmente, por um pico de demanda de comunicação *inter-cluster*. Nestes casos, há uma nova execução do algoritmo, como ocorre em t=3 e t=12. A diferença abrupta entre a comunicação *intra* e *inter-cluster*, após estes instantes, ocorre porque, a cada novo arranjo da topologia, uma nova execução do algoritmo é realizada. O pico presente no início da simulação ocorre devido à execução do algoritmo para a formação do cenário inicial.

A Figura 64 mostra resultados para o modelo WCA, quanto ao fluxo de comunicação *intra* e *inter-cluster* $\psi_{intra}(WCA)$ e $\psi_{inter}(WCA)$ . Observa-se a ausência na correlação na diferença de fluxo *intra-cluster e inter-cluster,* $\psi$ , com $\psi_{intra} < \psi_{inter}$ nos tempos *t=3,5,10,14*. Isto acontece porque, este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, $W_v = W_v(\Delta_v, D_v, P_v, M_v)$ . A determinação de $W_v$ utiliza *weight factors* como $\Delta_v$ e $D_v$ . Por isso, é possível que oscilações negativas no valor de $\psi$ sejam causadas por atualizações do *DS(G)*, motivadas por modificações de $\Delta_v$ ou $D_v$ .

A Figura 65 mostra resultados para o modelo HDA, quanto ao fluxo de comunicação *intra* e *inter-cluster* $\psi_{intra}(HDA)$ e $\psi_{inter}(HDA)$ . Observa-se a ausência na correlação na diferença de fluxo *intra-cluster e inter-cluster,* $\psi$ , com prevalência de $\psi_{intra} < \psi_{inter}$ nos intervalos *t=[4,11]* e *t=[13,20]*. Isto acontece porque, este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, o grau de cada nó. A formação de *clusters* é feita como base no *degree* para formar o *DS(G)*. Este modelo não apresenta limite de número de nós em um mesmo *cluster*, apresentando maior suscetibilidade à exaustão dos seus *clusterheads*. Também é mais suscetível a sofrer desequilíbrios na relação $\psi_{intra} - \psi_{inter}$ na ocorrência de eventos dinâmicos da rede. Por

exemplo, inicialmente temos uma topologia formada com $\psi_{t=1} >>> 0$ . Esta diferença sofre decréscimo nos dois intervalos seguintes, $\Delta_{\psi}(t=[2,3]) < 0$ , ainda com $\psi_{t=[2;3]} > 0$ , mas quando *t=4*, $\psi_{t=4} < 0$ , sem que haja tratamento. Esta diferença em $-\psi$ é mantida ao longo de boa parte da simulação, até *t=12* , quando então $\psi_{t=12} > 0$ , no entanto com $\psi_{t>12} < 0$ .

A Figura 66 mostra a diferença $\psi = \psi_{intra} - \psi_{inter}$ entre os modelos MHAdHoc, WCA e HDA ao longo do tempo de simulação para os cenários das figuras 63, 64 e 65, onde N=20. Observa-se a prevalência de $+\psi(MHAdHoc)$ ao longo do tempo de execução, com exceções nos instantes de tempo *t=3* e *t=12*, onde $\psi_{intra}(MHAdHoc) < \psi_{inter}(MHAdHoc)$ . As maiores variações de $\Delta_{\psi(MHAdHoc)}(t+1)$ ,onde $\Delta_{\psi}(t+1) = \psi_{T=t+1} - \psi_{T=t}$ , $t = [0..T_{exec}]$ , ocorrem nos respectivos instantes seguintes, $\Delta_{\psi}(t=4;13)$ , por causa do rearranjo de topologia devido à diferença negativa de fluxo. Observa-se ainda o comportamento difuso de $\psi(WCA)$ e $\psi(HDA)$ , com predominância de $-\psi(WCA, HDA)$ em vários instantes e oscilações entre valores positivos e negativos, porque não leva em consideração a comunicação entre os nós da rede como fator de decisão. Acreditamos que haja ausência de algum mecanismo de controle, mesmo que em alguns instantes, como *t=2* e *t=10* tenha ocorrido resultados em $\psi_t(MHAdHoc) < \{\psi_t(HDA) \vee \psi_t(WCA)\}$ .

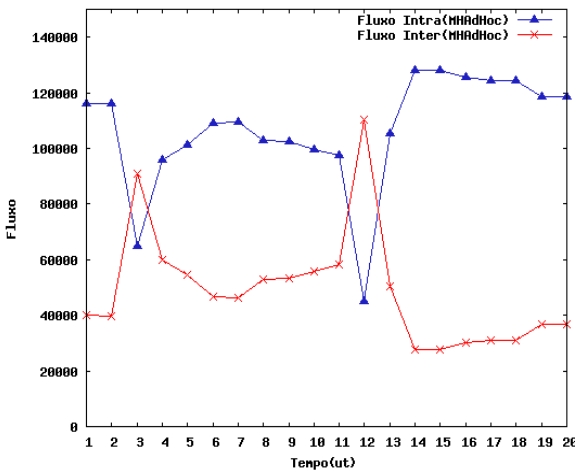


*Figura 63: Fluxo Intra/Inter-Cluster para uma execução do MHAdHoc com Cenário N=20*

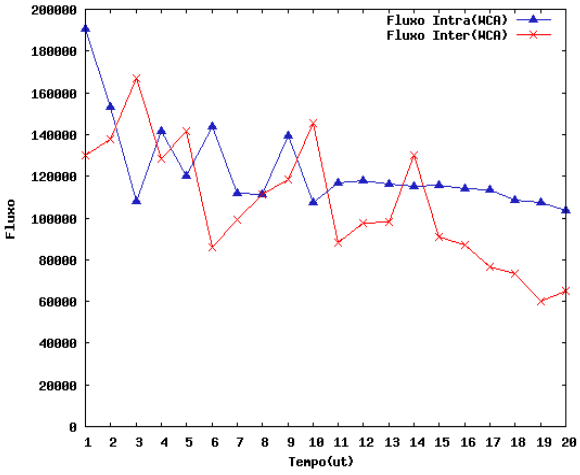


*Figura 64: Fluxo Intra/Inter-Cluster para uma execução do WCA com Cenário N=20*

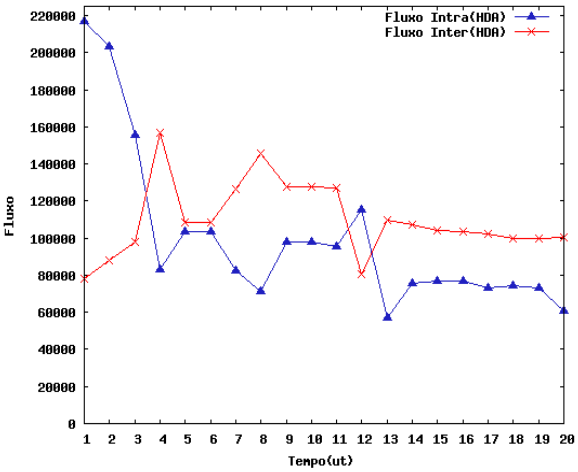


*Figura 65: Fluxo Intra/Inter-Cluster para uma execução do HDA com Cenário N=20*

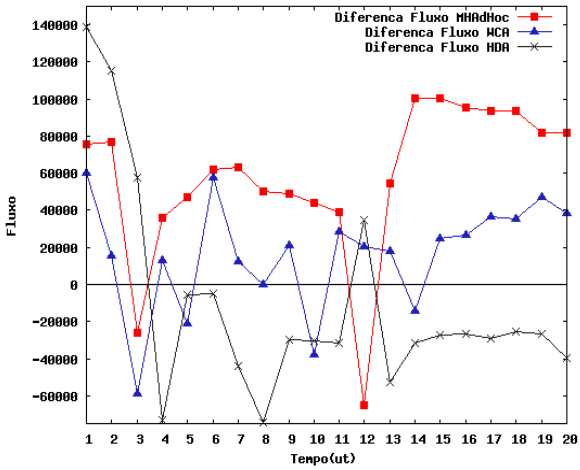


*Figura 66: Diferença de Fluxo Intra/Inter-Cluster para MHAdHoc,WCA e HDA com Cenário N=20*

As ilustrações 67, 68 e 70 mostram o comportamento de $\psi_{intra}$ e $\psi_{inter}$ para N=30 em execuções distintas das técnicas MHAdHoc, WCA e HDA , respectivamente ao longo de $T_{exec}$ com granulosidade $T_g$. Os nós foram dispostos aleatoriamente de forma uniformemente distribuída. A diferença de fluxo $\psi = \psi_{intra} - \psi_{inter}$ de cada técnica ao longo do tempo de execução é mostrada na Figura 70.

Na Figura 67 observa-se que $\psi_{intra}(MHAdHoc) > \psi_{inter}(MHAdHoc)$ ao longo de quase todo o tempo de simulação. Em alguns instantes temos aproximações e afastamentos entre $\psi_{intra}$ e $\psi_{inter}$ . As aproximações ocorrem em t=2, t=16 e t=10, com $-\Delta_{\psi(MHAdHoc)}$ ,para $\Delta_{\psi}(t=2)$ , $\Delta_{\psi}(t=16)$ e $\Delta_{\psi}(t=10)$ ,com $\Delta_{\psi} = \psi_t - \psi_{t-1}$ , este último com $\psi < 0$ , forçando rearranjo de topologia. Os correspondentes afastamentos ocorrem de três formas: imediatamente após uma aproximação, após uma aproximação, mas não imediatamente e obrigatoriamente após uma aproximação, se no instante anterior $\psi_{T=t-1} < 0$ , $t = [1..T_{exec}]$ .

No instante *t=3*, podemos observar um afastamento imediato, mas com $\psi_{T=t-1} > 0$ . Isto ocorre porque a aproximação em *t-1* foi provocado por eventos dinâmicos da rede. O mesmo acontece para o afastamento em *t*. A variação imediata é apenas uma casualidade, podendo ocorrer com algum retardo. Como exemplo ilustrativo, suponha que em *t-2* e *t-1* apenas um nó *v* sofre reafiliação, abandonando seu *cluster* atual i,

$C_i(v)$, passando para $C_j(v)$ e retornando para $C_i(v)$ em seguida. No instante seguinte, *t-1*, podemos ter uma variação negativa em $\Delta_\psi(t=2)=\psi_{t=2}-\psi_{t=1}$ , mas preservando $+\psi_{t=2}$ e no instante seguinte $+\Delta_\psi$ , $\Delta_\psi(t=3)=\psi_{t=3}-\psi_{t=2}$ , ainda com $+\psi_{t=3}$ .

No instante *t=11*, temos também um afastamento imediato, motivado pela diferença de fluxo negativa no instante anterior, $\psi_{T=t-1}<0$ . Isto ocorre motivado pelos eventos dinâmicos da rede. Houve a aproximação em *t-1*, onde podemos observar a variação negativa em $-\Delta_\psi(t-1)$ , $\Delta_\psi(t-1)=\psi_{t-1}-\psi_{t-2}$ , com aumento do fluxo *inter-cluster*. Isto irá provocar desequilíbrio na relação em $\psi_{t-1}$ , onde $\psi_{t-1}=\psi_{intra}(t-1)-\psi_{inter}(t-1)<0$ . Esta diferença negativa desencadeará o afastamento imediato, com rearranjo forçado da topologia no instante *t*. A grande variação de $\Delta_\psi(t)=\psi_t-\psi_{t-1}$ é causado pela ação do MHAdHoc. Uma nova topologia é formada segundo a estratégia de minimização de $\psi_{inter}$ .

No instante *t=20*, observamos um afastamento não imediato ocorrido em $\Delta_\psi(t=20)=\psi_t-\psi_{t-1}>0$ e nos instantes anteriores não há grandes variações na diferença de fluxo $\Delta_\psi(t-1)\approx\Delta_\psi(t-2)\approx\Delta_\psi(t-3)\approx cte$ . A aproximação é vista somente em *t-4*, onde $\Delta_\psi(t-4=16)=\psi_{t-4}-\psi_{t-5}<0$ . Isto ocorre porque a aproximação em *t-4*, provocado por eventos dinâmicos da rede, não foi suficiente para forçar um rearranjo da topologia, pois a relação de fluxo *intra/inter-cluster* se manteve positiva $\psi(t-4)>0$ , mesmo que a variação desta relação no tempo tenha se tornado negativa, $\Delta_\psi(t-4)<0$ . Ao longo dos próximos instantes de tempo, a diferença gerada em *t-4* não resultou em modificações para $\psi_{t'}$ ou para $\Delta_\psi(t')$ , no intervalo *t'={t-4,t-3,...,t-1}*, somente no instante de tempo *t*.

É importante considerar que o comportamento da rede ainda não é suficiente para justificar um novo rearranjo de topologia, mesmo que a diferença em $\psi$ tenha diminuído. A tolerância à algumas mudanças é justificada porque a reorganização também apresenta custo à rede, sugerindo a sua execução somente quando for realmente necessário. Logo, um bom método de particionamento em *clusters* deve ser robusto, isto é, que seja capaz de minimizar a necessidade de nova execução ao longo de $T_{exec}$ e ser tolerante aos eventos dinâmicos da rede. Espera-se que a robustez seja uma característica do MHAdHoc.

Suponha, como exemplo ilustrativo, que em *t-4* apenas um nó *v* sofreu reafiliação, abandonando seu *cluster* atual *i*, $C_i(v)$, passando para $C_j(v)$. Podemos ter uma variação negativa em $\Delta_\psi(t-3)=\psi_{t-3}-\psi_{t-4}$ , mas preservando $+\psi_{t-3}$ . Esta diferença em $\Delta_\psi$ é mantida no intervalo *t'={t-4,t-3,...,t-1}*, sem que haja rearranjo forçado de topologia pela diferença negativa de $\psi<0$ . Do exemplo, suponha que eventos dinâmicos da rede tenham ocorrido, mas impactos significativos em $\psi$ e $\Delta_\psi$ . Suponha que somente no instante *t*, o nó *v* tenha sofrido nova reafiliação, voltando para $C_i(v)$. Em conseqüência, temos o afastamento com $\Delta_\psi(t)>0$ .

No instante inicial de simulação, observamos um pico devido à execução do MHAdHoc. Uma nova topologia é formada segundo a estratégia de minimização de $\psi_{inter}$ .

A Figura 68 mostra resultados para o modelo WCA, com relação a $\psi_{intra}(WCA)$ e $\psi_{inter}(WCA)$ . Observa-se a ausência na correlação na diferença de fluxo *intra-cluster e inter-cluster,* $\psi$ , com prevalência de $\psi_{intra}<\psi_{inter}$ nos tempos *t=[5,12]* e *t=15*. Isto acontece porque este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, $W_v=W_v(\Delta_v, D_v, P_v, M_v)$ . A determinação de $W_v$ utiliza *weight factors* como $\Delta_v$ e $D_v$ . Por isso, é possível que oscilações negativas no valor de $\psi$ sejam causadas por atualizações do DS(G), motivadas por modificações de $D_v$ ou $\Delta_v$ .

A Figura 69 mostra resultados para o modelo HDA, com relação a $\psi_{intra}(HDA)$ e $\psi_{inter}(HDA)$ . Observa-se a ausência de correlação na diferença de fluxo *intra-cluster e inter-cluster,* $\psi$ , com prevalência de $\psi_{intra}<\psi_{inter}$ nos tempos *t=4*, *t=[6,12]* e *t=16*. Isto acontece porque, este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, o grau de cada nó. A formação de *clusters* é feita pela escolha de maiores graus de nós para formar o *DS(G)*.

A Figura 70 mostra $\psi=\psi_{intra}-\psi_{inter}$ entre os modelos MHAdHoc, WCA e HDA ao longo do tempo de simulação para os cenários das figuras 67, 68 e 69, onde N=30. Observa-se ao longo do tempo de simulação que as maiores diferenças de $+\psi$ são da técnica MHAdHoc, à exceção de *t=2* e *t=10*. Em *t=10,* temos $\Delta_{\psi(MHAdHoc)}(t=10)<0$ .

Assim, há um rearranjo de topologia, $\Delta_{\psi(MHAdHoc)}(t+1)>0$ . Em $t=2$, temos $\psi(HDA)>\psi(MHAdHoc),\psi(WCA)$ , mesmo com $+\psi$ para o modelo proposto. Esta diferença é circunstancial, visto nos instantes seguintes, temos a prevalência de valores negativos de $\psi(WCA)$ e $\psi(HDA)$ . Observa-se ainda o comportamento difuso de $\Delta\psi_{(WCA)}$ e $\Delta\psi_{(HDA)}$ , com predominância de $-\Delta\psi_{(WCA,HDA)}$ em vários instantes, por não considerar a comunicação entre os nós da rede como fator de decisão. Acreditamos que haja ausência de algum mecanismo de controle, mesmo com a ocorrência de $\psi_{t_1=2,t_2=10}(MHAdHoc)<\{\psi_{t_1=2,t_2=10}(HDA)\vee\psi_{t_1=2,t_2=10}(WCA)\}$ .

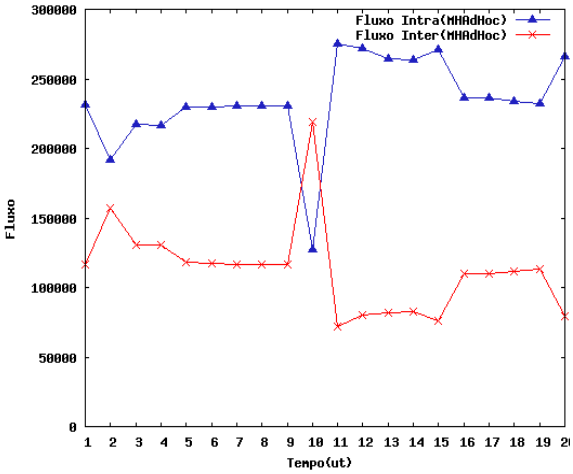


*Figura 67: Fluxo Intra/Inter-Cluster para uma execução do MHAdHoc com Cenário N=30*

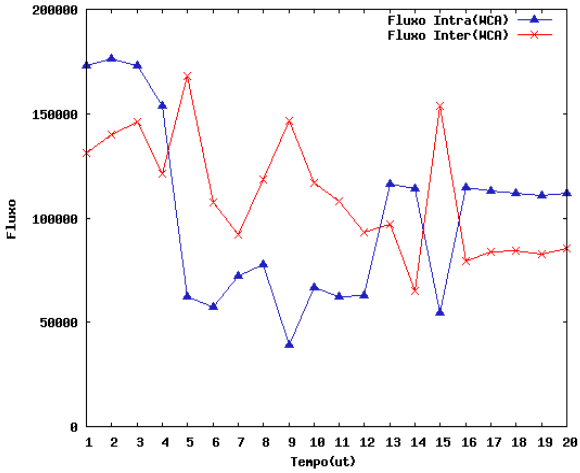


*Figura 68: Fluxo Intra/Inter-Cluster para uma execução do WCA com Cenário N=30*

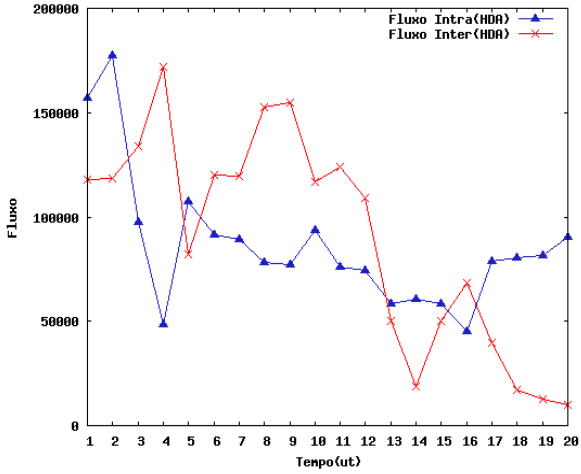


*Figura 70: Fluxo Intra/Inter-Cluster para uma execução do HDA com Cenário N=30*

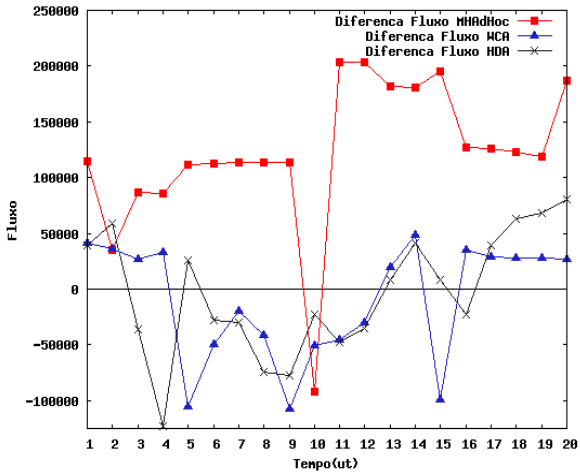


*Figura 69: Diferença de Fluxo Intra/Inter-Cluster para MHAdHoc,WCA e HDA com Cenário N=30*

As ilustrações 71, 72 e 73 mostram o comportamento de $\psi_{intra}$ e $\psi_{inter}$ para N=40 em execuções distintas das técnicas MHAdHoc, WCA e HDA , respectivamente ao longo de $T_{exec}$ com granulosidade $T_g$. Os nós foram dispostos aleatoriamente de forma uniformemente distribuída. A diferença de fluxo $\psi = \psi_{intra} - \psi_{inter}$ de cada técnica ao longo do tempo de execução é mostrada na Figura 74.

Observa-se na Figura 71 que $\psi_t > 0, \forall t \in [1..T_{exec}]$ . Este é um aspecto positivo, visto que durante $T_{exec}$ temos a ocorrência de eventos dinâmicos da rede, mas sem a necessidade de rearranjos forçados de topologia por valores negativos de $\psi$ . Em alguns instantes temos aproximações e afastamentos entre $\psi_{intra}$ e $\psi_{inter}$ . As aproximações ocorrem em *t=2* e *t=16*, com $-\Delta_{\psi(MHAdHoc)}(t=2; t=16)$ , sendo $\Delta_\psi = \psi_t - \psi_{t-1}$ . Os correspondentes afastamentos ocorrem somente após uma aproximação, mas não imediatamente.

No instante *t=13*, observamos um afastamento não imediato ocorrido em $\Delta_\psi(t=13) = \psi_t - \psi_{t-1} > 0$ e nos instantes anteriores não há grandes variações na diferença de fluxo $\Delta_\psi(t-1) \approx \Delta_\psi(t-2) \approx ... \approx \Delta_\psi(t-10) \approx cte$ . A aproximação é vista somente em *t-11*, onde $\Delta_\psi(t-11=2) = \psi_{t-11} - \psi_{t-12} < 0$ . Isto ocorre porque a aproximação em *t-11*, provocado por eventos dinâmicos da rede, não foi suficiente para forçar um rearranjo da topologia, pois a relação de fluxo *intra/inter-cluster* se manteve positiva $\psi(t-11) > 0$ , mesmo que a variação desta relação no tempo tenha se tornado negativa, $\Delta_\psi(t-11) < 0$ . Ao longo dos próximos instantes de tempo, a diferença gerada em *t-11* não resultou em modificações para $\psi_{t'}$ ou para $\Delta_\psi(t')$ , no intervalo *t'={t-11,t-10,t-9,...,t-2,t-1}*, somente no instante de tempo *t*.

Este é um aspecto importante, pois o comportamento da rede ainda não é suficiente para justificar um novo rearranjo de topologia, mesmo que a diferença em $\psi$ tenha diminuído. A tolerância à algumas mudanças é justificada porque a reorganização também apresenta custo à rede, sugerindo a sua execução somente quando for realmente necessário. Logo, um bom método de particionamento em *clusters* deve ser robusto, isto é, que seja capaz de minimizar a necessidade de nova execução ao longo de $T_{exec}$ e ser tolerante aos eventos dinâmicos da rede. Espera-se que a robustez seja uma característica do MHAdHoc.

Como exemplo ilustrativo, suponhamos que em *t-11* apenas um nó *v* sofreu reafiliação, abandonando seu *cluster* atual *i*, $C_i(v)$, passando para $C_j(v)$. Podemos ter uma variação negativa em $\Delta_\psi(t-11)=\psi_{t-11}-\psi_{t-12}$, mas preservando $+\psi_{t-11}$. Esta diferença em $\Delta_\psi$ é mantida no intervalo *t'={t-11,t-10,t-9,...,t-2,t-1}*, sem que haja rearranjo forçado de topologia pela diferença negativa de $\psi<0$. Do exemplo, suponha que eventos dinâmicos da rede tenham ocorrido, mas impactos significativos em $\psi$ e $\Delta_\psi$. Suponha que somente no instante *t*, o nó *v* tenha sofrido nova reafiliação, voltando para $C_i(v)$. Em conseqüência, temos o afastamento com $\Delta_\psi(t)>0$. Observamos, no instante t=1, um pico devido à formação do cenário inicial, com a execução do MHAdHoc. Uma nova topologia é formada segundo a estratégia de minimização de $\psi_{inter}$.

A Figura 72 mostra resultados para o modelo WCA, quanto ao fluxo de comunicação *intra* e *inter-cluster,* $\psi_{intra}(WCA)$ e $\psi_{inter}(WCA)$. Observa-se a ausência na correlação na diferença de fluxo *intra-cluster* e *inter-cluster,* $\psi$, com predominância de $\psi_{intra}<\psi_{inter}$ nos tempos *t=[1,4]* e *t=[11..13]*. Maiores valores negativos de $-\psi$ são observados principalmente em *t=[11,13]*, onde $\psi_{intra}<<<\psi_{inter}$. Isto acontece porque este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, $W_v=W_v(\Delta_v, D_v, P_v, M_v)$. A determinação de $W_v$ utiliza *weight factors* como $\Delta_v$ e $D_v$. Por isso, oscilações no valor de $\psi$ causadas por modificações de $\Delta_v$ ou $D_v$ podem causar nova atualização de *DS(G)*, com risco de prejuízo em $\psi$.

A Figura 73 mostra resultados para o modelo HDA, com relação a $\psi_{intra}(HDA)$ e $\psi_{inter}(HDA)$. Observa-se a ausência na correlação na diferença de fluxo *intra-cluster e inter-cluster,* $\psi$, neste caso com predominância de $\psi_{intra}>\psi_{inter}$, exceto no intervalo *t=[16..18]*. Este resultado é circunstancial, visto que este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, o grau de cada nó. Outro problema relacionado a este modelo é a ausência de um limite de número de nós em um mesmo *cluster*, apresentado maior suscetibilidade à exaustão de *clusterheads*.

A Figura 74 mostra a diferença $\psi=\psi_{intra}-\psi_{inter}$ entre os modelos MHAdHoc, WCA e HDA ao longo do tempo de simulação para os cenários das figuras 71, 72 e 73, onde N=40. Observa-se que a técnica MHAdHoc permaneceu com $+\psi$ ao longo

de todo o tempo de simulação, com algumas oscilações positivas e negativas em $\Delta_{\psi}$ . Concluímos que a geração da topologia no início da simulação suportou eventos dinâmicos da rede ao longo todo o tempo de simulação, sem que houvesse a necessidade de novo rearranjo da rede. A tolerância à algumas oscilações em $\Delta_{\psi}$ é justificada porque a reorganização da topologia também apresenta custo à rede, sugerindo a sua execução somente quando for realmente necessário. Um bom método de particionamento em *clusters* deve ser capaz de minimizar a necessidade de nova execução ao longo de $T_{exec}$ e ser tolerante aos eventos dinâmicos da rede. Observa-se ainda o comportamento difuso de $\Delta\psi(WCA)$ e $\Delta\psi(HDA)$ , com predominância de $-\psi(WCA, HDA)$ em quase todo $T_{exec}$ e oscilações entre valores positivos e negativos, motivados por eventos dinâmicos da rede. Acreditamos que haja ausência de algum mecanismo de controle em função de $\psi$ .

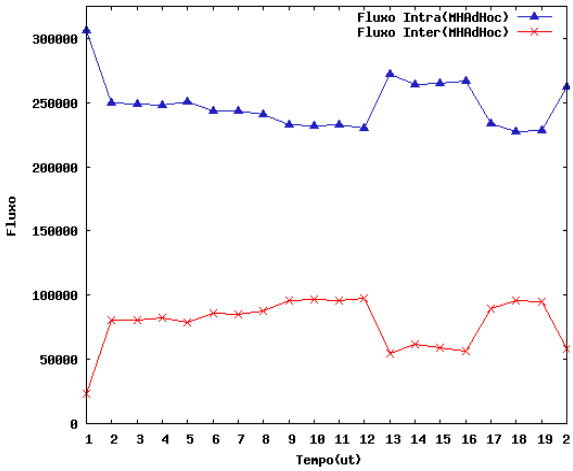


*Figura 71: Fluxo Intra/Inter-Cluster para uma execução do MHAdHoc com Cenário N=40*

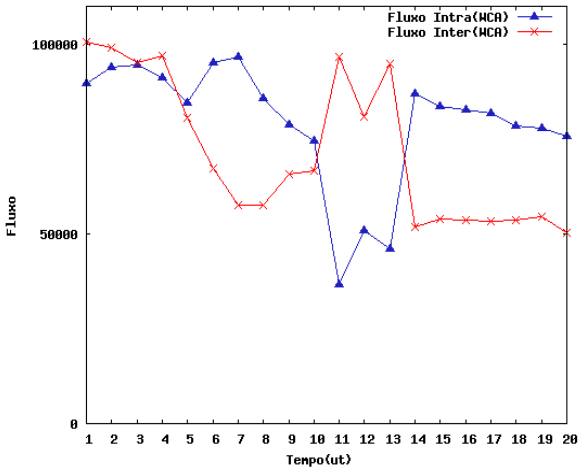


*Figura 72: Fluxo Intra/Inter-Cluster para uma execução do WCA com Cenário N=40*

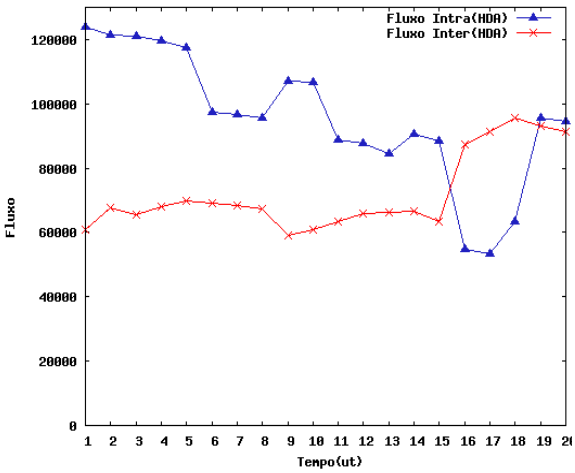


*Figura 73: Fluxo Intra/Inter-Cluster para uma execução do HDA com Cenário N=40*

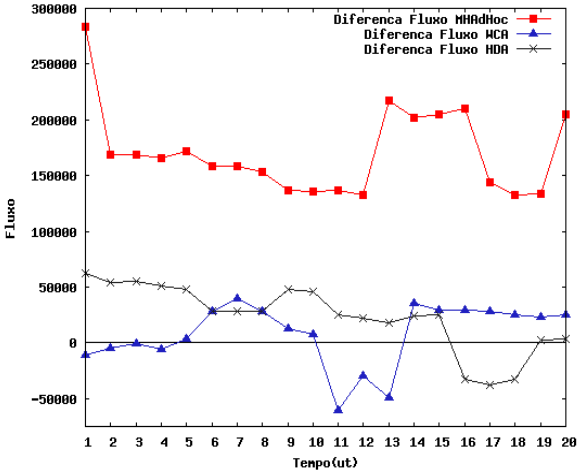


*Figura 74: Diferença de Fluxo Intra/Inter-Cluster para MHAdHoc,WCA e HDA com Cenário N=40*

As ilustrações 75, 76 e 77 mostram o comportamento de $\psi_{intra}$ e $\psi_{inter}$ para N=50 em execuções distintas das técnicas MHAdHoc, WCA e HDA , respectivamente ao longo de $T_{exec}$ com granulosidade $T_g$. Os nós foram dispostos aleatoriamente de forma uniformemente distribuída. A diferença de fluxo $\psi = \psi_{intra} - \psi_{inter}$ de cada técnica ao longo do tempo de execução é mostrada na Figura 78.

Observa-se na Figura 75 que $\psi_t > 0, \forall t \in [1..T_{exec}]$ . Temos duas aproximações, em t=5 e t=16, com apenas um afastamento em t=18. Verifica-se que as maiores diferenças de $\psi$ são verificadas inicialmente. Mesmo assim, este é um aspecto positivo, visto que durante $T_{exec}$ temos a ocorrência de eventos dinâmicos da rede, mas sem a necessidade de rearranjos forçados de topologia por valores negativos de $\psi$ , pois ao longo de $T_{exec}$ não há $\psi < 0$ .

No instante *t=5*, observamos uma aproximação, $\Delta_\psi(t=5) = \psi_t - \psi_{t-1} < 0$ , sem que haja um afastamento correspondente. Nos instantes anteriores não há grandes variações na diferença de fluxo $\Delta_\psi(t-1) \approx \Delta_\psi(t-2) \approx ... \approx \Delta_\psi(t-4) \approx cte$ . Isto ocorre porque houve a formação inicial da topologia no início da simulação e até o instante *t* os eventos dinâmicos da rede não foram suficientes para forçar um rearranjo da topologia, pois a relação de fluxo *intra/inter-cluster* se manteve positiva até então, com $\psi > 0$ , mesmo com pequenas oscilações positivas/negativas de $\Delta_\psi(t')$ , *t'=[t-4,...,t-2,t-1]*.

Como exemplo ilustrativo, suponhamos que em *t=5* apenas um nó *v* sofreu

reafiliação, abandonando seu *cluster* atual *i*, $C_i(v)$, passando para $C_j(v)$. Teremos uma variação negativa em $\Delta_\psi(t=5)=\psi_t-\psi_{t-1}$ , mas preservando $+\psi_{t=5}$ . Ainda neste exemplo, suponha que eventos dinâmicos da rede tenham ocorrido, mas impactos significativos em $\psi$ e $\Delta_\psi$ , não ocasionando afastamento nos instantes seqüentes *{t+1,t+2,t+3,...}*.

No instante t=16, observamos uma nova aproximação, $\Delta_\psi(t=16)<0$ , sem que haja grandes variações na diferença de fluxo nos instantes anteriores $\Delta_\psi(t-1)\approx...\approx\Delta_\psi(t-11)\approx cte$ . Isto ocorre motivado por eventos dinâmicos da rede, não foi suficientes para forçar um rearranjo da topologia, pois a relação de fluxo *intra/inter-cluster* se mantém positiva $\psi(t),\psi(t-1),...,\psi(t-11)>0$ , mesmo que em *t*, $\Delta_\psi(t)<0$ . Verifica-se um afastamento não imediato em t=18, $\Delta_\psi(t=18)>0$ .

O comportamento observado da rede indica um aspecto positivo. A ocorrência de eventos dinâmicos em $T_{exec}$ ainda não é suficiente para justificar um novo rearranjo de topologia, mesmo que a diferença em $\psi$ tenha diminuído. A tolerância à algumas mudanças é justificada porque a reorganização também apresenta custo à rede, sugerindo a sua execução somente quando for realmente necessário. É desejável que um método de particionamento em *clusters* deva ser capaz de minimizar a necessidade de nova execução ao longo de $T_{exec}$ e ser tolerante aos eventos dinâmicos da rede.

Observamos no início da simulação um pico devido à formação do cenário inicial, com a execução do MHAdHoc. Uma nova topologia é formada segundo a estratégia de minimização de $\psi_{inter}$ .

A Figura 76 mostra resultados para o modelo WCA, quanto ao fluxo de comunicação *intra* e *inter-cluster* $\psi_{intra}(WCA)$ e $\psi_{inter}(WCA)$ . Observa-se a ausência na correlação na diferença de fluxo *intra-cluster* e *inter-cluster,* $\psi$ , neste caso, com predominância de $\psi_{intra}>\psi_{inter}$ , exceto em *t=12*. Este resultado é circunstancial, visto que este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, $W_v=W_v(\Delta_v,D_v,P_v,M_v)$ . A determinação de $W_v$ utiliza *weight factors* como $\Delta_v$ e $D_v$ . Por isso, é possível que atualizações do DS(G) causem oscilações negativas no valor de $\psi$ , motivadas por modificações de $\Delta_v$ ou $D_v$ , embora neste caso não tenha sido observado.

A Figura 77 mostra resultados para o modelo HDA, com relação a $\psi_{intra}(HDA)$ e $\psi_{inter}(HDA)$ . Observamos bom desempenho inicial. Em *t=1,* a topologia gerada apresenta $\psi_{intra} >>> \psi_{inter}$ , como indicativo de boa organização do particionamento em *clusters*. No entanto, verifica-se que nos instantes seqüentes esta organização não é sustentada com a ocorrência de eventos dinâmicos. Em *t'=2* temos $\Delta_{\psi}(t') < 0$ . Até que em *t=5* temos $\psi(t=5) < 0$ com $\Delta_{\psi} < 0$ . Além disso, observa-se a ausência na correlação na diferença de fluxo *intra-cluster e inter-cluster,* $\psi$ , com predominância de $\psi_{intra} < \psi_{inter}$ nos intervalos *t=[6,13] e t=[18,20]*. Isto acontece porque, este método não leva em consideração a comunicação entre os nós da rede como fator de decisão, e sim, o grau de cada nó.

A Figura 78 mostra a diferença $\psi = \psi_{intra} - \psi_{inter}$ entre os modelos MHAdHoc, WCA e HDA ao longo do tempo de simulação para os cenários das figuras 75, 76 e 77, onde N=50. Observa-se que a técnica MHAdHoc permaneceu com $+\psi$ ao longo de todo o tempo de simulação, embora $\psi_{t=16}(HDA) > \psi_{t=16}(MHAdHoc)$ e com algumas oscilações positivas e negativas em $\Delta_{\psi}$ . Concluímos que a geração da topologia no início da simulação foi robusta o suficiente para suportar eventos dinâmicos da rede ao longo todo o tempo de simulação, sem que houvesse a necessidade de novo rearranjo da rede, além da ausência de falhas em *clusterheads* e nós sem adjacência. A tolerância à algumas oscilações em $\Delta_{\psi}$ é justificada porque a reorganização da topologia também apresenta custo à rede, sugerindo a sua execução somente quando for realmente necessário.

Observa-se ainda o comportamento difuso de $\psi(WCA)$ e $\psi(HDA)$ , mesmo com $+\psi(WCA)$ em quase todo o tempo de simulação. Por outro lado, observou-se a prevalência de $-\psi(HDA)$ em quase todo $T_{exec}$. Para ambos os casos, foram observadas oscilações entre valores positivos e negativos, motivados por eventos dinâmicos da rede. Acreditamos que haja ausência de algum mecanismo de controle em função de $\psi$ , mesmo com o bom resultado para o modelo WCA neste experimento.

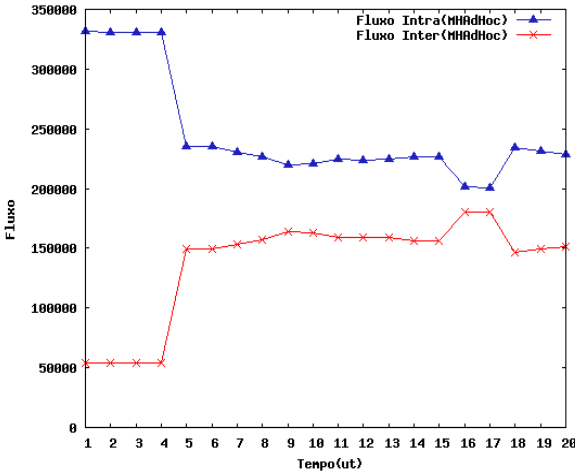


*Figura 75: Fluxo Intra/Inter-Cluster para uma execução do MHAdHoc com Cenário N=50*

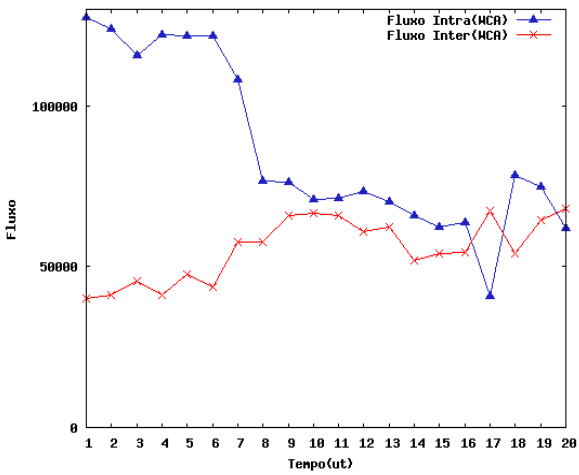


*Figura 76: Fluxo Intra/Inter-Cluster para uma execução do WCA com Cenário N=50*

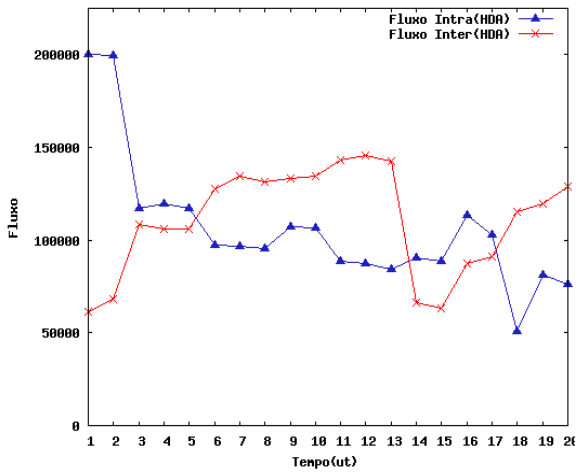


*Figura 77: Fluxo Intra/Inter-Cluster para uma execução do HDA com Cenário N=50*

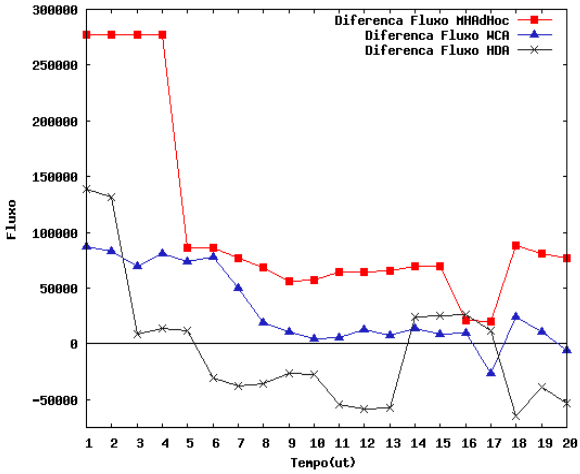


*Figura 78: Diferença de Fluxo Intra/Inter-Cluster para MHAdHoc,WCA e HDA com Cenário N=50*

## 6.8. ANÁLISE DE DESEMPENHO EM RELAÇÃO AO NÚMERO DE REAFILIAÇÕES

Experimentos foram realizados para verificar o parâmetro de desempenho Número de Reafiliações (*NReaf*) entre o modelo proposto, *MHAdHoc,* e outras técnicas já existentes: *Weighted Clustering Algorithm* (WCA) [CHA00] e *Highest-Degree Algorithm*

(HDA) [GER95]. Para efeito de simplificação na coleta e análise de dados, o *MHAdHoc* foi empregado apenas com a metaheurística Busca Tabu.

É importante avaliar o Número de Reafiliações pois é desejável que um bom método de particionamento em *clusters* seja capaz de tolerar, até um certo nível, eventos dinâmicos da rede. Isto inclui as reafiliações dos nós da rede. Cada reafiliação provoca alterações na relação $\psi_{intra} x \psi_{inter}$ , com variações positivas ou negativas de $\Delta\psi(t)$ , no instante *t*. Entende-se como tolerar até um certo nível a manutenção de $\psi > 0$ . Por se tratar de uma rede dinâmica, é muito provável que reafiliações ocorram durante $T_{exec}$, mas é desejável que não causem grande impacto na topologia. Este é um aspecto quantitativo de *NReaf*.

Avaliar *NReaf* também é importante pois uma boa técnica de particionamento deve ser capaz de evitar reafiliações que causem rearranjos forçados de topologia, sob qualquer contexto. Por exemplo, no instante *t*, se os nós $\sum_{m=0}^{M-1} v_m$ , $M < n-2$ , $v_m \in N_{open}(h_{C_i}), C_i \in C$ , sofrem reafiliação, abandonando seus respectivos *clusters* e passando para um *cluster* vizinho $C_j(v)$. Suponha que o impacto na rede é a variação negativa de $\Delta_\psi(t)$ , com $\psi < 0$ . Neste caso, temos o rearranjo forçado da topologia. Em uma primeira análise, podemos supor que somente a quantidade exerce influência sobre $\psi$ . No entanto, em alguns casos, verificou-se experimentalmente que o impacto da reafiliação de um nó é diferente de outro reafiliado. Logo, o fator qualitativo também deve ser levado em consideração. Resta saber como proceder para minimizar as reafiliações de nós que causem maior impacto ao equilíbrio da rede. Outro exemplo, suponha que um nó *v*, $v \in N(h_i)_{open}, C_i(v)$ , sofre RQS se $\Gamma_h(t) = falho, \pi_h = 0, \rho(h(v)) \to 0$ .Se $\exists! h_j(v) \in DS(G) | \rho(h_i(v)) \leqslant \rho(h_j(v))$ então haverá rearranjo forçado de topologia. O objetivo é minimizar a necessidade de rearranjos forçados de topologia ao longo de $T_{exec}$. Este é um aspecto qualitativo de *NReaf*.

Portanto, *NReaf* será analisado sob dois aspectos: quantitativo e qualitativo. Conforme o contexto, um cenário poderá apresentar um grande Número de Reafiliações, mas com baixo impacto na rede. Por outro lado, podemos ter um cenário com pequeno Número de Reafiliações, mas com grande número de rearranjos forçados na topologia.

Considerou-se como reafiliação os seguintes casos: RDP e RQS. O primeiro,

RDP, ocorre quando um nó *v* recebe o sinal com maior potência, $\rho$ de um *clusterhead* vizinho $\rho(h_j(v))$ em relação ao *clusterhead* atual $\rho(h_i(v))$ . Quando isso ocorre, *v* abandona $C_i(v)$ e passa para $C_j(v)$ . A diferença de potência é acrescida por um valor limiar, $\tau$ , com agregação de histerese, $C_i(v), \exists h_j(v) \in DS(G) | \rho(h_i(v)) + \tau \leqslant \rho(h_j(v))$ ou $C_i, \exists h_j(v), h_m(v) \in DS(G) | \rho(h_i(v)) + \tau \leqslant \rho(h_m(v)) \leqslant \rho(h_j(v))$ . O segundo caso é chamado do RQS. Ocorre quando há falha do *clusterhead* de *v*, $v \in N(h)_{open}$ e $\rho(h(v)) \to 0$ ou simplesmente por movimentação abrupta de *v*. O nó *v* é reafiliado para outro *cluster* $C_j$ , cujo *clusterhead* é $h_j$ , da seguinte forma: $\rho(h_i(v)) \to 0, \forall \rho(h'_j(v)) \geqslant 0, \exists h_j(v) | \rho(h_j(v)) \geqslant \rho(h'_j(v)) \geqslant \rho(h_i(v))$ . Em ambos os casos, podemos ter rearranjo forçado ou não.

Para as simulações, o Número de Reafiliações foi observado sob o aspecto quantitativo e qualitativo. O quantitativo é dado pelo número médio de reafiliações sem rearranjo de topologia, $NReaf_s$. O qualitativo é dado pelo número médio de reafiliações com rearranjo forçado de topologia em cada ocorrência, $NReaf_c$. Observou-se desde uma topologia mais simples até uma rede mais complexa. A complexidade aqui é traduzida pelo número de nós da rede (*N*). Considerou-se experimentalmente *N=20*, disposta de forma aleatória uniformemente distribuída. Incrementou-se o número de nós em dez unidades até um valor máximo viável computacionalmente para todos os experimentos apresentados. Obteve-se *N=50*.

Para o tratamento estatístico de dados, foi calculada a Média $\mu$ , Desvio Padrão $\sigma$ , Coeficiente de Variação (*CV*) para cada informação apresentada, a partir de amostragem de experimentos de cada cenário de simulação, *N=[20..50],* aplicada em cada técnica, com $CV \leqslant 5\%$ . O Intervalo de Confiança é de 95%, para o intervalo $[\bar{X} - SE, \bar{X} + SE]$ ,com $\bar{X} = \mu$ , se a amostra for suficientemente grande, $n \geqslant 30$ , e $SE = 1.96 * \sigma / \sqrt{n}$ , onde SE é o Erro Padrão da Média [MON94], [TRI01].

A Figura 79 mostra o desempenho segundo *NReaf* para diferentes topologias, com *N=[20..50]*. Na legenda, MHAdHoc, WCA e HDA representam *NReaf* sem rearranjo de topologia, $Nreaf_s$, como constituintes da visão quantitativa de *NReaf*. MHAdHoc' , WCA' e HDA' representam $NReaf_c$ com rearranjo forçado de topologia, como constituintes da visão qualitativa de *NReaf*.

Observamos que $NReaf_s(WCA) > NReaf_s(MHAdHoc) > NReaf_s(HDA)$ , em média, para todos os cenários de simulação. Observa-se, no entanto, o inverso para as reafiliações com rearranjo de topologia, em quase todos os cenários de simulação. Obteve-se, em média, $NReaf_c(MHAdHoc) < NReaf_c(WCA) \leqslant NReaf_c(HDA)$ .

Concluímos que:

- O modelo proposto apresenta desempenho superior com relação a *NReaf*, pela tolerância a reafiliações sem a necessidade de rearranjos de topologia. O $NReaf_c$ médio foi inferior, em média, para todos os cenários de simulação, mesmo que $NReaf_s$ médio não tenha sido o mais inferior.
- O HDA apresenta menor *NReafs*. Acreditamos que contribua para isso a ausência de um limite de número de nós em um mesmo *cluster*.

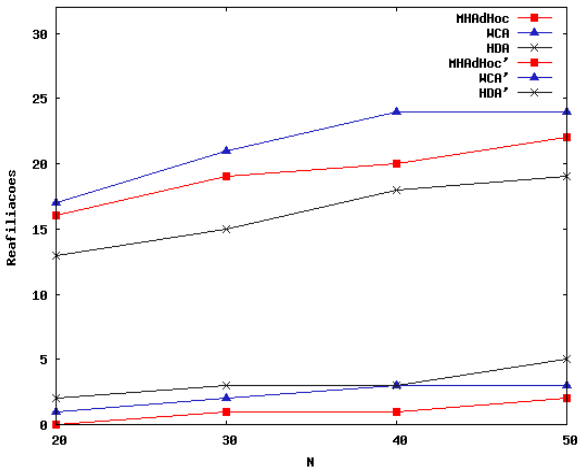


*Figura 79: Análise de Desempenho segundo o Número de Reafiliações (NReaf) para diferentes topologias*

## 6.9. ANÁLISE DE DESEMPENHO EM RELAÇÃO AO NÚMERO DE ATUALIZAÇÕES DO *DOMINANT SET* DS(G)

Experimentos foram realizados para verificar o parâmetro de desempenho Número de Atualizações do *Dominant Set* (*DS(G)*) entre o modelo proposto, *MHAdHoc,* e outras técnicas já existentes: *Weighted Clustering Algorithm* (WCA) [CHA00] e *Highest-*

*Degree Algorithm* (HDA) [GER95]. Para efeito de simplificação na coleta e análise de dados, o *MHAdHoc* foi empregado apenas com a metaheurística Busca Tabu.

É importante avaliar o Número de Atualizações de DS(G) pois está diretamente relacionado aos rearranjos de topologia. Deseja-se que uma boa técnica de particionamento em *clusters* tenha a capacidade de minimizar a necessidade de atualizações de DS(G) à medida que a rede se movimenta [CHA00].

Para as simulações, o Número de Atualizações de DS(G) foi observado desde uma topologia mais simples até uma rede mais complexa. A complexidade aqui é traduzida pelo número de nós da rede (*N*). Considerou-se experimentalmente *N=20*, disposta de forma aleatória uniformemente distribuída. Incrementou-se o número de nós em dez unidades até um valor máximo viável computacionalmente para todos os experimentos apresentados. Obteve-se *N=50*.

Para o tratamento estatístico de dados, foi calculada a Média $\mu$ , Desvio Padrão $\sigma$ , Coeficiente de Variação (*CV*) para cada informação apresentada, a partir de amostragem de experimentos de cada cenário de simulação, *N=[20..50],* aplicada em cada técnica, com $CV \leqslant 5\%$ . O Intervalo de Confiança é de 95%, para o intervalo $[\bar{X} - SE, \bar{X} + SE]$ ,com $\bar{X} = \mu$ , se a amostra for suficientemente grande, $n \geqslant 30$ , e $SE = 1.96 * \sigma / \sqrt{n}$ , onde SE é o Erro Padrão da Média [MON94], [TRI01].

A Figura 80 mostra o desempenho segundo o Número de Atualizações de DS(G) para diferentes topologias, com *N=[20..50]*.

Observamos que em quase todos os cenários de simulação que o Número de Atualizações de DS(G), em média, é inferior para o modelo proposto.

Concluímos que:

- O modelo proposto apresenta desempenho superior com relação ao Número de Atualizações do *Dominant Set*, com a minimização de rearranjos indesejados de topologia ao longo do tempo de cada simulação.

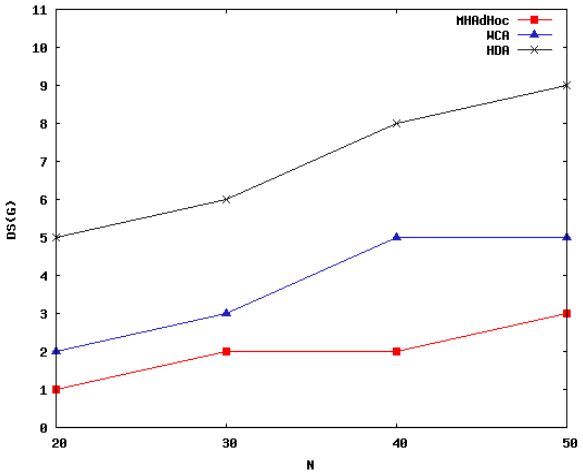


*Figura 80: Análise de Desempenho segundo o Número de Atualizações de DS(G) para diferentes topologias*

## 6.10. OTIMIZAÇÃO DO TSADHOC POR SELEÇÃO DE VIZINHO LOCAL POR EXTREMIDADES DE PARTIDA (TS+SLEP) – DESEMPENHO EM RELAÇÃO À FUNÇÃO OBJETIVO E AO TEMPO DE EXECUÇÃO

É proposto um melhoramento no processo de seleção de um vizinho $s$ de $s_0$ do algoritmo TS acoplado ao *TSAdHoc*.

O objetivo é melhorar o desempenho da técnica para a convergência de resultados quase-ótimos e sem prejuízo no tempo de execução, $T_{exec.}$

A implementação básica do TS apresenta como critério de seleção a aleatoriedade de $s$, $s \in N_{open}(s_0)$ [GLO89], como base do método de seleção de Vizinho Aleatório (VA). Foi implementado o método *Tsadhoc::geraUmVizinhoAleatorio()*, na classe *Tsadhoc*, associado ao método *Tsadhoc::selecionaVizinhoTabuS()*.

A idéia básica é melhorar o desempenho do algoritmo com base no conceito de Extremidades de Partida ($OE(C_i)$) de um *cluster* $C_i$, $C_i \in C$, selecionado de forma aleatória em distribuição uniforme.

Inicialmente, imaginou-se o desenvolvimento de um método baseado em Busca

Local (BL) [RES95] associado ao procedimento de seleção de *s*. Foi implementado o método *Tsadhoc::geraUmVizinhoAleatorioAPartirDeBuscaLocal()*, na classe *Tsadhoc*, associado ao método *Tsadhoc::selecionaVizinhoTabuS()*, que seleciona o método escolhido, conforme cada caso.

No entanto, a aplicação de um mecanismo de BL obteve problemas. Para o problema de particionamento em *clusters*, constatou-se experimentalmente uma sensível melhora de desempenho, mas com grande prejuízo no tempo de execução. Constatou-se que peculiaridades ligadas às condicionantes restritivas de cada solução $s$ , em particular, aos possíveis movimentos, *m,* para soluções vizinhas, poderiam prejudicar não a busca, mas o tempo de busca. Em alguns casos, o algoritmo atinge soluções inválidas indesejadas após um movimento *m*, $s \leftarrow s_0 \circ m$ , com *s* apresentando alguma condição restritiva inválida. Algumas vezes, não é capaz de sair de *s* e buscar novas movimentações ou escapa após um intervalo significativo de tempo. Por se tratar de uma técnica heurística, o emprego de BL por si pode não ser tão eficaz como melhoramento no processo de seleção *s* de $s_0$.

A solução foi adaptar a Busca Local sobre um domínio restritivo de movimentos *m*, baseando-se no conceito de Extremidades de Partida (EP) ou *Outgoing Edge* $OE(\langle f \rangle_{MST})$ sobre uma árvore geradora mínima (MST), definido da seguinte forma: $G=(V,E), \langle f \rangle_{MST} \in G, v \in \langle f \rangle, v' \in \overline{\langle f \rangle}, |OE(\langle f \rangle) \Leftrightarrow \overline{v, v'}$ , sem que haja a formação de ciclos em $\langle f \rangle_{MST}$ [GAL83].

Esta variante é chamada de Busca Local por Extremidades de Partida (BLEP) ou Seleção Local por Extremidades de Partida (SLEP). Como adaptação, o fragmento de uma árvore geradora mínima é um *cluster* da rede, isto é, $\langle f \rangle_{MST} \rightarrow C_i, C_i \in C$ . O nó *v* pertence ao *cluster* $C_i(v)$, $v \in N_{open}(h_{C_i})$ e $v'$ é alcançável por *v* e pertence a outro *cluster* $C_j, C_j \in C$ , ou seja, $v' \in N_{open}(h_{C_j}), i \neq j$ , desde que $\exists \overline{v, v'}$ .

A implementação é dada da seguinte forma: a matriz $g_{ik}$ reprsenta uma solução *s*, onde cada linha representa um nó da topologia e cada coluna um *cluster* de G, *G(V,E)*. $C_i$ é representado pela i-ésima coluna, *i=[0..K-1]*, formada pela seqüência de *clusternodes w* que a compõem, $w=\{w_0, w_1, \ldots, w_{M-1}\} | C_i(w), M \leq N-2$ . Verifica-se a lista de Extremidades de Partida obtidas a partir de $\{w_0, w_1, \ldots, w_k, \ldots, w_{M-1}\}$ ,do cluster

$C_i$, *OEL($C_i$),* dado por:

$$OEL(C_i)=\begin{matrix}\{OE_0(C_i(w_0)), & OE_1(C_i(w_0)), & OE_2(C_i(w_0)), & \cdots\\ ,OE_0(C_i(w_1)), & OE_1(C_i(w_1)), & OE_2(C_i(w_1)), & \cdots\\ \cdots & \cdots & \cdots & \cdots\\ ,OE_0(C_i(w_k)), & OE_1(C_i(w_k)), & OE_2(C_i(w_k)), & \cdots\\ \cdots & \cdots & \cdots & \cdots\\ ,OE_0(C_i(w_{M-1})), & OE_1(C_i(w_{M-1})), & OE_2(C_i(w_{M-1})) & \cdots\}\end{matrix}$$

Seleciona-se aleatoriamente $w_k$ e OE a partir de $w_k$. , $OE(C_i(w_k))$ . Se não existir OE a partir de $w_k$, outro nó $w_{k'}$ é selecionado até que seja obtido $OE(C_i(w_k'))$ . A Extremidade de Partida $OE(C_i(w_k))$ constituir-se-á no movimento *m*, da solução atual $s_0$ para uma solução vizinha *s,* $s \leftarrow s_0 \circ m$ . Esta variante está implementada no método *Tsadhoc::geraUmVizinhoAleatorioAPartirDeExtremidadeDePartida()*, associado ao método *Tsadhoc::selecionaVizinhoTabuS()*.

A aleatoriedade no processo de busca de *s*, $s \in N_{open}(s_0)$ , pelo movimento *m*, $s \leftarrow s_0 \circ m$ , é obtida pela seleção aleatória de uma Extremidade de Partida do *cluster* $C_i$ a partir de *w*, $OE(C_i(w))$ .

Experimentos foram realizados para verificar o desempenho do método de Seleção Local por Extremidade de Partida (TS+SLEP) em relação à Busca Local (TS+BL) e Vizinho Aleatório (TS+VA). Para efeito de simplificação na coleta e análise de dados, a seleção de um vizinho aleatório *s* de $s_0$ foi empregado apenas com a metaheurística *Busca Tabu*, por não ser um PBA.

Para as simulações, utilizou-se como parâmetros de desempenho o resultado final de cada técnica, isto é, o valor da Função Objetivo $F_o$ alcançado, para medir a qualidade da técnica, e o tempo necessário para se obtê-la, ou seja, o Tempo de Execução ($T_{exec}$) entre diferentes cenários, desde uma topologia mais simples até uma rede mais complexa. A complexidade aqui é traduzida pelo número de nós da rede (*N*). Considerou-se experimentalmente *N=20*, disposta de forma aleatória uniformemente distribuída. Incrementou-se o número de nós em dez unidades até um valor máximo viável computacionalmente para todos os experimentos apresentados. Obteve-se *N=50*.

Como tratamento estatístico, foi calculada a Média $\mu$ , Desvio Padrão $\sigma$ , Coeficiente de Variação (*CV*) para cada informação apresentada, a partir de amostragem de

experimentos de cada cenário de simulação, *N=[20..50],* aplicada em cada técnica, com $CV \leqslant 5\%$ . O Intervalo de Confiança é de 95%, para o intervalo $[\bar{X} - SE, \bar{X} + SE]$ ,com $\bar{X} = \mu$ , se a amostra for suficientemente grande, $n \geqslant 30$ , e $SE = 1.96 * \sigma / \sqrt{n}$ , onde SE é o Erro Padrão da Média [MON94], [TRI01].

As ilustrações 81 e 82 mostram o desempenho segundo o resultado da Função Objetivo ($F_0$) e o Tempo de Execução ($T_{exec}$), respectivamente, para diferentes topologias, com *N=[20;50]*, envolvendo as técnicas TS+SLEP (TS+BLEP), TS+BL e TS+VA, implementadas como métodos de seleção de vizinho *s* de $s_0$.

Observamos menores valores, em média, da Função Objetivo e o Tempo de Execução em quase todos os cenários para o método TS+SLEP. Para $F_0$, verificam-se diferenças pequenas de TS+SLEP em relação ao TS+BL, acentuando-se para N=50. Ambos com desempenho superior ao TS+VA. Por outro lado, para $T_{exec}$, verificam-se diferenças de TS+SLEP em relação agora com TS+VA, com uma pequena vantagem deste para N=20. Observa-se desempenho bem inferior para TS+BL.

Concluímos que:

- A aplicação de um mecanismo de Busca Local apresenta melhora de desempenho, mas com grande prejuízo a $T_{exec}$. Acredita-se que peculiaridades ligadas às condicionantes restritivas de cada solução *s*, em particular, aos possíveis movimentos, *m,* para soluções vizinhas de $s_0$, poderiam prejudicar não a busca, mas o tempo de busca. Durante as simulações, observou-se, em alguns casos, que o algoritmo atinge soluções inválidas indesejadas após um movimento *m*, $s \leftarrow s_0 \circ m$ . Algumas vezes, não é capaz de sair de *s* e buscar novas movimentações ou escapa após um intervalo significativo de tempo. Por se tratar de uma técnica heurística, o emprego de BL por si pode não ser tão eficaz como melhoramento no processo de seleção *s* de $s_0$.
- A adaptação do conceito de $OE(\langle f \rangle_{MST})$ como domínio restritivo de movimentos *m* para o mecanismo de Busca Local no TS, chamada de TS+SLEP, apresenta melhor convergência para soluções quase-ótimas em relação ao TS+BL e ao TS+VA, este último com grande diferença.
- O TS+SLEP não sacrifica o Tempo de Execução ($T_{exec}$) em detrimento do

desempenho de $F_0$, como ocorre no TS+BL. Pelo contrário, em geral, apresentou $T_{exec}$ médio inferior ao TS+VA.

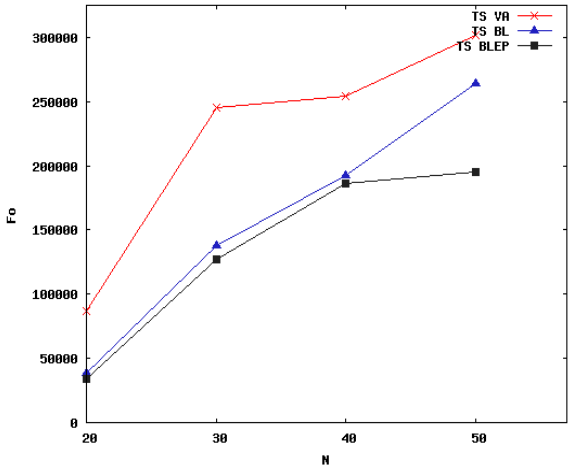


*Figura 81: Análise de Desempenho das técnicas SLEP, BL e VA, segundo o resultado da Função Objetivo para diferentes topologias*

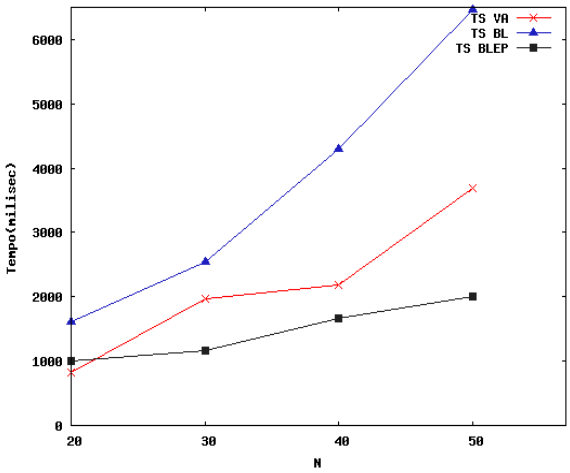


*Figura 82: Análise de Desempenho das técnicas SLEP, BL e VA, segundo o Tempo de Execução para diferentes topologias*

# 7-CONCLUSÕES

## 7.1. SOBRE O TRABALHO REALIZADO

No presente trabalho, foram estudadas três técnicas de metaheurísticas (MH), a citar: Busca Tabu, *Simulated Annealing* e Algoritmos Genéticos.

Estas MH foram aplicadas para determinar um quase-ótimo *clustering* em redes *ad hoc*, com certo grau de mobilidade. A primeira etapa determina o melhor *cluster* dentro de um intervalo considerado. A segunda, determina o melhor *clusterhead*. Esta etapa é realizada a partir da divisão de *clusternodes* feita na etapa anterior.

Foi implementado um protótipo, chamado MHAdHoc, com base nas MH estudadas, recursos da linguagem C++, *Shell Script/Gnuplot*, para automatização de tarefas e plotagem de gráficos e foi utilizada a biblioteca de funções para algoritmos genéticos, chamada *libGA*. Parte do ambiente de simulação foi desenvolvido em linguagem JAVA (JAVA2D). A modelagem do sistema foi realizada com o Umbrello UML Modeller, versão para KDE Linux.

Cada MH foi implementada e estudada de forma distinta no protótipo MHAdHoc: TSAdHoc, SAdHoc e GAdHoc, com base nos algoritmos estudados, TS, SA e GA, respectivamente. Cada qual foi submetida a simulações e análise comparativa entre si e com outros modelos de *clustering*, a fim de analisar vantagens e desvantagens de cada uma e concluir, de forma geral, a melhor.Estes modelos também são comparados com outras técnicas já existentes para comparação de resultados e validação do presente estudo.

Para o estudo comparativo, apresentamos uma análise do comportamento da topologia de *clusters* já formada, à medida que os nós se movimentam, realizam Reafiliações por Diferença de Potência e Reafiliações por Queda de Sinal, consomem energia, transmitem dados, falham, mudam de estados, conforme o seu ciclo de vida, e participam da organização da rede, se ativos, como *clusterhead* ou *clusternode*. Avalia-se a necessidade de rearranjo de topologia conforme critérios de desempenho em um determinado instante de tempo. Um bom modelo de particionamento em *clusters* também

deve garantir rearranjos mínimos de topologia, sem prejuízo de desempenho, mantendo e minimizar o mínimo de Atualizações do *Dominant Set*. Outro fator importante ligado à topologia já formada ao longo do tempo *t* é avaliar o comportamento da rede em função das reafiliações (*handoffs*) de nós da rede, não somente ao seu quantitativo, mas particularmente aos nós reafiliados que forçam rearranjos de topologia, sob aspecto qualitativo. Uma boa topologia deve ser menos suscetível a rearranjos de topologia por nós reafiliados. Por fim, outro fator importante para uma boa topologia é o número de rearranjos da rede ao longo de *t*.

Demonstrou-se que o modelo proposto apresentou, em geral, desempenho melhor que outros já existentes, pois adota um modelo de mobilidade e disponibilidade mais realísticos, agregados ao MHAdHoc.

O modelo de mobilidade adotado no presente estudo tem como base o modelo BSAMM [HAA97b] para criar um raio de curvatura mínimo, $\theta$ , obtido experimentalmente. As mudanças de direção, $\Delta\theta$ , ocorrem de forma suavizada, com base em uma adaptação simplificada do modelo SRMM [BET01b]. Para os casos de mobilidade em grupo, foi adotado o modelo RPGM [GER99b]. A regra de borda adotada foi o modelo *bounce* [HAA98], [BET01]. Utilizou-se, como base, o movimento vetorial dos nós de forma relativa à vizinhança de cada nó, diferentemente de modelos escalares, mais distantes da realidade. Isto foi feito com a finalidade de tornar o modelo mais realístico.

A disponibilidade foi escolhida para implementação, dada a natureza limitada de recursos de uma rede *ad hoc*. É desejável que um modelo realístico considere também a possibilidade de falhas dos elementos da rede. Por isso, foi proposto um melhoramento no modelo baseado em pesos [CHA00]. Adaptou-se ao WCA um quinto elemento, disponibilidade, $\zeta$ . Modelos já existentes não levam em consideração a disponibilidade dos nós.

## 7.2. SOBRE CADA METAHEURÍSTICA EM ESTUDO

Entre os modelos baseados nas diferentes MH estudadas, verificou-se que, em geral, o modelo TSAdHoc apresentou melhores resultados entre os modelos propostos e os

já existentes.

O TSAdHoc apresentou melhor desempenho, ao longo dos diferentes cenários de simulação, desde uma topologia mais simples até uma rede mais complexa. Em todos os cenários de simulação analisados, verificou-se que $F_0(TSAdHoc) < F_0(SAdHoc)$ e $F_0(TSAdHoc) \ll F_0(GAdHoc), F_0(WCA, HDA)$ .

O SAdHoc apresentou o segundo melhor desempenho em relação aos propostos e os já existentes, citados nesta trabalho, com ligeira diferença em relação ao TSAdHoc. Isto é justificado pela suscetibilidade à complexidade da topologia em função da determinação da Temperatura Inicial Experimental ($T_0$). Na implementação do SA, acoplado ao *SAdHoc*, optou-se por obter experimentalmente a Temperatura Inicial ($T_0$) e não de forma simplesmente aleatória. Como o valor ideal de $T_0$ depende de cada problema, selecionar aleatoriamente $T_0$ há a possibilidade de surgimento de condições não-ideais de aquecimento do sistema, seja para o excesso ou para a falta. Durante esta fase, temos a verificação de $p_{aceitacao}(s_0)$ , como referência de aquecimento inicial ideal. Quanto maior o número de nós, este processo consome mais tempo de processamento. Em suma, trata-se de uma troca, visando maximizar melhores resultados de $F_0$ para o *SAdHoc*, sem que haja grandes comprometimentos de $T_{exec}$, o que de fato foi observado.

O GAdHoc apresentou resultados superiores em relação aos modelos já existentes, mas com prejuízo em $T_{exec}$. Isto ocorre porque tem-se a necessidade de geração populacional, em geral com um número significativo de cromossomos para maximizar a diversidade de indivíduos, dentro de um domínio de soluções. Mas isto tem um preço, todos os indivíduos passam por verificações de validade para não se tornarem soluções impossíveis ou não realísticas, acaba consumindo grande para do tempo total de execução do algoritmo. Isto ocorre também nas demais MH propostas, mas por se tratarem de algoritmos não-populacionais, esta verificação se restringe à ordem de grandeza de indivíduos e não de populações, como ocorre no GA. Em conseqüência, colabora também para o baixo desempenho, a aplicação destes mesmos mecanismos de validação para as gerações seguintes. Mesmo que possam herdar melhor aptidão para sobreviver em relação aos seus cromossomos pais, verificou-se experimentalmente que alguns cromossomos filhos podem não corresponder à soluções realísticas da rede. A melhoria no processo de *crossover*, onde temos a escolha de um ponto de corte aleatório distribuído uniformemente,

como referência de divisão de material genético à próxima geração de cada casal de indivíduos que passam pelo processo de *crossover*, pode ser proposta como trabalhos futuros, numa possível tentativa de melhora de desempenho do GA acoplado ao *GAdHoc*.

Outro estudo apresentado está relacionado à determinados parâmetros das MH em estudo. Esta análise complementar foi motivada por serem obtidos experimentalmente e variam não só conforme o problema e a sua complexidade, mas sim, a cada execução do algoritmo. Analisou-se a obtenção da Temperatura Inicial ($T_0$) no SA e o ciclo de aquecimento e resfriamento do sistema no SA.

A grande vantagem da obtenção de uma temperatura inicial $T_0$ é a particularização do estado ideal de aquecimento do sistema para cada problema. Isto evita a aleatoriedade de $T_0$ passível de ser mais baixa ou mais alta que o necessário. Uma desvantagem é o custo computacional acrescido a $T_{exec.}$ A observação do ciclo de aquecimento e resfriamento é importante pois $iter \neq iter'$ e $T_{final} \neq T'_{final}$, mesmo com a igualdade de cenário de simulação para execuções distintas. Isto ocorre porque, mesmo para duas execuções distintas de um mesmo cenário, o SA sempre parte de uma solução inicial aleatória $s_0$.

### 7.3. SOBRE O MELHORAMENTO PROPOSTO TS+SLEP

Um melhoramento no processo de seleção de vizinhos $s$ de $s_0$ é proposto baseado em Extremidades de Partida (EP), $OE(\langle f \rangle_{MST})$, com a adaptação do domínio restritivo de movimentos $m$ para o mecanismo de Busca Local, chamado de Seleção por Extremidades de Partida (SLEP). Experimentalmente, foram obtidos melhores resultados quase-ótimos e com menor tempo de execução $T_{exec}$, em relação a outros critérios de seleção tradicionais: Vizinho Aleatório (VA) e Busca Local (BL). Todos implementados apenas no TS, para simplificação de escopo de problema, denominados, respectivamente de TS+SLEP, TS+VA e TS+BL.

A aleatoriedade no processo de busca de $s$, $s \in N_{open}(s_0)$, pelo movimento $m$, $s \leftarrow s_0 \circ m$, é obtida pela seleção aleatória de uma Extremidade de Partida do *cluster* $C_i$ a partir de $w$, $OE(C_i(w))$.

A grande vantagem do TS+SLEP em relação ao TS+BL é não sacrificar o

Tempo de Execução ($T_{exec}$) em detrimento do desempenho de $F_0$, como ocorre no TS+BL. Por se tratar de uma técnica heurística, o emprego de BL por si não foi adequado como melhoramento no processo de seleção $s$ de $s_0$. A solução foi adaptar a Busca Local sobre um domínio restritivo de movimentos $m$, baseando-se no conceito de Extremidades de Partida (EP) ou *Outgoing Edge* $OE(\langle f \rangle_{MST})$ sobre uma árvore geradora mínima (MST), sem que haja a formação de ciclos em $\langle f \rangle_{MST}$ [GAL83].

## 7.4. TRABALHOS FUTUROS

Como trabalhos futuros, serão estudadas novas técnicas de MH, para análise de viabilidade e desempenho em relação às técnicas propostas. Outra proposta de trabalho é o melhoramento das técnicas em estudo, a exemplo do TS+SLEP. Serão implementados protocolos de roteamento, como *Dynamic Source Routing* [JOH96], *Ad Hoc On-demand Distance Vector routing* [PER99] e *FSR* [GER99a] para adição de novos parâmetros de análise, como a perda de pacotes e avaliar adequação destes protocolos ao modelo proposto. Procuraremos aumentar a quantidade de modelos de mobilidade para comparação de resultados. Por fim, procurar a melhoria da complexidade de tempo, espaço e mensagens no presente estudo.

## 7.5. PUBLICAÇÕES REFERENTES A ESTE TRABALHO

GARCIA, H. F., GONDIM, P. R. de L. "*GAdHoc: um Algoritmo Genético de Particionamento k-Clustering em Redes Ad hoc"*.XXXVII Simpósio Brasileiro de Pesquisa Operacional - SBPO, Setembro de 2005, Gramado-RS-Brasil.

GARCIA, H. F., GONDIM, P. R. de L. "*Clustering in Wireless Ad Hoc Networks using Simulated Annealing and Graph Domination*". (A ser submetido).

GARCIA, H. F., GONDIM, P. R. de L. "A *Tabu Search Technique for Clustering in Wireless Mobile Ad hoc Networks*". (A ser submetido).

GARCIA, H. F., GONDIM, P. R. de L. "*Improving Tabu Search Technique for Clustering in Wireless Ad Hoc Networks using Outgoing Edges"*. (A ser submetido).

GARCIA, H. F., GONDIM, P. R. de L. "*Clustering em Redes Ad Hoc using*

*Metaheuristics Techniques and Graph Domination"*. (A ser submetido).

# REFERÊNCIAS BIBLIOGRÁFICAS


[ALZ02a] K. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks," in Proceedings of the 3rd ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), EPFL Lausanne, Switzerland, 2002, pp. 157–164.

[ALZ02b] P. Wan, K. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in Proceedings of Infocom 2002, 2002.

[ALZ03] K. M. Alzoubi, P.-J. Wan, and O. Frieder, Weakly-connected dominating sets and sparse spanners in wireless ad hoc networks, in The 23rd International Conference on Distributed Computing Systems (IEEE, ICDCS), May 2003.

[AMI99] A. Amin. Simulated Jumping. In Annals of Operations Research, 86, pages 23-38, 1999.

[ARR02] J.E.C.Arroyo. Heurísticas e metaheurísticas para otimização combinatória multiobjetivo. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas,256f., 2002.

[BAI03] F.Bai, A.Helmy. A Survey of Mobility Models in Wireless Adhoc Networks.Chapter 1, University of Southern Califórnia. 2003.

[BAI03b] F. Bai, N. Sadagopan, A. Helmy, Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks, Proceedings of IEEE Information Communications Conference (INFOCOM 2003), San Francisco, April. 2003.

[BAI03c] F.Bai, N. Sadagopan, B. Krishnamachari, A. Helmy. Paths: Analysis of path duration statistics and their impact on reactive manet routing protocols. In MobiHoc 03, June 2003.

[BAN00] S. Banerjee and S. Khuller. A clustering scheme for hierarchical routing in wireless networks. Technical Report CS-TR-4103, University of Maryland, College Park, February 2000.

[BAN98] W. Banzhar, P. Nordin, R.E. Keller, F.D.Francone. Genetic Programming: an Introduction. ISBN 155860510X. Morgan Kaufmann,1998.

[BAS97] S. Basagni, I. Chlamtac and A. Farago, A generalized clustering Algorithm for peer-to-peer networks, in Proceedings of Workshop onAlgorithmic Aspects of Communication (satellite workshop of ICALP),July 1997.

[BAS99a] S. Basagni, Distributed clustering for ad hoc networks, Proceedings of International Sysmposium on Parallel Architectures, Algorithms and Networks, pp.310-315, Junho de 1999.

[BAS99b] S. Basagni, Distributed and mobility-adaptive clustering for multimedia

support in multi-hop wireless networks, Proceedings of Vehicular Technology Conference-VTC, vol. 2, pp. 880-893, June 1999.

[BAS99c] S. Basagni, Distributed clustering for ad hoc networks, in Proc. ISPAN'99 Int. Symp. On Parallel Architectures, Algorithms, and Networks, pp. 310-315, 1999.

[BET01] C.Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement and border effectsACM Mobile Computing and Comunications Review, May 2001.

[BET01b] C. Bettstetter. Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks, ACM Intern. Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, July 2001.

[BOE94] K.Boese, A.B.Kahng,S.Muddu. A New Adaptive Multi-Start Technique for Combinatorial Global Optimizations. In Operations Research Letters, 16(2),Sept 1994, pages 101-113, 1994.

[BOL02] J. Boleng, W. Navidi, T. Camp. Metrics to enable adaptive protocols for mobile ad hoc networks.Proceedings of the International Conference on Wireless Networks, pp. 293–298, 2002.

[BRO98] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva. Multi-hop wireless ad hoc network routing protocols. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), pp. 85–97, 1998.

[CHA00] M. Chatterjee, S.K. Das and D. Turgut, An On-Demand Weighted Clustering Algorithm (WCA) for Ad Hoc Networks, Proceedings of IEEE GLOBECOM 2000, San Francisco, November 2000, pp.1697-1701.

[CHA02] M. Chatterjee, S.K. Das and D. Turgut, WCA: An Weighted ClusteringAlgorithm for Mobile Ad Hoc Networks, Journal of Clustering Computing, Vol.5, No.2, April 2002, pp.193-204.

[CHA97] M. Chatterjee, P. Krishna, N. H. Vaidya and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. Computer Communication Review, 1997.

[CHE02] G. Chen, F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic, Connectivity-based k-hop clustering in wireless networks, in Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35), January 2002.

[CHE02b] Y. P. Chen and A. L. Liestman, Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks, in The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02), pp. 165-172, June 2002.

[CHE03] Y. Chen, A. L. Liestman, and J. Liu. Clustering Algorithms for Ad Hoc Wireless Networks. 2003.

[CHE03b] Y. P. Chen and A. L. Liestman. A zonal algorithm for clustering ad hoc networks, International Journal of Foundations of Computer Science, pp. 305-322, 2003.

[CHE99] G. Chen and I. Stojmenovic. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, June 1999.

[COR05] A.L.Corcoran, disponível em http://www-cgi.cs.cmu.edu/afs/cs/project/ai repository/ai/areas/genetic/ga/systems/libga/0.html. Último acesso em 20 de Agosto de 2005.

[COR90] T.H. Cormen, C.E. Leiserson, R.L. Rivest. Introduction to algorithms. The MIT Press, 1990.

[CUN97] V.Cung, T. Mautor, P.Michelon, A Tavares. A Scatter Search Based Approach for the Quadratic Assignment Problem. In Proceedings of IEEE International Conference on Evolutionary Computation and Evolutionary Programming, Indianapolis, U.S.A., pages 161-170, 1997.

[DAS97] B. Das, V. Bharghavan, Routing in ad-hoc networks using minimum connected dominating sets, in IEEE International Conference on Communications (ICC'97), Vol.1, pp. 376-380, June 1997.

[DAV00] V. Davies. Evaluating mobility models within an ad hoc network. Master's thesis, Colorado School of Mines,2000.

[DAV02] T.Camp, J.Boleng, V.Davies. A Survey of Mobility Models for Ad Hoc Network Research. Colorado School of Mines. 2002.

[DAW76] R.Dawkins.The Selfish Gene, Oxford-UK, Oxford University Press.,1976.

[DOR92] M. Dorigo. Optimization, learning and natural algorithms. In Ph.D.Thesis, Politecnico di Milano, Italy, 1992

[DOR94] R.E. Dorsey, W.J. Mayer, "A Genetic Algorithm for the Training of Feedforward Neural Networks," Advances in Artificial Intelligence in Economics, Finance and Management, Vol. 1. Greenwich, CT: JAI Press Inc., 1994.

[DOR95a] R.E. Dorsey, W.J. Mayer, "Optimization Using Genetic Algorithms," Advances in Artificial Intelligence in Economics, Finance, and Management, Vol. 1. Greenwich, CT: JAI Press Inc., 69-91,1995.

[DOR95b] R.E. Dorsey, W.J. Mayer, "Genetic Algorithms for Estimation Problems with Multiple Optima, Non-Differentiability, and other Irregular Features," Journal of Business and Economic Statistics, 13(1), 53-66,1995.

[DOR95c] R.E. Dorsey, W.J. Mayer, "The Genetic Adaptive Neural Network Training (GANNT) for Generic Feedforward Artificial Neural Systems," School of Business Administration Working Paper, 1995.

[DOR99] M. Dorigo, G. Di Caro, L. Cambardella. Ant Algorithms for Discrete Optimization. Artificial Life, 5(2):137-172. Also available as Technical Report No. 98-10, Université Libre de Bruxelles, Belgium,1999.

[DUB03] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, A. Srinivasan, Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons, in Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 717-724, 2003.

[EPH87] A. Ephremides, J. E. Wieselthier, and D.J.Baker, A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling,

Proceedings of the IEEE, Vol. 75, No.1, January 1987, pp.56-73.

[FAU94] L.Fausset. Fundamentals of Neural Networks: architectures, algorithms, and applications, New York: Prentice Hall,ISBN 0-13-334186-0, 1994.

[FRE01] J. A. Freebersyser and B. Leiner, A DoD perspective on mobile ad hoc networks, in Ad Hoc Networking, C. Perkins, ed., Addison-Wesley, 2001.

[GAL83] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning tree. ACM Transactions on Programming Languages and Systems, January 1983.

[GAO01] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Discrete Mobile Centers," in Proceedings of the 17 th annual symposium on Computational geometry (SCG). ACM Press, 2001, pp. 188–196.

[GAR05] H. F. Garcia, P. R. de L. Gondim. *GAdHoc: um Algoritmo Genético de Particionamento k-Clustering em Redes Ad hoc*. XXXVII Simpósio Brasileiro de Pesquisa Operacional - SBPO, Setembro de 2005.

[GAR99] J.J.Garcia-Luna-Aceves, M. Spohn. Source-tree routing in wireless networks. In Proceedings of the 7th International Conference on Network Protocols (ICNP), 1999.

[GER00] M. Gerla, T. J. Kwon, G. Pei, On demand routing in large ad hoc wireless networks with passive clustering, in Proceedings of IEEE WCNC, September 2000.

[GER01] M. Gerla, X. Hong, T. Kwon, D. Gu, and G. Pei. A mobility framework for ad hoc wireless networks. Proceedings of ACM 2nd International Conference on Mobile Data Management (MDM 2001).January 2001.

[GER95] M. Gerla and T.-C. Tsai. Multicast, mobile, multimedia radio network.ACM Journal of Wireless Networks, 1(3):255-265, 1995.

[GER96] M. Gerla, A. Alwan, R. Bagrodia, N. Bambos,L. Kleinrock, J. Short and J. Villasenor, ''Adaptive mobile multimedia networks,'' IEEE Personal Communications, pp. 34-51, April 1996.

[GER97] M.Gerla adn C.R.Lin, Adaptive clustering for mobile wireless networks.IEEE Journal on selected areas in communications, Vol.15, No.7, Sept1997,pp.1265-1275.

[GER98] M. Gerla, C.Chiang. On-demand multicast in mobile wireless networks. In Proceedings of the IEEE International Conference on Network Protocols (ICNP), 1998.

[GER99a] M.Gerla, A.Iwata, C.-C. Chiang, G.Pei, T.-W.Chen Scalable routing strategies for ad hoc wireless networks, in IEEE Journal on Selected Areas in Communications, Aug 1999, pp.1369-1379.

[GER99b] M. Gerla, X. Hong, G. Pei and C.-C. Chiang, A Group Mobility Model for Ad Hoc Wireless Networks,in Proceedings of ACM/IEEE MSWiM'99, Seattle, WA, Aug. 1999, pp.53-60.

[GHU96] S. Guha and S. Khuller, Approximation algorithms for connected dominating sets, Tech. Report 3660, University of Maryland, College Park, June 1996.

[GHU98] S. Guha, S. Khuller. Approximation algorithms for connected dominating sets. Algorithmica, pp.374-387, 1998.

[GLO00] F. Glover, M.Laguna, R. Martí. Fundamentals of Scatter Search and Path Relinking. In Control and Cybernetics, Volume 29, Number 3, pages 653-684, 2000.

[GLO89] F. Glover, Tabu search Part I. Operations Research Society of America (ORSA), Journal on Computing, 1, pages 109-206, 1989.

[GLO90] F. Glover, Tabu search Part II. Operations Research Society of America (ORSA), Journal on Computing, 2, pages 4-32, 1990.

[GLO96] F.Glover. Tabu search and adaptive memory programing: Advances, applications and challenges.Interfaces in Computer Science and Operations Research. Kluwer,pp. 1–75, 1996.

[GLO97] F. Glover, M. Laguna. Tabu Search. Kluwer Academic Publishers,1997.

[GNU06] Gnuplot for Unix/Linux Systems, http://www.gnuplot.info , último acesso em Março de 2006.

[GOL93] D.E.Goldberg, Genetic Algorithms in Search, Optimization and MachineLearning, Addison-Wesley, 1993.

[HAA97] Z.J.Haas and M.R.Pearlman. The performance of query control schemes for the zone routing protocol.in: Proceedings of IEEE INFOCOM'97, pp.1405-1413, Apr 1997.

[HAA97b] Z. Haas. A new routing protocol for reconfigurable wireless networks. In Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC), pp. 562–565, October 1997.

[HAA98] J. Haartsen, “Bluetooth” - the universal radio interface for ad hoc wireless connectivity, Ericsson Reviews, pp. 110-117, 1998.

[HAA99] Z. Haas, B. Liang. Predictive distance-based mobility management for PCS networks. In Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), March 1999.

[HER92] A. Hertz, E. Taillard, D. de Werra. A Tutorial On Tabu Search. Proc. of Giornate di Lavoro AIRO'95, 1992.

[HOL75] J.H. Holland.. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press, 1975.

[HOU01] T.C. Hou, T.-J. Tsai, An access-based clustering protocol for multihop wireless ad hoc networks, IEEE Journal on Selected Areas in Communications, pp. 1201-1210, 2001.

[HWA98] K. Hwang and Z. Xu. Scalable Parallel Computing. Technology, Architecture,Programming. McGraw-Hill, 1998.

[JIA01] L. Jia, R. Rajaraman, and R. Suel, “An Efficient Distributed Algorithm for Constructing Small Dominating Sets,” in Proceedings of the 20 th ACM Symposium on Principles of Distributed Computing (PODC),2001, pp. 33–42.

[JOH96] D.B.Johnson and D.A.Maltz. Dynamic source routing in ad hoc wireless networks, in Mobile Computing, edited by T. Imielinski and H.Korth, Chapter 5, Kluwer Publishing Co., 1996,pp.153-181.

[JOH99] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, M. Degermark. Routing protocols for mobile ad-hoc networks - a comparative performance analysis. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), pp. 195–206, 1999.

[JOH99b] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek,M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. International Conference on Mobile Computing and Networking (MobiCom'99), pp. 195–206, 1999.

[JWU99] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," in Proc. of the 3rd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM), 1999, pp. 7–14.

[KAH78] R. E. Kahn, S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman, Advances in packet radio technology, Proceedings of the IEEE,pp. 1468-1496, 1978.

[KIR83] S.Kirkpatrick,C.D.Gelatt Jr, M.P.Vecchi Optimization by simutated annealing, Science 220,pp.671-680,1983.

[KUH03] F. Kuhn and R. Wattenhofer, "Constant-Time Distributed Dominating Set Approximation," in In Proceedings of 22 nd ACM Int. Symposium on the Principles of Distributed Computing (PODC), 2003, pp. 25–32.

[LAM99] D. Lam, D. C. Cox, J. Widom. Teletraffic modeling for personal communication services, in IEEE Communications Magazine, pp.79-87, October 1999.

[MAR97] J. G. Markdoulidakis, G. L. Lyberopoulos, D. F. Tsirkas, E. D. Sykas, Mobility modeling in third-generation mobile telecommunication systems, in IEEE Personal Communications, pp. 41-56, August 1997.

[MER99] P.Merz, B.Freisleben. A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem. In Proceedings of the 1999 International Congress of Evolutionary Computation (CEC'99), IEEE Press, pages 2063-2070, 1999.

[MET53] N.Metropolis,A.W.Rosenbluth,M.H.Rosenbluth,A.H.Teller,E.Teller. Equation of state calculatins by fast computing machines, The Journal of Chemical Physics 21,pp.1087-1092, 1953.

[MIC96] Z. Michalewicz,. Genetic Algorithms + data structures = evolution programs, 3.ed.,Ed.Springer, New York, USA, 1996.

[MIL04] P. Mills, E. Tsang, Q. Zhang and J. Ford. A survey of AI-based meta-heuristics for dealing with local optima in local search. Report Number CSM-416, University of Essex, ISSN 1744-8050, September 2004

[MIL89] G. Miller, P. Todd, S. Hegde. Designing Neural Networks using Genetic Algorithms, Proceedings of the Third Internatioanl Conference on Genetic Algorithms, 1989, pp. 379-384.

[MIT97] M. Mitchell, An introduction to genetic algorithms. Cambridge: Mit Press, p. 207, 1997.

[MLA97] N.Mladenovic,P.Hansen. Variable Neighborhood Search. In Computers in Operations Research, 24, pages 1097-1100, 1997.

[MON94] D.C. Montgomery, G.C. Runger. Applied Statistics and Probability for Engineers, John Wiley and Sons, 2ed, 1994.

[MOS89] P. Moscato. On evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards memetic algorithms, C3P Report 826, Caltech Concurrent Computation Program, Caltech, California, U.S.A, 1989.

[MOS93] P.Moscato. An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search. In Annals of Operations Research, 41, pages 85-121, 1993.

[PAP01] S. Papavassiliou, B. An. A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks, International Journal of Network Management, pp. 387-395, November 2001.

[PER99] C.Perkins and E. Royer. Ad hoc on-demand distance vector routing, in Proceedingsof the 2nd IEEE Workshop on Mobile Computer Systems and Applications, pages 90-100, Feb 1999.

[RAD94] N.J.Radclife, P.D.Surry. Formal Memetic Algorithms, In T.Fogarty,editor, Evolutionary Computing: AISB Workshop, volume 865 of Lecture Notes in Computer Science, pp. 1-16, Springer-Verlag, Berlin, 1994.

[RAM98] R. Ramanathan and M. Steenstrup. Hierarchically-organized multihopmobile networks for quality-of-service support,Mobile Networks andApplications, Vol.3, No.2, Aug 1998.

[REE93] C. Reeves Modern Heuristic Techniques for Combinatorial Problems,John Wiley and Sons Inc., ISBN 0-470-22079-1, pp.320, 1993.

[REE98] C. R. Reeves, T. Yamada. Genetic Algorithms, Path Relinking and the Flowshop Sequencing Problem. In Evolutionary Computation Journal 6, pp. 45-60, 1998.

[RES03] M.G.C. Resende, C.C. Ribeiro. Greedy Randomized Adaptive Search Procedures. In Handbook of Metaheuristics, Kluwer Academic Publishers, pages 219-249, 2003.

[RES03b] M.G.C.Resende,C.C.Ribeiro.GRASP and path-relinking: Recent advances and applications. AT&T Labs Research Technical Report, April, 2003.

[RES95] M.G.C. Resende, T.A.Feo. Greedy Randomized Adaptive Search Procedures. In Journal of Global Optimization, vol. 6, pages 109-133, 1995.

[RFC1771] Y. Rekhter, , T.Li. Requests for Coments 1771 – RFC 1771 - A Border Gateway Protocol 4 (BGP-4). T.J. Watson Research Center, IBM Corp.March 1995.

[RFC2501] S.Corsor and J.Marker. Mobile ad hoc networking (MANET). RFC 2501,Janeiro de 1999.

[SAN01] M. Sanchez, P. Manzoni. Anejos: A java based simulator for ad-hoc networks. Future Generation Computer Systems, pp. :573–583, 2001.

[SEX99] R. Sexton, R. Dorsey, J. Johnson. Optimization of Neural Networks: A Comparative Analysis of the Genetic Algorithm and Simulated Annealing, European Journal of Operational Research, 1999.

[SOU06] M.J.F. Souza. Departamento de Computação - Universidade Federal de Ouro Preto www.decom.ufop.br/prof/marcone/ - último acesso em Março de 2006

[STA01] W.Stallings. Wireless Communications & Networks. Prentice Hall. 1 edição,ISBN 0130408646, August 2001.

[STU99] T. Stützle. Iterated Local Search for the Quadratic Assignment Problem. Research Report AIDA-99-03, Department of Computer Schience, Darmstadt University of Technology, Germany, 1999.

[SWA94] J. Szwarcfiter, L.Markenzon. "Estruturas de Dados e seus Algoritmos". 2° ed., LTC, 1994.

[TRI01] K. Trivedi. Probability and Statistics with Reliability, Queuing, and Computer Science Applications, John Wiley and Sons, ISBN 0-471-33341-7 New York, 2001.

[UMB06] Umbrello UML Modeller versão KDE para Linux, disponível em http://uml.sourceforge.net. Último acesso em 20 de Maio de 2006.

[VOU97] C.Voudouris. Guided Local Search for Combinatorial Optimisation Problems, Ph.D. thesis. Department of Computer Science, University of Essex, U.K,1997.

[WES01] D. West, Introduction to Graph Theory, Second edition, Prentice Hall, Upper Saddle River,N.J., 2001.

[WUL99] J. Wu and H. Li, On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks, Proceedings of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 7-14 , Aug. 1999.

[ZHE04] Q. Zheng, X. Hong and S. Ray. Recent advances in mobility modeling for mobile ad hoc network research. Proceedings of the 42nd annual Southeast regional conference, Alabama, 2004.

[ZON97] M.M.Zonoozi and P. Dassanayake. User Mobility modeling and characterization of mobility patterns. IEEE Journal on Selected d Áreas in Communications, 15(7):1239-1252, September 1997.

# APÊNDICES

# A – CONVERGÊNCIA DE SOLUÇÕES: GADHOC, SADHOC, TSADHOC

## A.1. INTRODUÇÃO

Este apêndice apresenta alguns gráficos mostrando a convergência para soluções quase-ótimas das técnicas propostas no presente estudo, em diferentes execuções dos modelos durante a fase de experimentação.

A proposta desta galeria é caracterizar, por amostragem, o funcionamento de cada técnica dentro de suas peculiaridades: resultado final e movimentos de piora.

O resultado final é obtido pela minimização da Função Objetivo ao longo do número de iterações, ou seja, a convergência de $F_0$ para uma solução quase-ótima.

Uma diferença entre técnicas baseadas em Metaheurísticas de Heurísticas é a aceitação de movimentos de piora. Deve ocorrer quando se quer escapar de mínimos locais indesejados, na esperança de encontrar $F_0(iter+k)$, $F_0(iter+k) \leqslant F_0(iter), k>0$ . Para tanto, é necessário um bom ajuste de parâmetros de cada técnica. Isso é realizado experimentalmente.

As ilustrações foram plotadas em *Gnuplot*, versão Unix/Linux [GNU06] para visualização da rede, a partir das informações armazenadas em *logs* de cada execução.

## A.2. SIMULAÇÕES

Nem todos os experimentos deste Apêndice foram considerados para análise no Capítulo 6 por apresentarem comportamento indesejado ou não esperado. Basicamente, por duas ocorrências: quando o algoritmo não é capaz de minimizar $F_0$, escapando de um mínimo desejado e quando entra em ciclos. Este último acontece na tentativa de escapar de um mínimo local em *iter*, com uma seqüência indejada $F_0(iter), F'_0(iter+1)$, $F_0(iter+2), F'_0(iter+3), \ldots, F_0(iter+k-1), F'_0(iter+k)$ , com $F_0 < F'_0$ , $k>0$ . Estes resultados indesejados foram obtidos, em geral, durante a fase de ajuste e aferição de

parâmetros de cada técnica e guardados para efeito ilustrativo.

Os gráficos são apenas uma amostragem das inúmeras simulações executadas neste trabalho. Não se compromete em definir parâmetros particulares das MH em estudo, caso a caso.

### A.2.1. GAdHoc

Os gráficos a seguir mostram a convergência de $F_0$ para o GAdHoc. Por ser um algoritmo populacional, foi adotado como referência $\overline{F_0(iter)}$ .

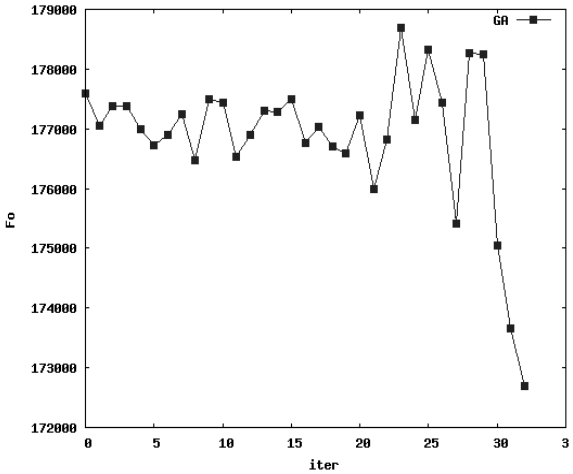


*Figura 83: Exemplo de Convergência de Fo para o GAdHoc*

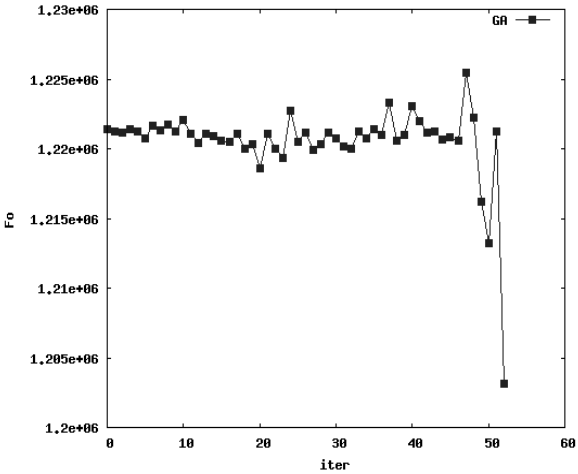


*Figura 84: Exemplo de Convergência de Fo para o GAdHoc*

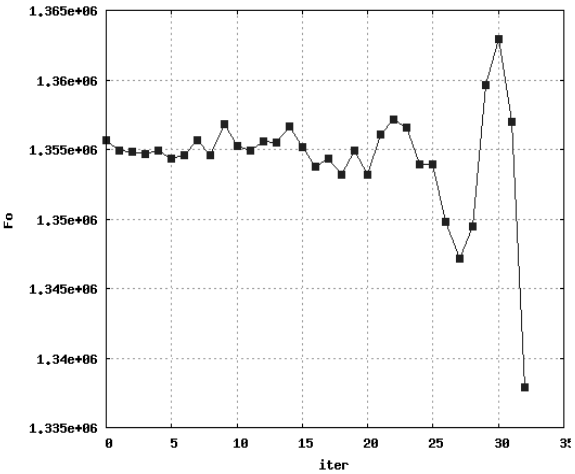


*Figura 86: Exemplo de Convergência de Fo para o GAdHoc*

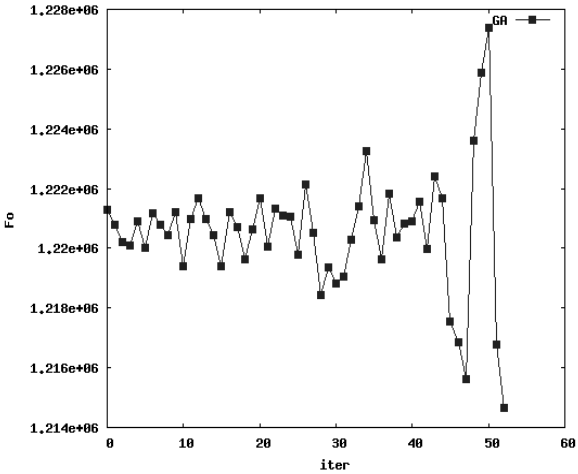


*Figura 85: Exemplo de Convergência de Fo para o GAdHoc*

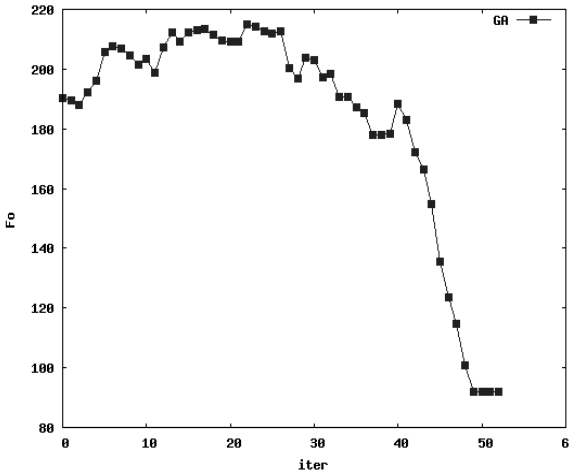


*Figura 87: Exemplo de Convergência de Fo para o GAdHoc*

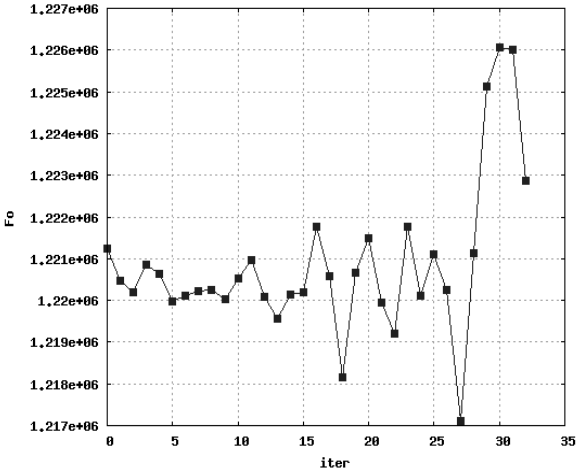


*Figura 88: Exemplo de Convergência indesejada de Fo para o GAdHoc*

### A.2.2. SAdHoc

Os gráficos a seguir mostram a convergência de $F_0$ e aceitação de movimentos de piora para o SAdHoc em algumas simulações do processo de experimentação do Capítulo 6. Em preto, temos $F_{s0}(iter)$ e em branco, $F_s(iter)$.

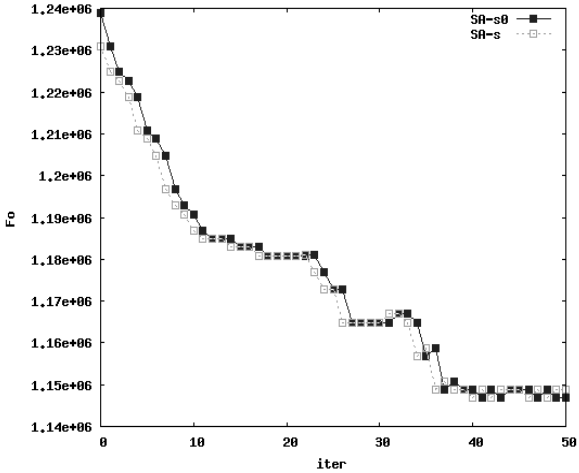


*Figura 90: Exemplo de Convergência de Fo para o SAdHoc*

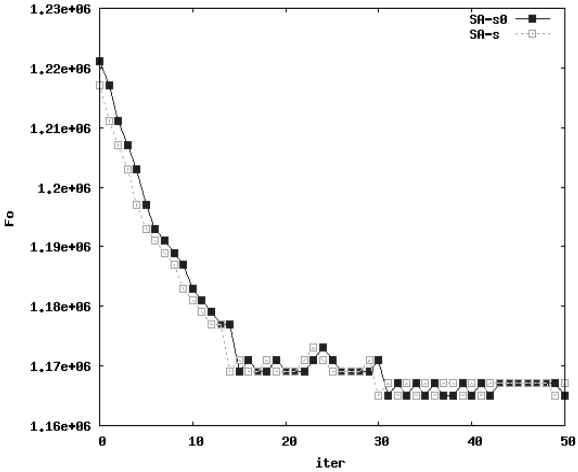


*Figura 89: Exemplo de Convergência de Fo para o SAdHoc*

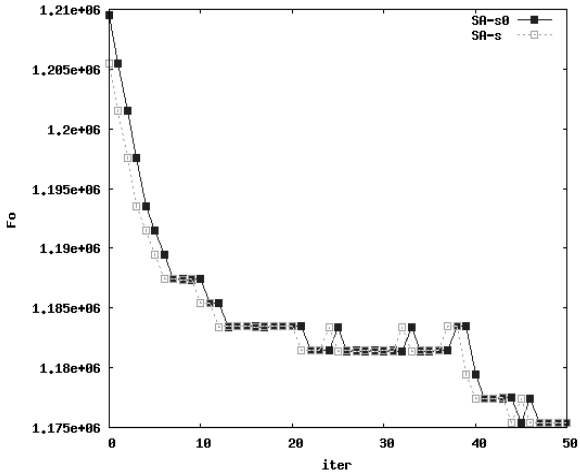


*Figura 91: Exemplo de Convergência de Fo para o SAdHoc*

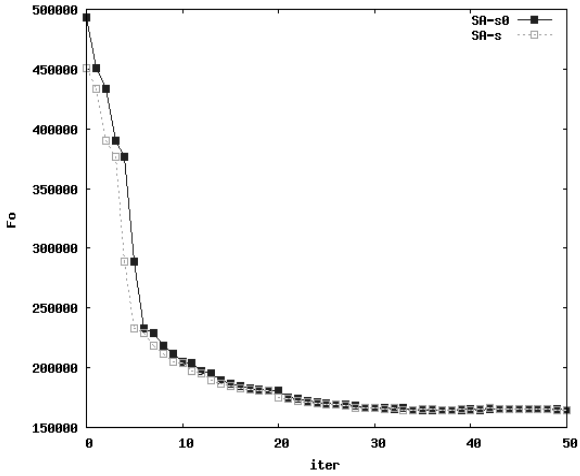


*Figura 92: Exemplo de Convergência de Fo para o SAdHoc*

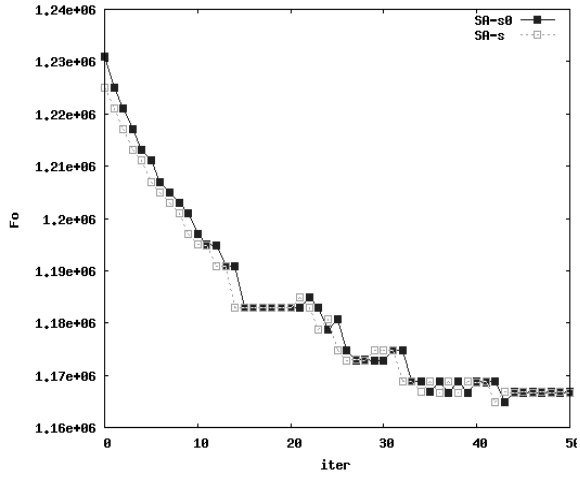


*Figura 93: Exemplo de Convergência indesejada de Fo para o SAdHoc*

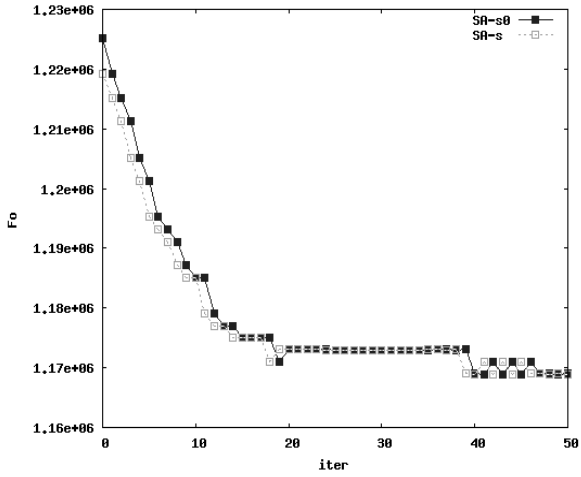


*Figura 94: Exemplo de Convergência indesejada de Fo para o SAdHoc*

### A.2.3.TSAdHoc

Os gráficos a seguir mostram a convergência de $F_0$ e aceitação de movimentos de piora para o TSAdHoc em algumas simulações do processo de experimentação do Capítulo 6. Em preto, temos $F_{s0}(iter)$ e em branco, $F_s(iter)$.

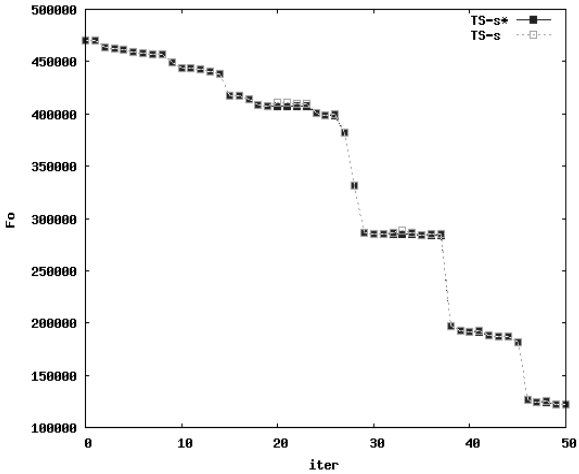


*Figura 95: Exemplo de Convergência de Fo para o TSAdHoc*

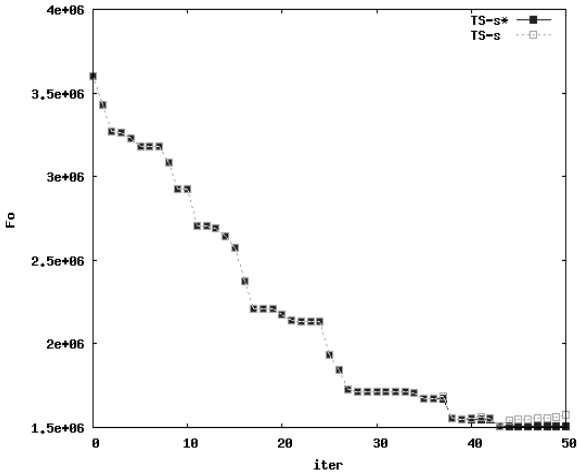


*Figura 96: Exemplo de Convergência de Fo para o TSAdHoc*

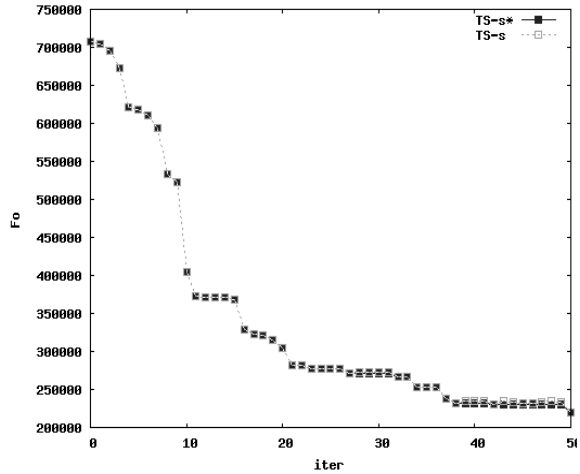


*Figura 97: Exemplo de Convergência de Fo para o TSAdHoc*

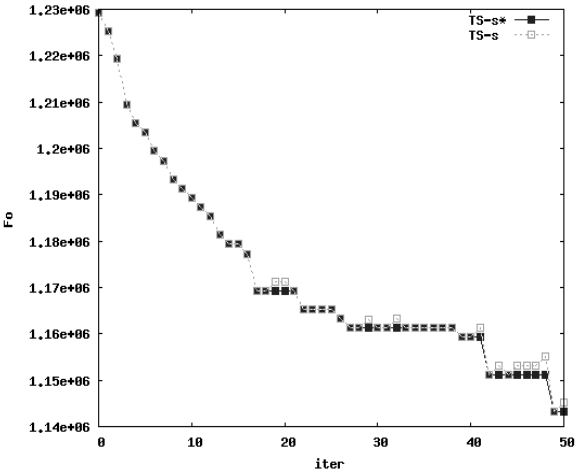


*Figura 98: Exemplo de Convergência de Fo para o TSAdHoc*

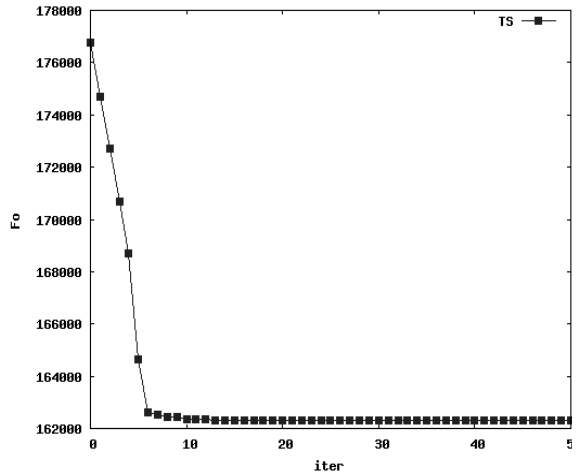


*Figura 99: Exemplo de Convergência indesejada de Fo para o TSAdHoc*

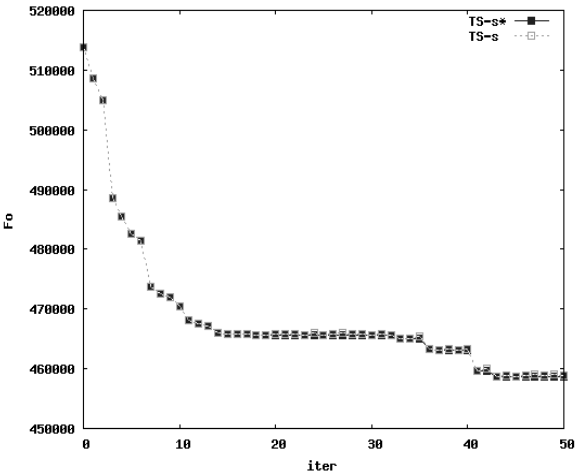


*Figura 100: Exemplo de Convergência indesejada de Fo para o TSAdHoc*

# B – VISUALIZANDO EXECUÇÕES EM JAVA2D

## B.1. CENÁRIO

Este apêndice apresenta um exemplo simplificado de simulações com G=(V,E), $V=\{n_0, n_1, \ldots, n_8\}$ , disposição determinística em terreno 100x100, $T_{exec}$=10, K=3. Para melhor observação de Reafiliações, apenas os nós $n=\{n_1,n_5,n_7\}$ realizam movimento, com: $\overrightarrow{v_n(t)}=\overrightarrow{v_n(t-1)}+\overrightarrow{\Delta v_n(t)}$ , e variações $\overrightarrow{\Delta v_{n1}(t)}=(+\Delta x,0)$ , $\overrightarrow{\Delta v_{n5}(t)}=(0,+\Delta y)$ , $\overrightarrow{\Delta v_{n7}(t)}=(-\Delta x,0)$ . A composição dos três movimentos forma uma espécie de "cata-vento".

As ilustrações em cada *t* foram desenvolvidas em linguagem JAVA (JAVA2D) para visualização 2-D do comportamento da rede, a partir das informações armazenadas em *logs* de cada execução.

## B.2. SIMULAÇÕES

Foram executadas simulações para os seguintes modelos: MHAdHoc, WCA, HDA, para $T_{exec}=10$ e $T_g=1$. Os nós estão identificados por números, circunscrito por $tx_{range}$ . Cada $C_i$ apresenta uma cor distinta e $h(C_i)$ está com envoltória em destaque. As visualizações estão organizadas por instante de tempo $T=[0..10]$ para comparação de comportamento da rede em técnicas distintas, na ordem MHAdHoc, WCA e HDA.

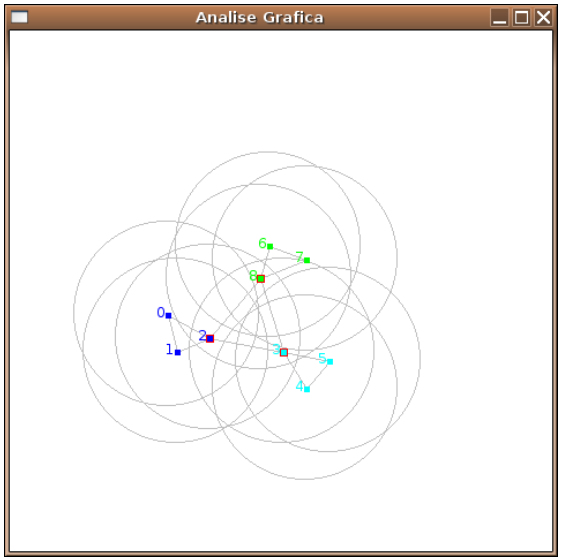


*Figura 101: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=0*

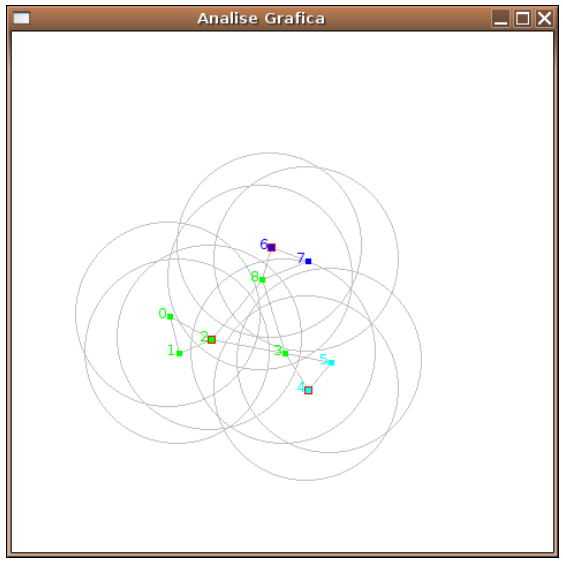


*Figura 102: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=0*

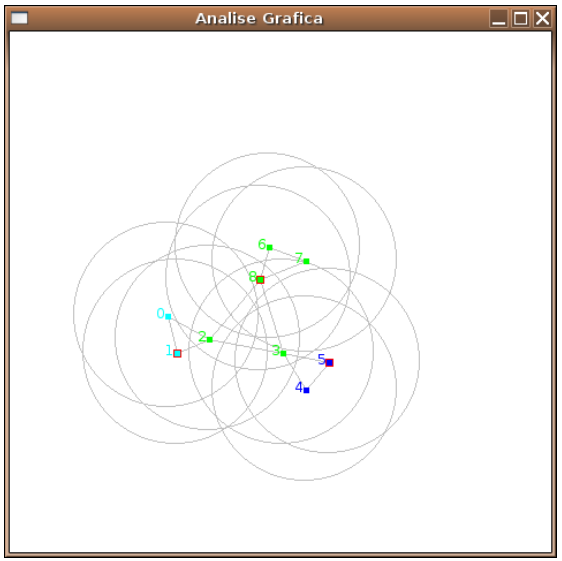


*Figura 103: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=0*

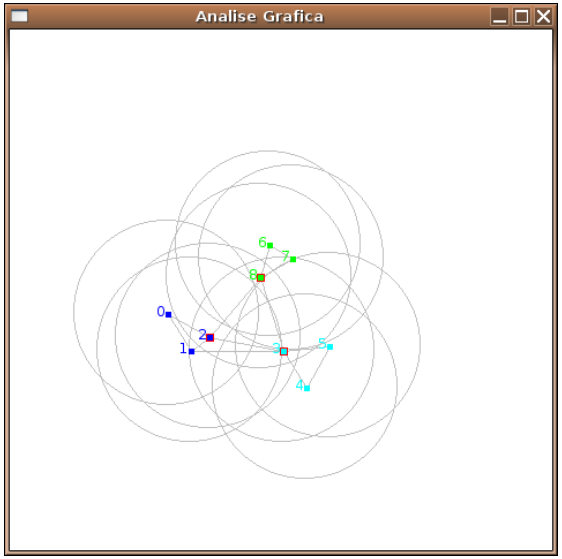


*Figura 104: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=1*

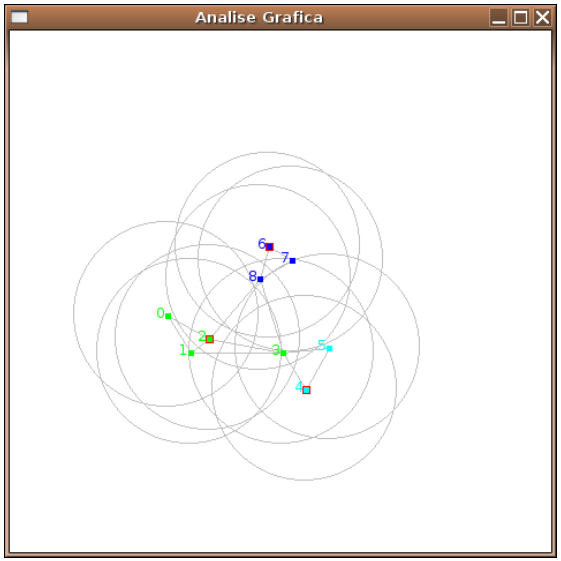


*Figura 106: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=1*

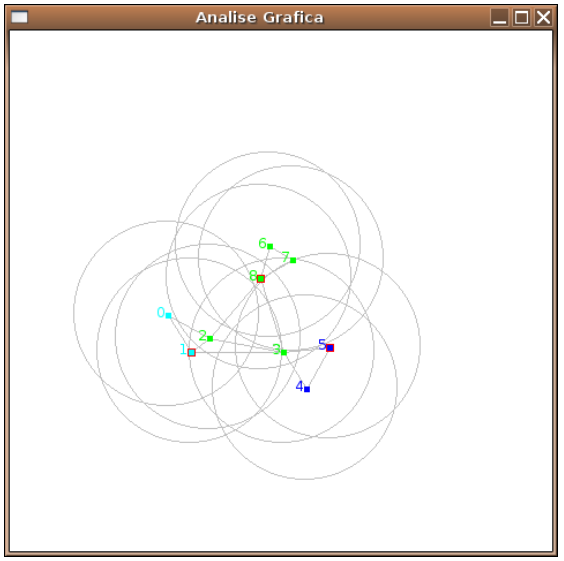


*Figura 105: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=1*

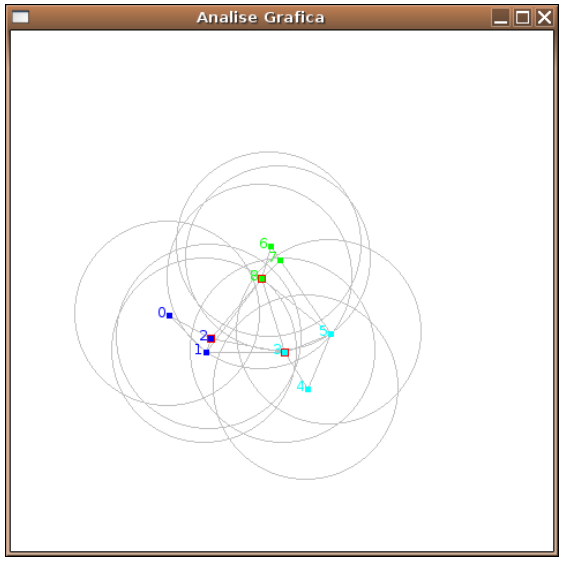


*Figura 107: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=2*

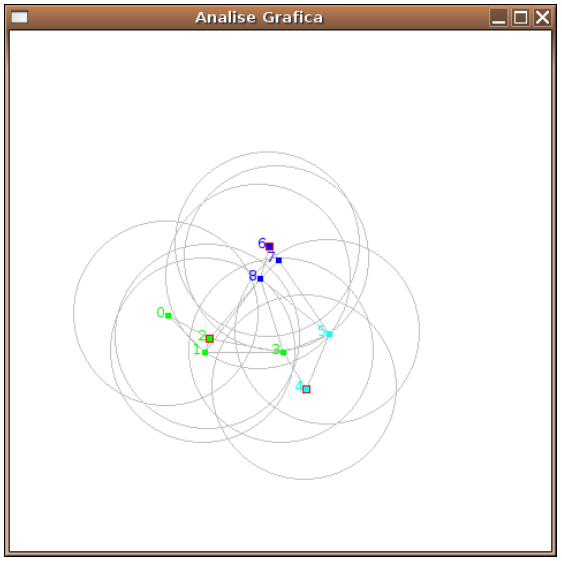


*Figura 108: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=2*

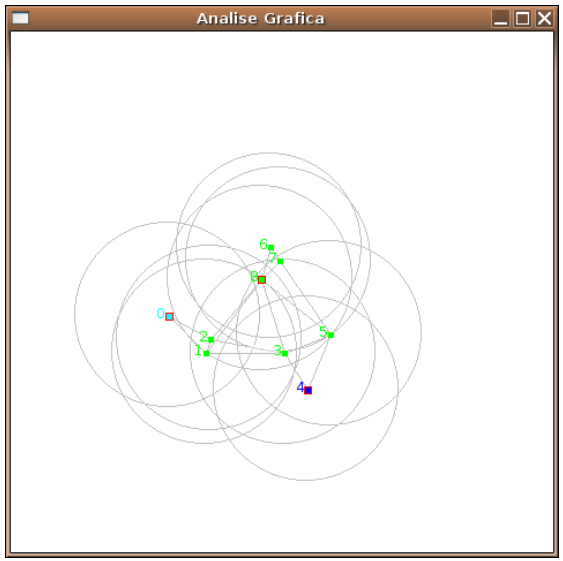


*Figura 109: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=2*

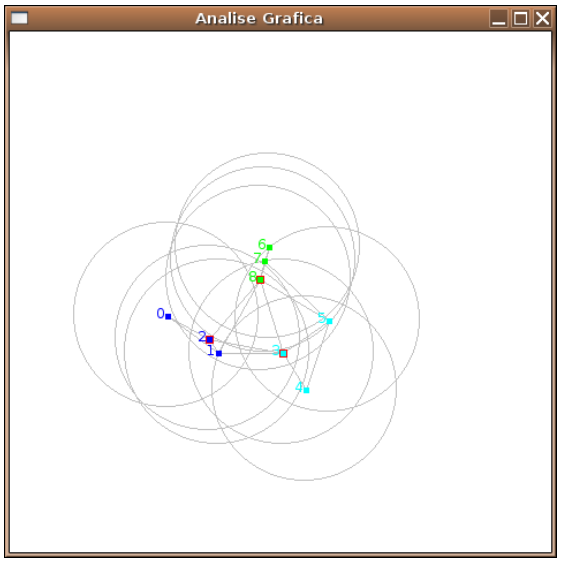


*Figura 110: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=3*

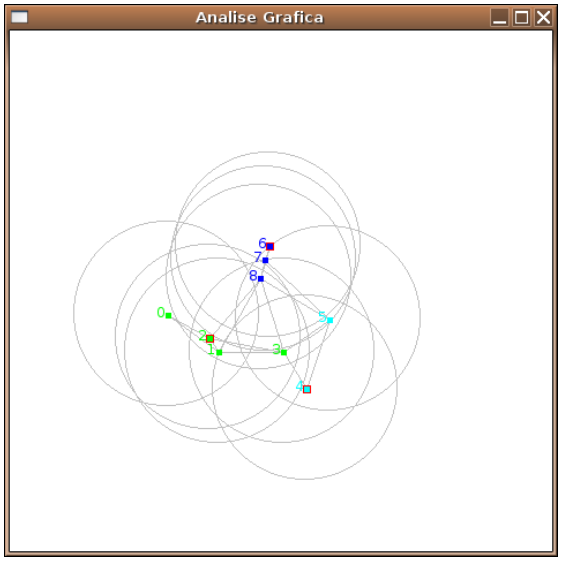


*Figura 111: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=3*

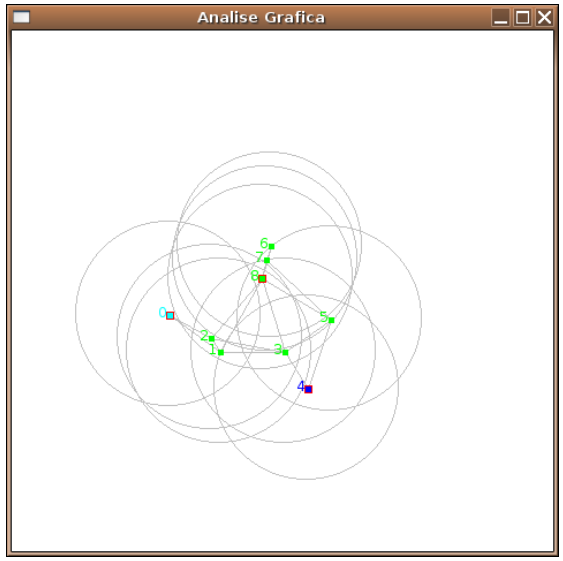


*Figura 112: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=3*

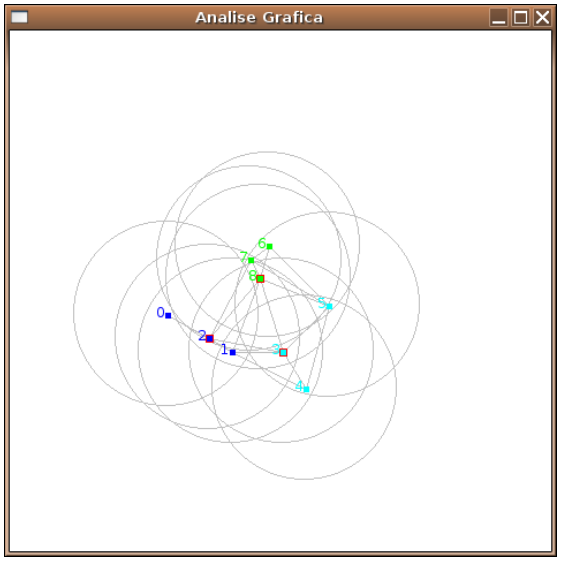


*Figura 113: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=4*

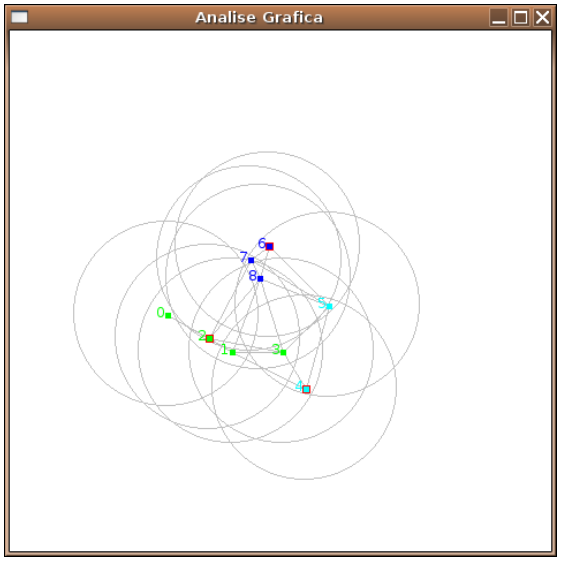


*Figura 114: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=4*

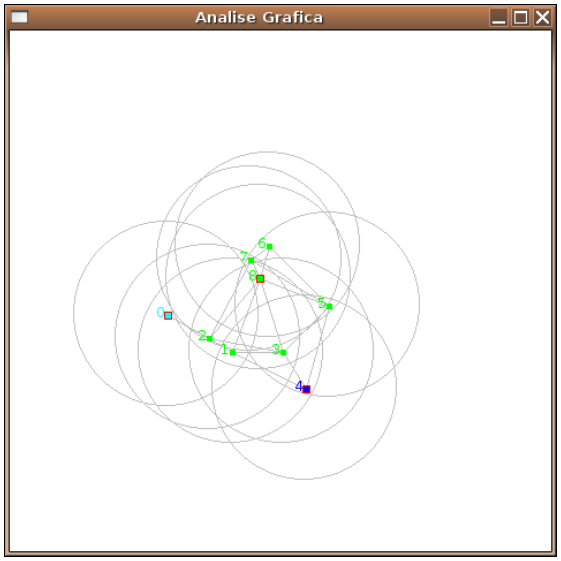


*Figura 115: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=4*

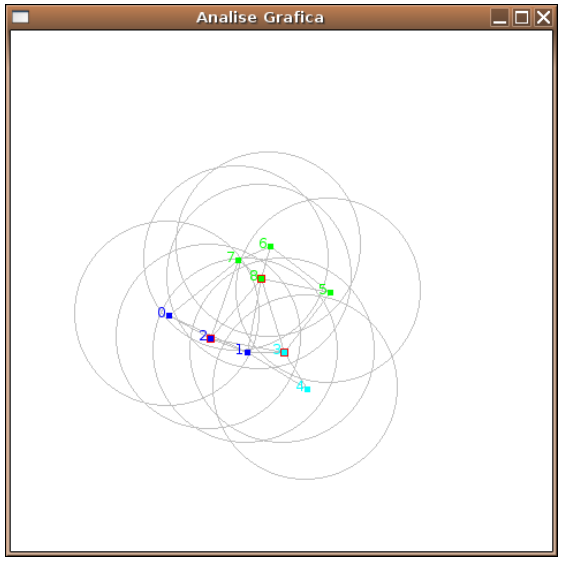


*Figura 116: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=5*

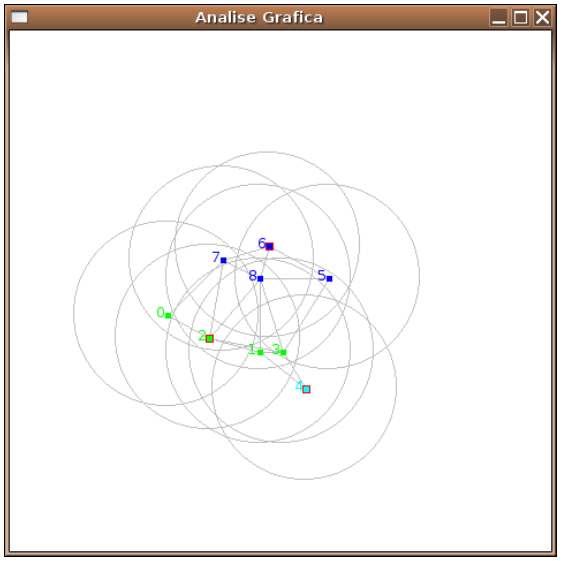


*Figura 118: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=5*

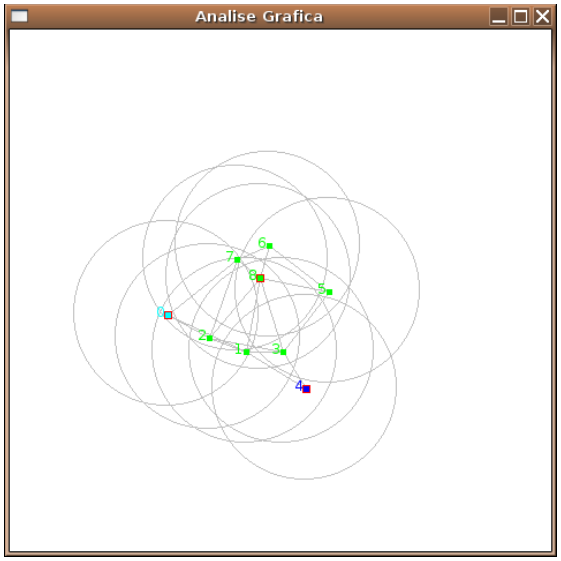


*Figura 117: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=5*

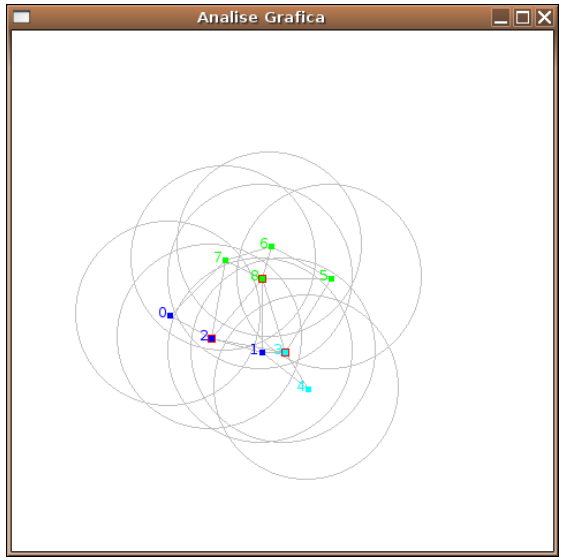


*Figura 119: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=6*

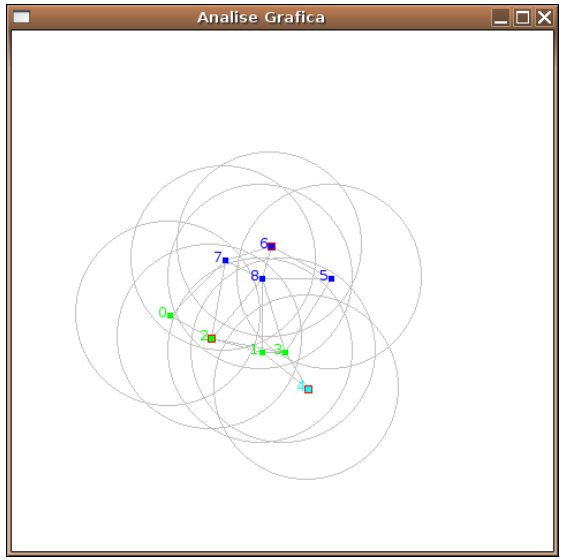


*Figura 120: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=6*

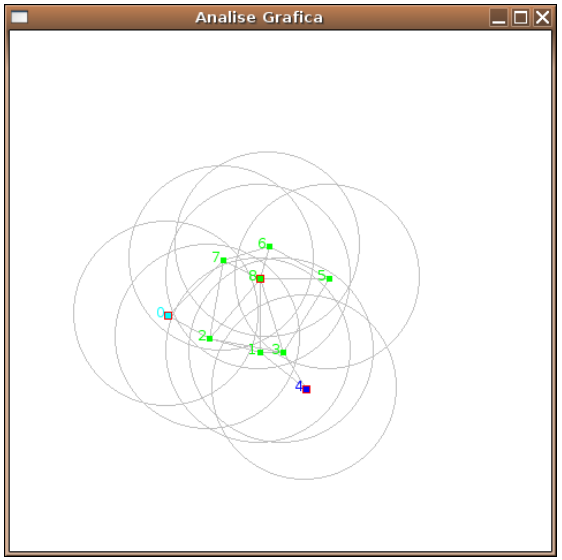


*Figura 121: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=6*

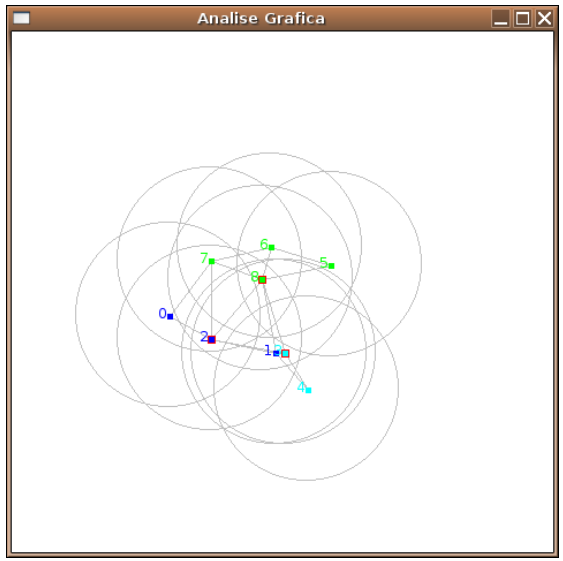


*Figura 122: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=7*

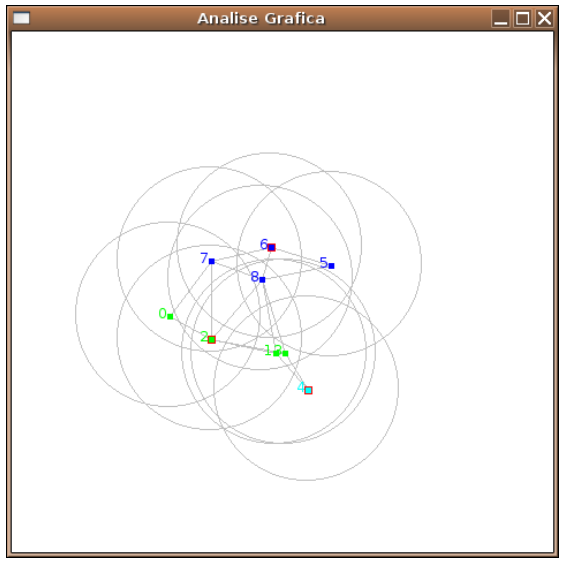


*Figura 123: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=7*

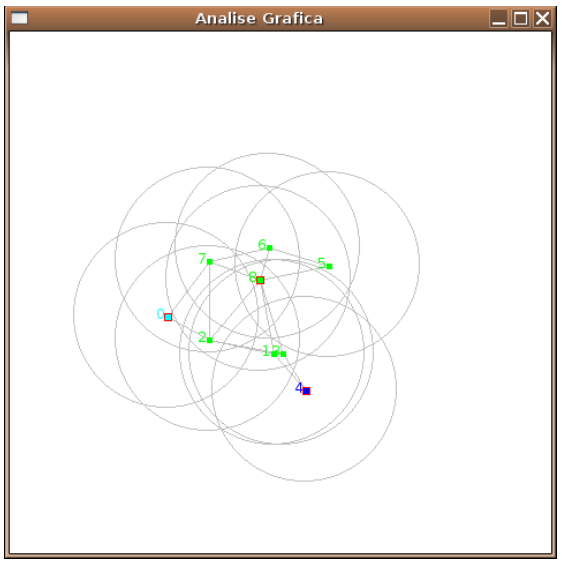


*Figura 124: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=7*

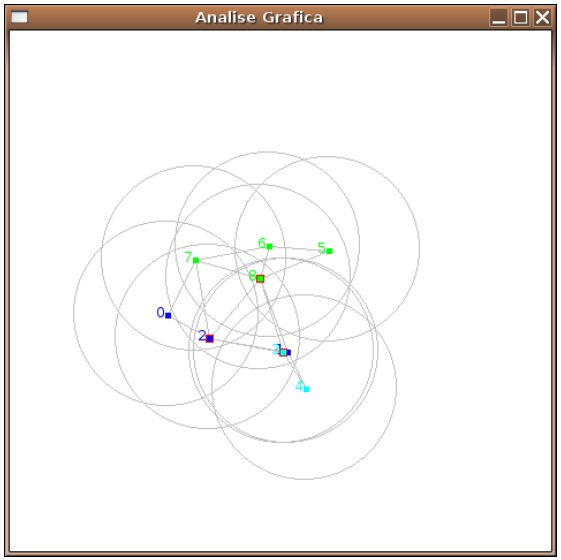


*Figura 127: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=8*

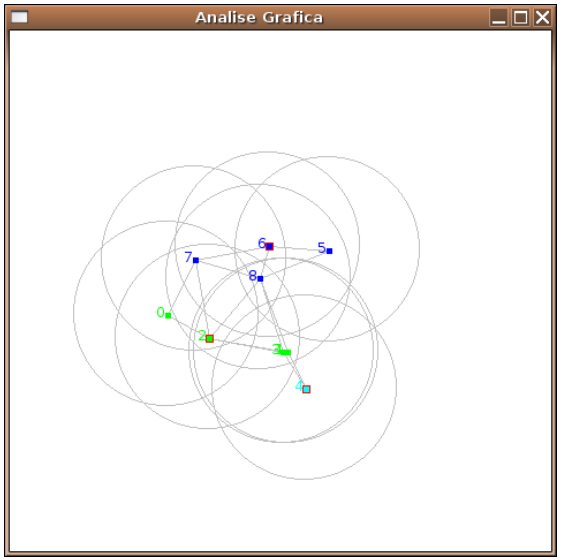


*Figura 125: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=8*

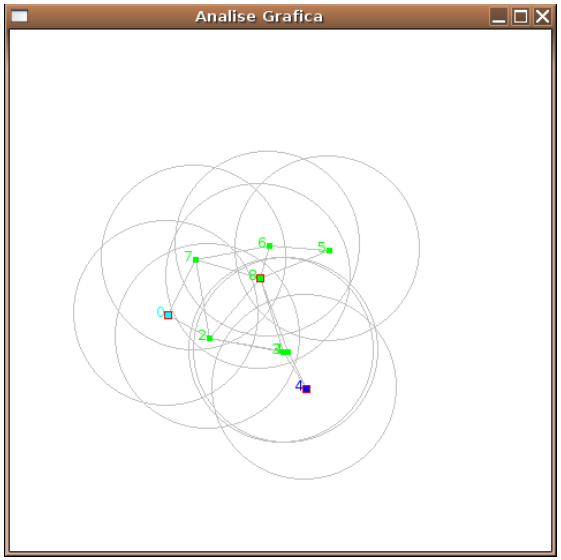


*Figura 126: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=8*

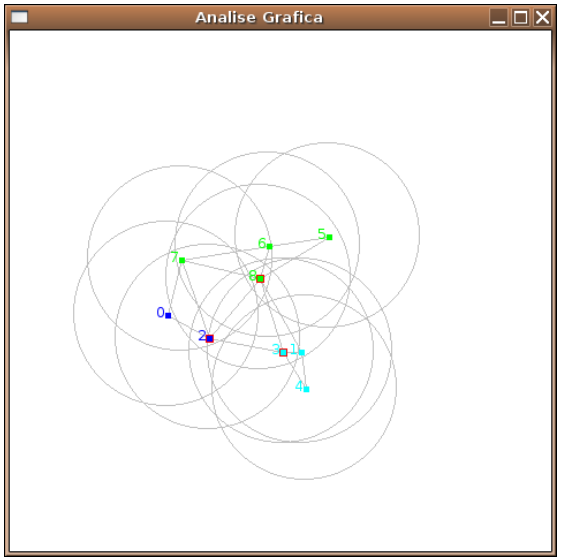


*Figura 129: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=9*

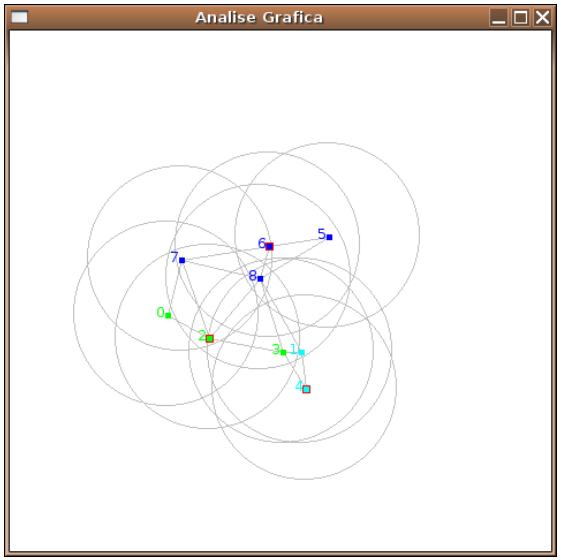


*Figura 128: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=9*

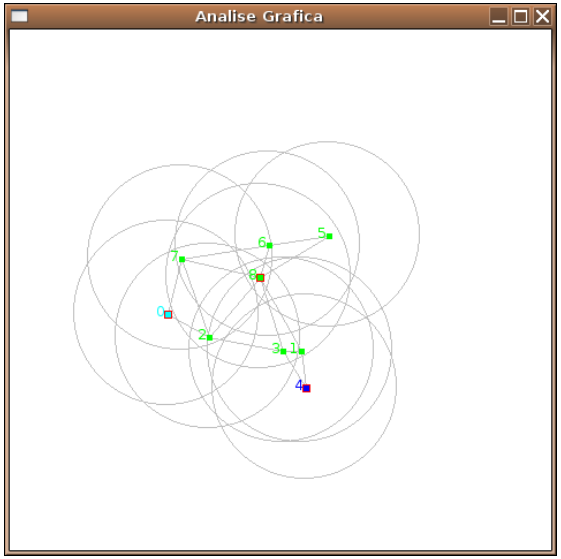


*Figura 130: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=9*

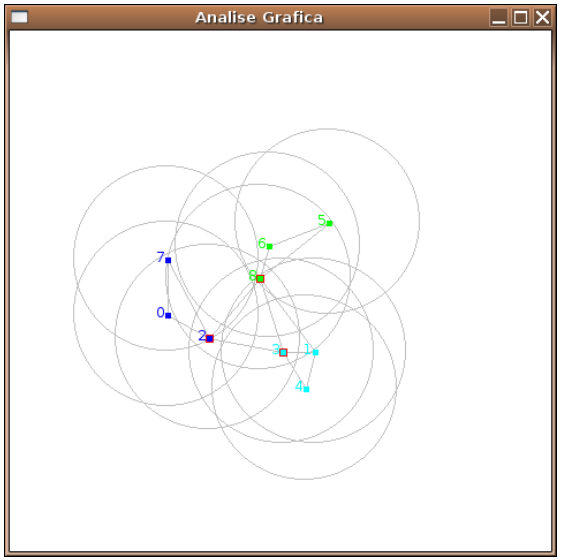


*Figura 131: Simulação para MHAdHoc,G(V,E), N=9,K=3,grafo conexo,determinado em T=10*

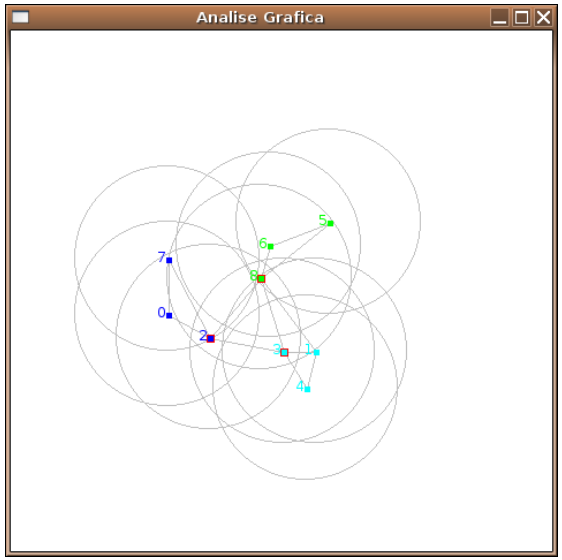


*Figura 132: Simulação para WCA, G(V,E), N=9,K=3,grafo conexo,determinado em T=10*

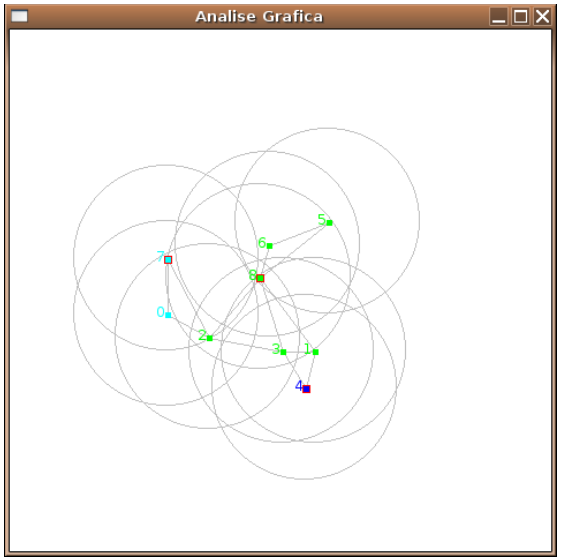


*Figura 133: Simulação para HDA, G(V,E), N=9,K=3,grafo conexo,determinado em T=10*

# C – VISUALIZAÇÃO DE TOPOLOGIAS

## C.1. INTRODUÇÃO

Este apêndice apresenta alguns resultados obtidos no processo de experimentação no Capítulo referente aos Estudos de Caso. Em cada cenário, temos G=(V,E), $V=\{n_0, n_1, \ldots, n_{V-1}\}$ , disposição aleatória uniformemente distribuída em terreno 100x100, sob condição restritiva de formação a partir de um grafo conexo.

As ilustrações foram plotadas em linguagem Java (JAVA2D), dentro do ambiente de desenvolvimento Eclipse, versão Linux para visualização da rede, a partir das informações armazenadas em *logs* de cada execução.

## C.2. GALERIA DE TOPOLOGIAS

Serão mostradas algumas topologias, formadas a partir do modelo proposto durante o processo de experimentação do Capítulo 6. Os nós são identificados por pontos e cada aresta $\overline{n_i, n_j}, n_i, n_j \in V$ representa $\overline{v_i, v_j} \leqslant tx_{range}$ . Os *clusters* estão identificados por cores distintas, os *clusterheads* $h(C_i), \{C_0, C_1, \ldots, C_i, \ldots, C_K\}$ são mostrados em destaque e em vermelho. Os nós destacados com um retângulo envoltório em preto são *dominators* e $tx_{range}$ circunscrito em *n* é omitido. Juntamente com cada solução, apresenta-se também um resultado preliminar ligado à primeira fase do algoritmo Wu-Li [WUL99]. Em um primeiro momento, temos o resultado da primeira fase do algoritmo, com um conjunto redundante de *dominators*. Na segunda fase, temos o refinamento, com a eliminação de redundâncias.

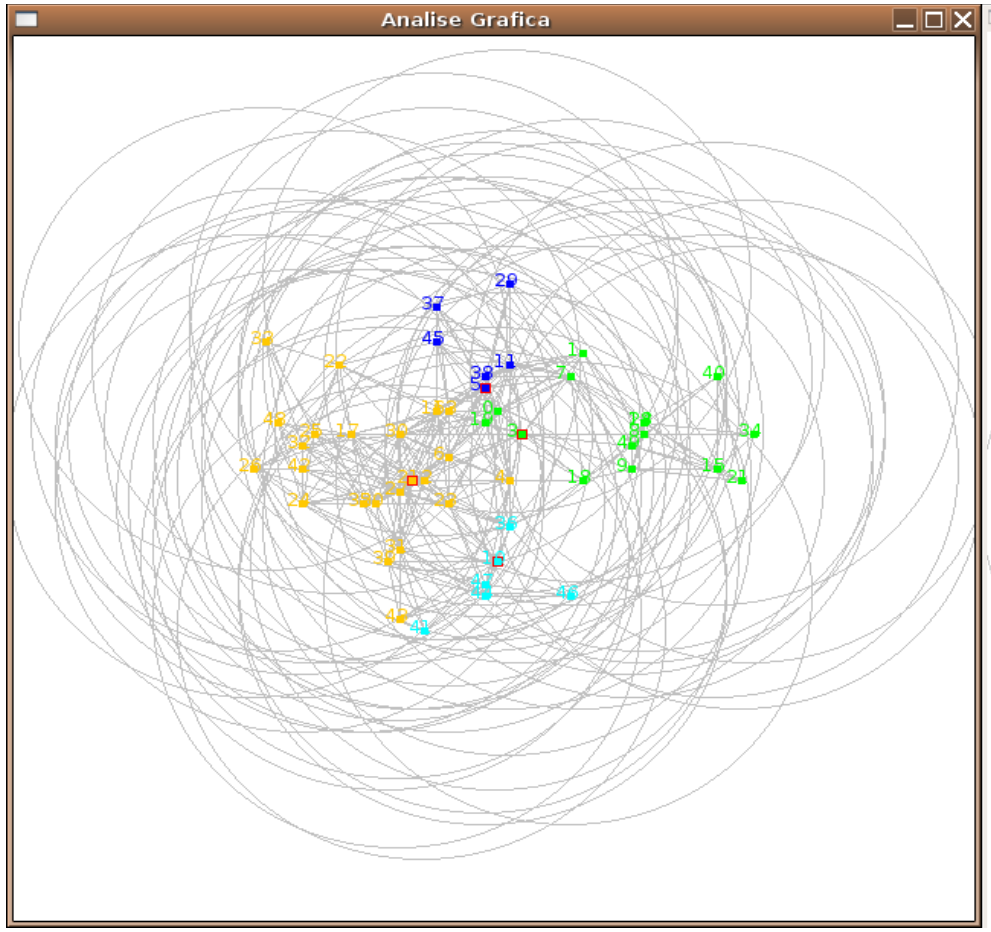


*Figura 134: Exemplo 1: Topologia ao final da execução do MHAdHoc, contendo clusters e clusterheads (N=50 e K=4)*

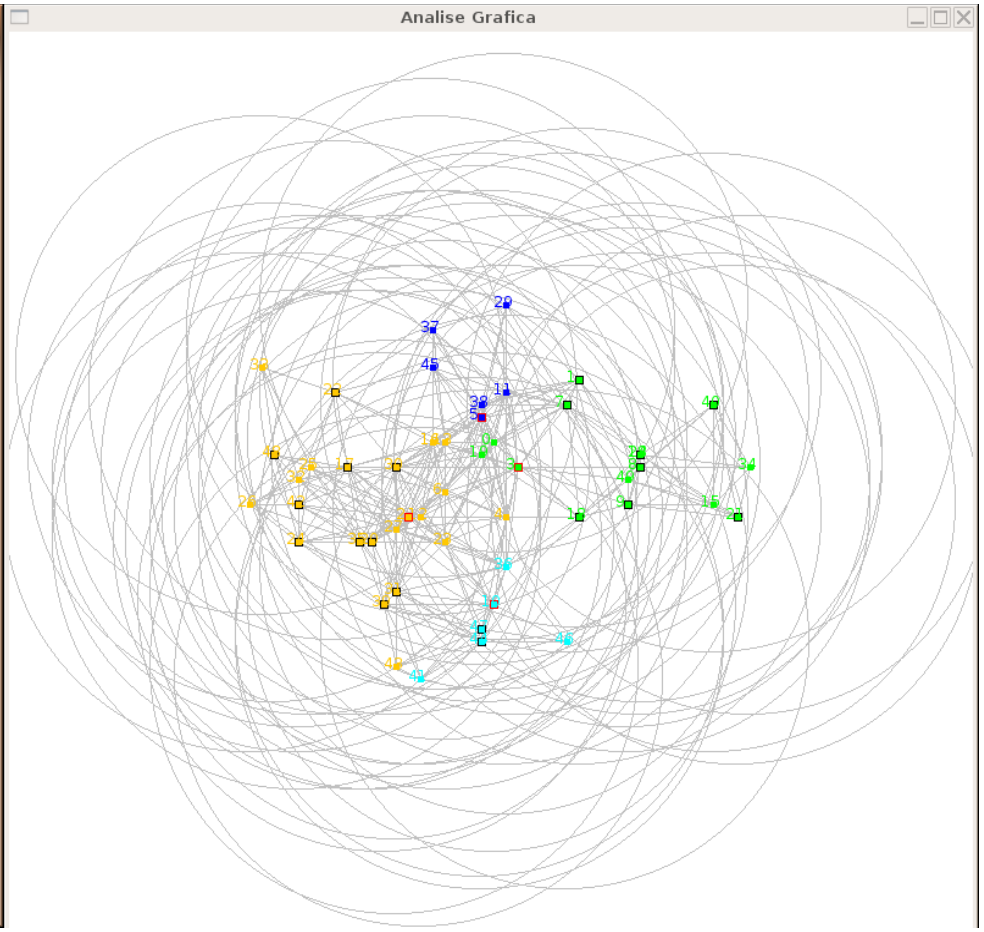


*Figura 135: Exemplo 1: Topologia ao final da primeira etapa do algoritmo Wu-Li, com a determinação de dominators redundantes*

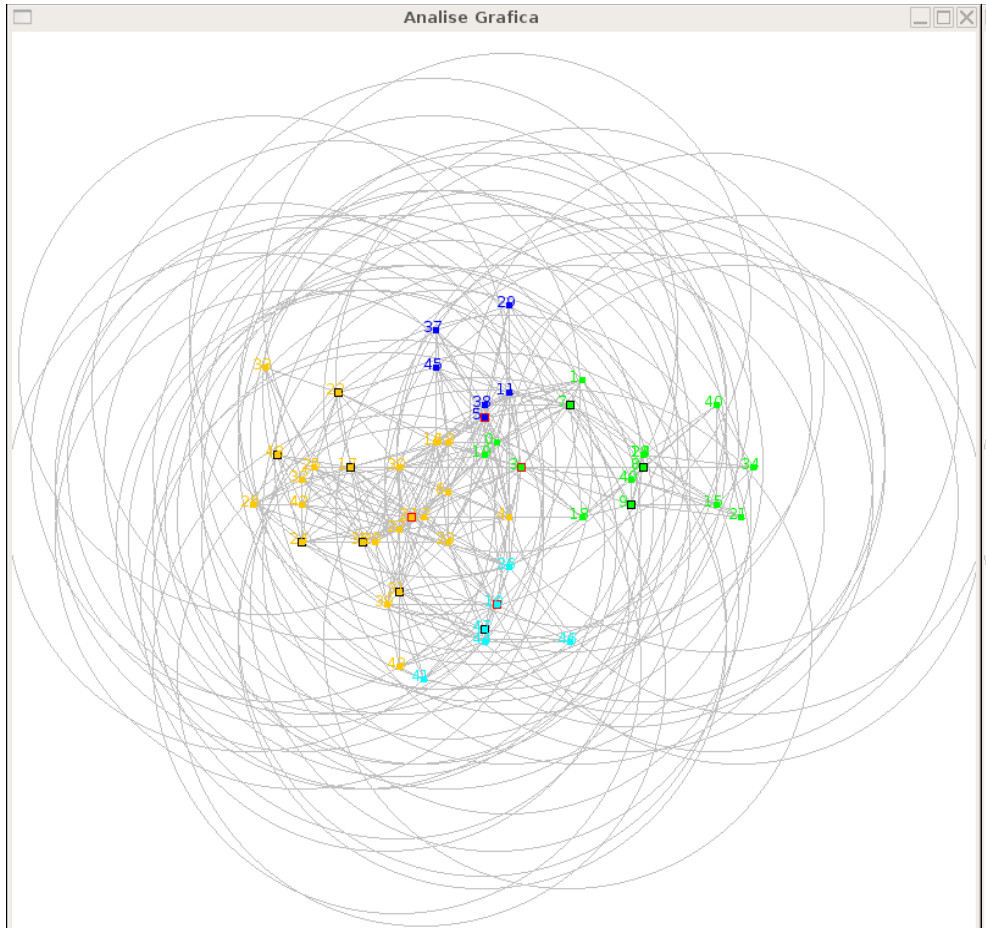


*Figura 136: Exemplo 1: Topologia ao final da segunda etapa do algoritmo Wu-Li, com a determinação de dominators não-redundantes*

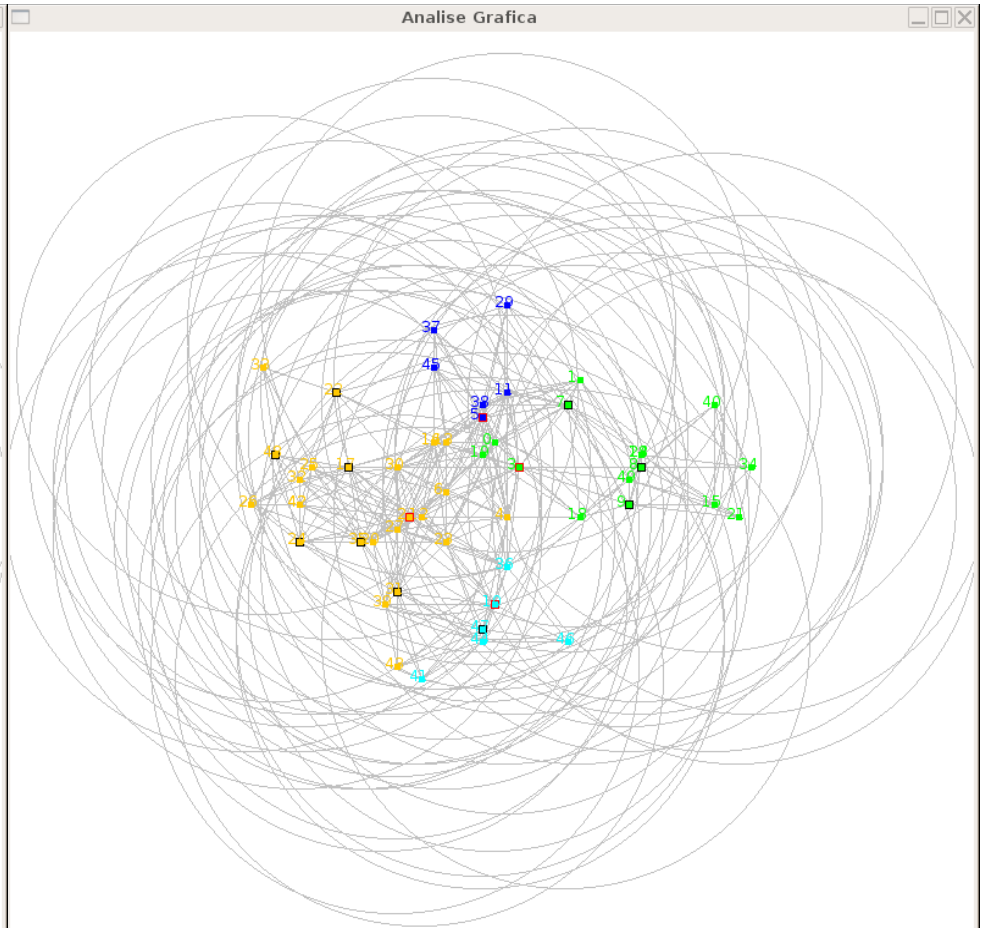


*Figura 137: Exemplo 1: Topologia ao final do ajuste (border fixing)*

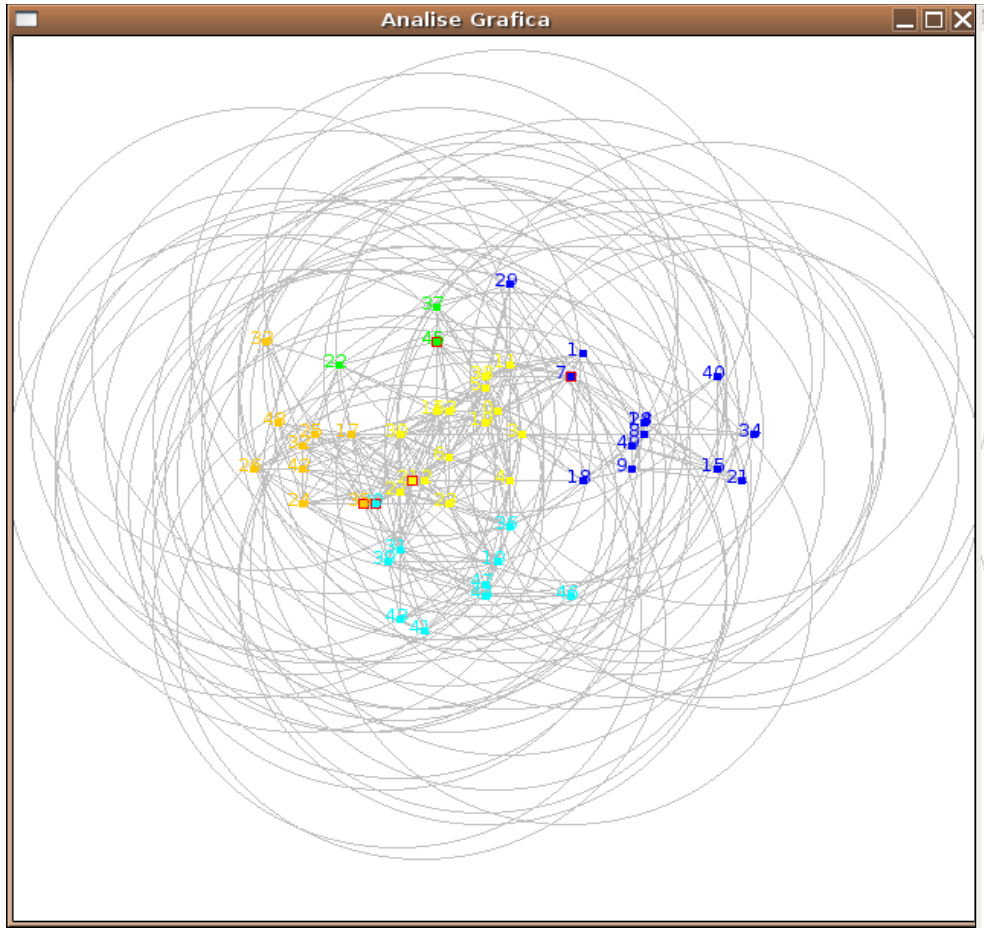


*Figura 138: Exemplo 2: Topologia ao final da execução do MHAdHoc, contendo clusters e clusterheads (N=50 e K=5)*

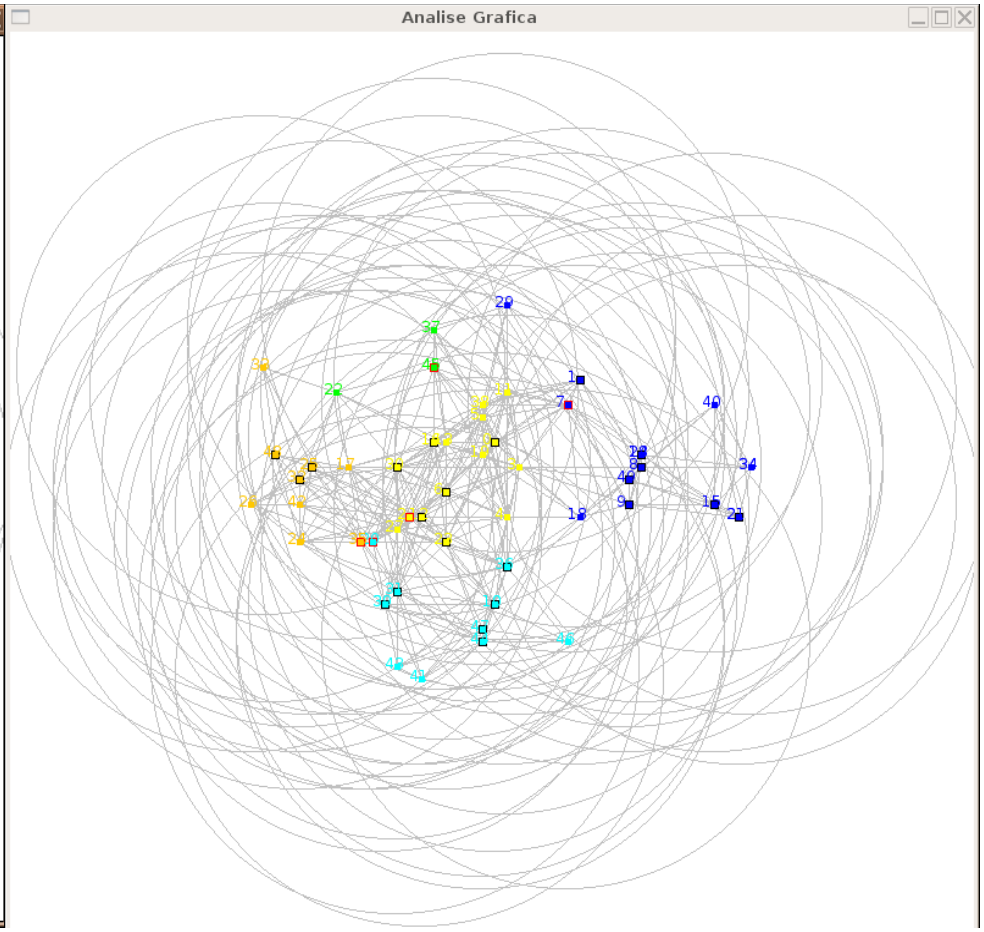


*Figura 139: Exemplo 2: Topologia ao final da primeira etapa do algoritmo Wu-Li, com a determinação de dominators redundantes*

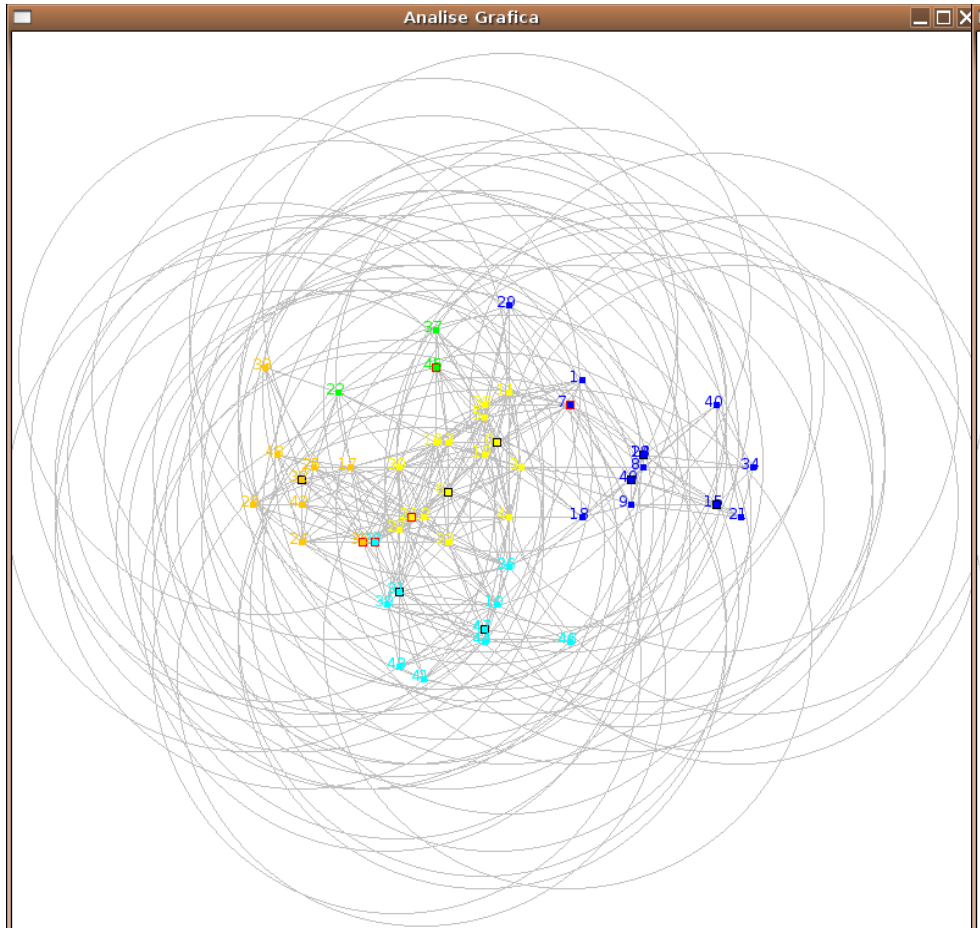


*Ilustração 140: Exemplo 2: Topologia ao final da segunda etapa do algoritmo Wu-Li, com a determinação de dominators não-redundantes*

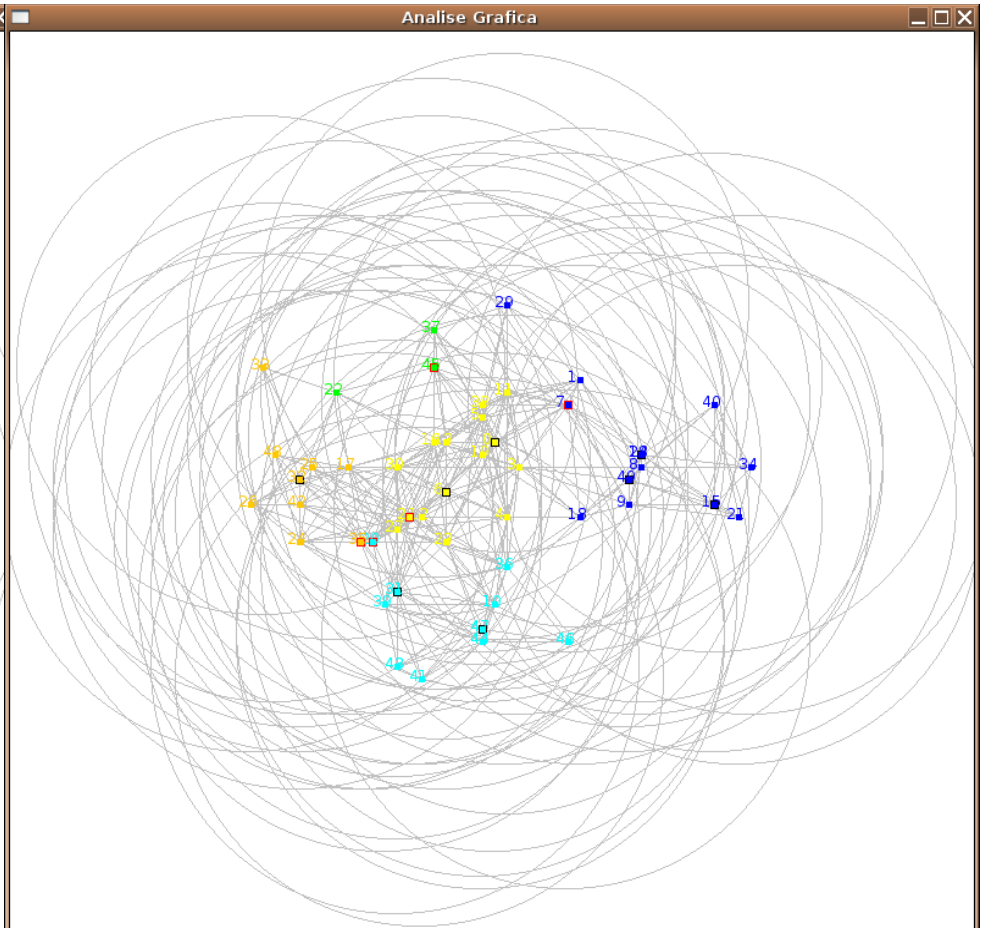


*Ilustração 141: Exemplo 2: Topologia ao final do ajuste (border fixing)*

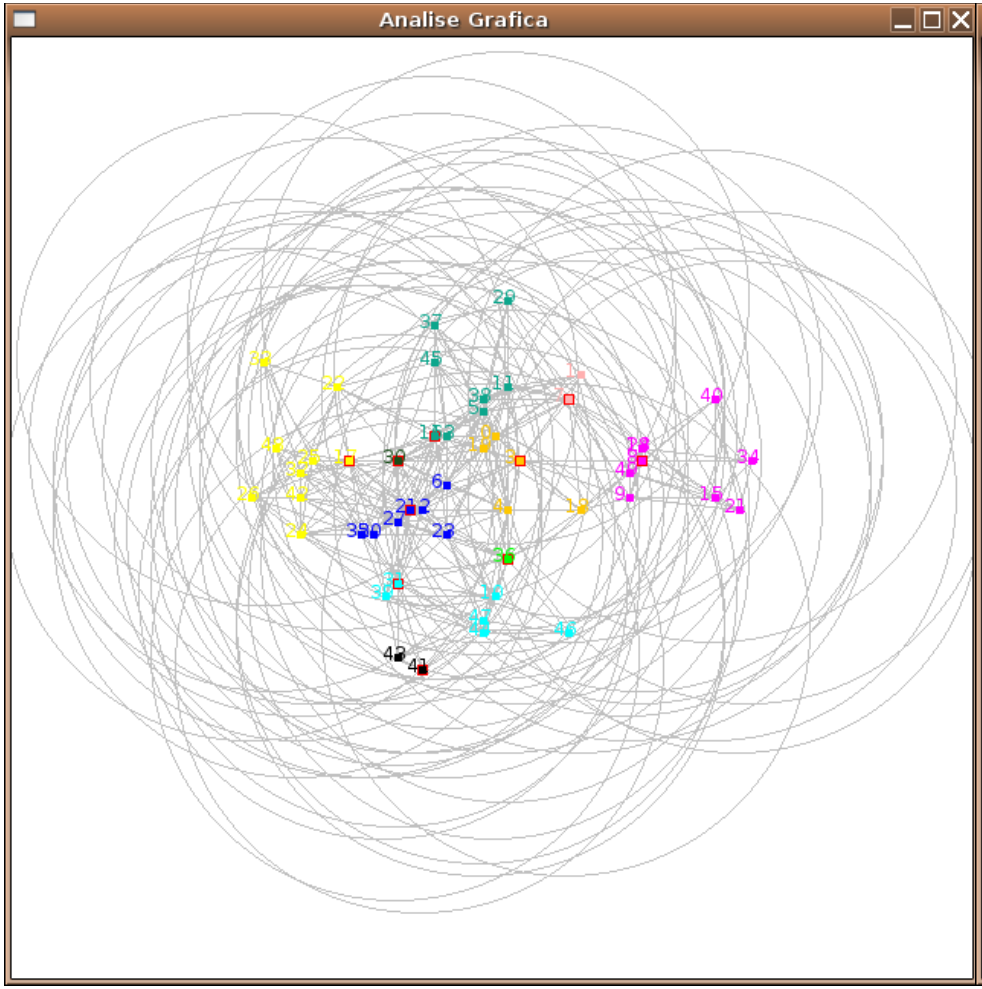


*Figura 142: Exemplo 3: Topologia ao final da execução do MHAdHoc, contendo clusters e clusterheads (N=50 e K=10)*

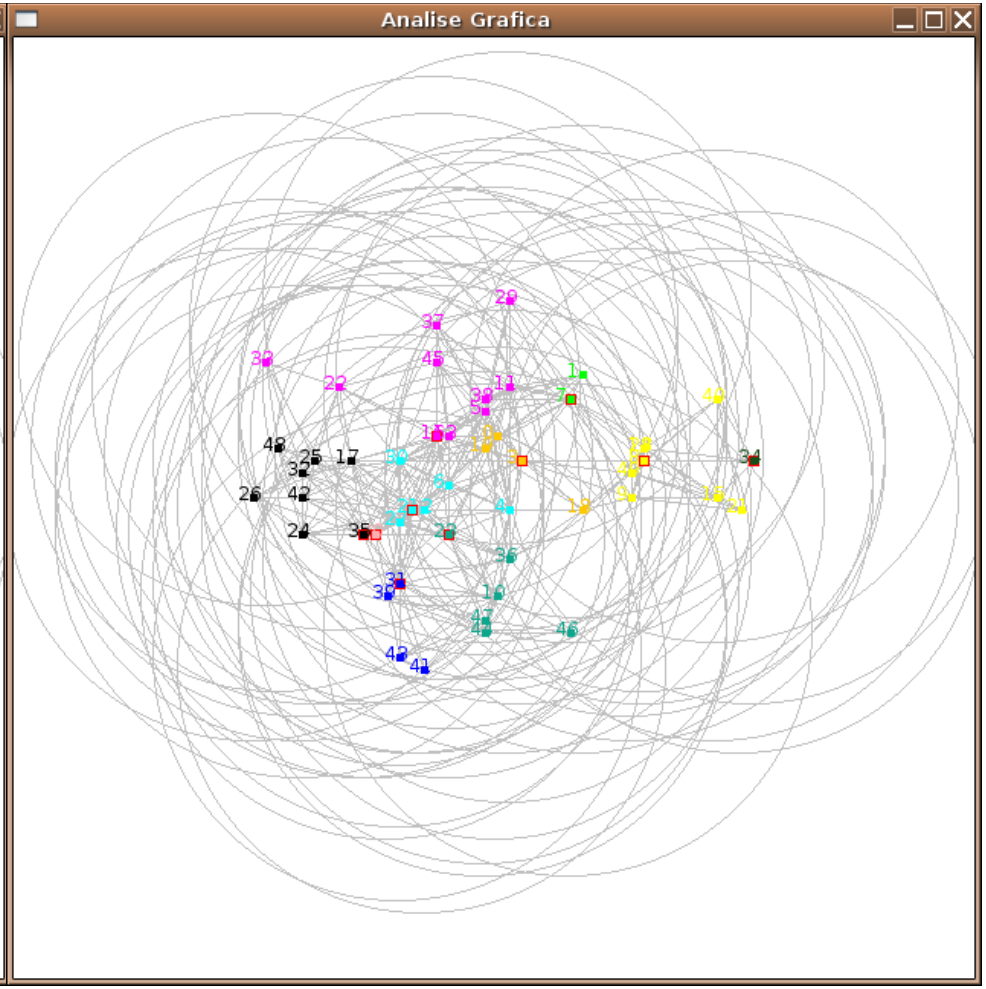


*Figura 143: Exemplo 4: Topologia ao final da execução do MHAdHoc, contendo clusters e clusterheads (N=50 e K=10)*

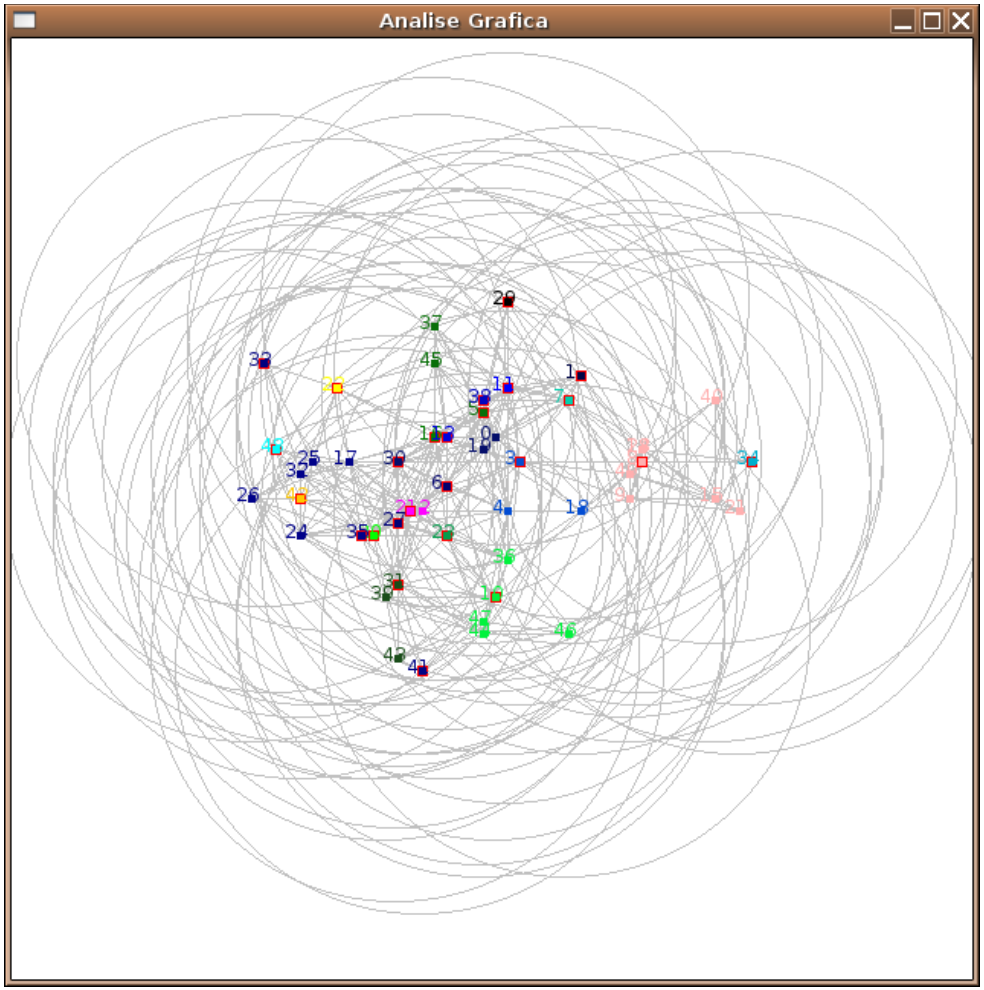


*Ilustração 144: Exemplo 5: Topologia ao final da execução do MHAdHoc, contendo clusters e clusterheads (N=50 e K=25)*

# D – CENÁRIOS DE SIMULAÇÃO

## D.1. INTRODUÇÃO

Este apêndice apresenta alguns cenários de simulação utilizados no processo de experimentação no Capítulo referente aos Estudos de Caso. Em cada cenário, temos G=(V,E), $V=\{n_0, n_1, \ldots, n_{V-1}\}$ , disposição aleatória uniformemente distribuída em terreno 100x100, sob condição restritiva de formação a partir de um grafo conexo.

As ilustrações foram plotadas em *Gnuplot*, versão Unix/Linux [GNU06] para visualização da rede, a partir das informações armazenadas em *logs* de cada execução.

## D.2. GALERIA DE CENÁRIOS

Serão apresentados alguns cenários *G(V,E)* utilizados no Capítulo 6, para diferentes nós, *N=|V|=[20..50]*. Os nós são identificados por pontos e cada aresta $\overline{n_i, n_j}, n_i, n_j \in V$ representa $\overline{v_i, v_j} \leqslant tx_{range}$ . Vide Figuras 145 a 153.

## D.3. IDENTIFICAÇÃO DE CLUSTERHEADS

Serão mostradas algumas topologias, formadas a partir do modelo proposto durante o processo de experimentação do Capítulo 6. Os nós são identificados por pontos e cada aresta $\overline{n_i, n_j}, n_i, n_j \in V$ representa $\overline{v_i, v_j} \leqslant tx_{range}$ . Para efeito de simplificação, apenas $h(C_i), \{C_{0,} C_{1,} \ldots, C_i, \ldots, C_K\}$ é mostrado (em destaque e em vermelho) e $tx_{range}$ circunscrito em *n* é omitido. Vide Figuras 154 a 157.

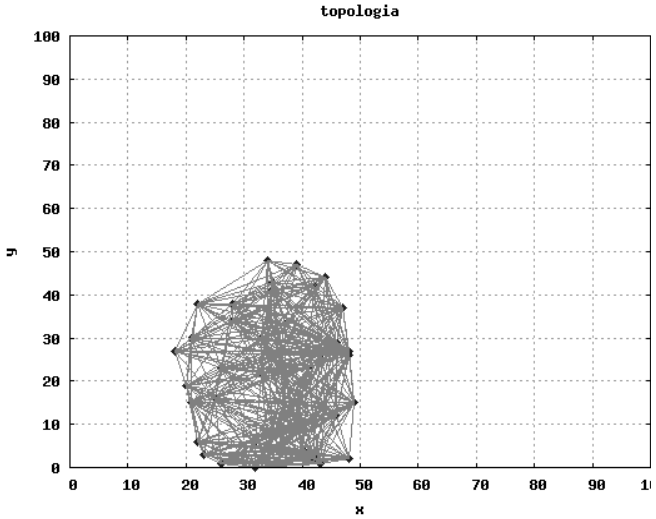


*Figura 145: G(V,E), N=50, conexo, aleatório uniformemente distribuído*

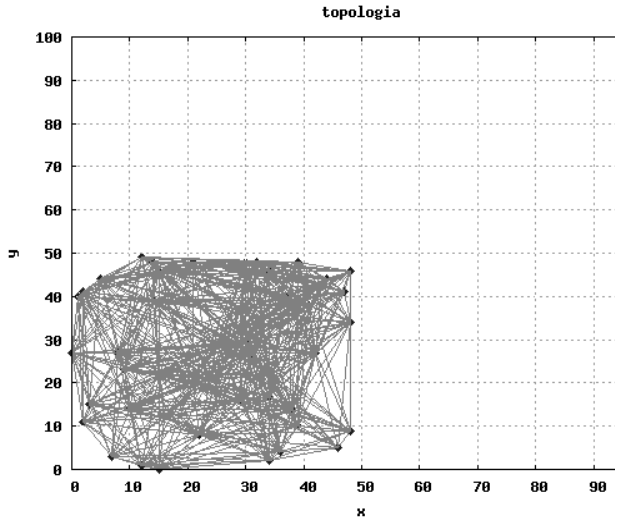


*Figura 146: G(V,E), N=50, conexo, aleatório uniformemente distribuído*

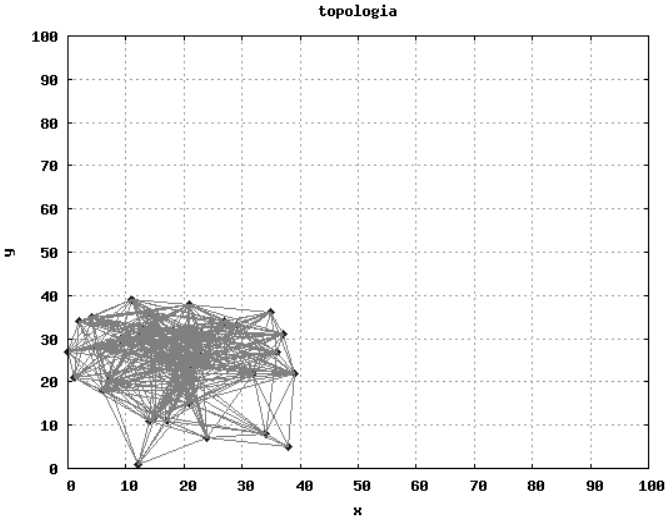


*Figura 147: G(V,E), N=50, conexo, aleatório uniformemente distribuído*

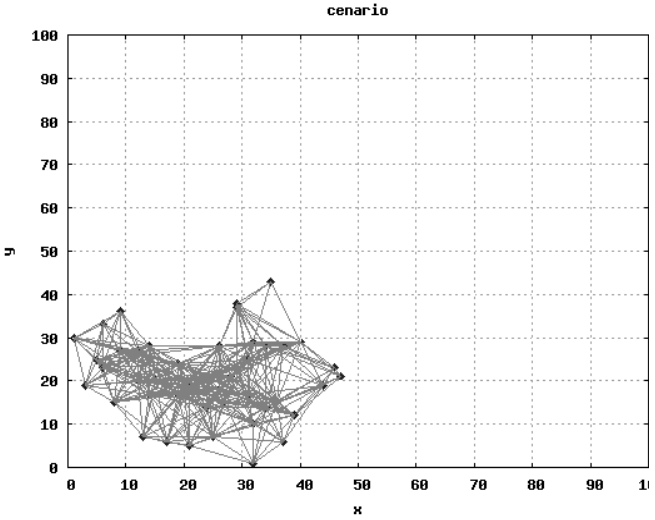


*Figura 148: G(V,E), N=50, conexo, aleatório uniformemente distribuído*

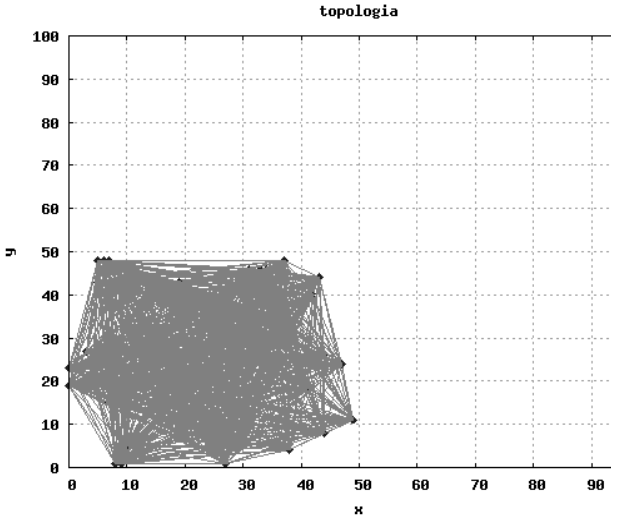


*Figura 150: G(V,E), N=50, conexo, aleatório uniformemente distribuído*

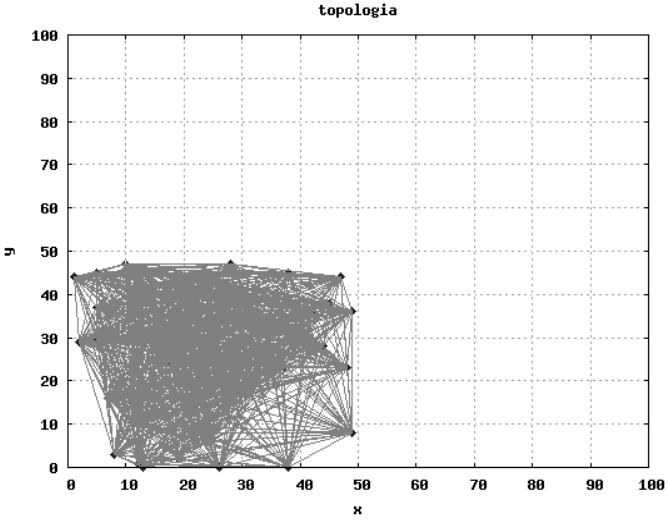


*Figura 149: G(V,E), N=50, conexo, aleatório uniformemente distribuído*

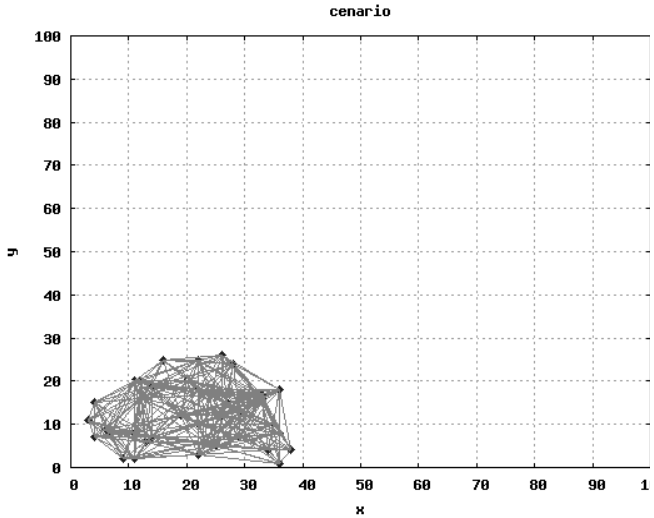


*Figura 151: G(V,E), N=40, conexo, aleatório uniformemente distribuído*

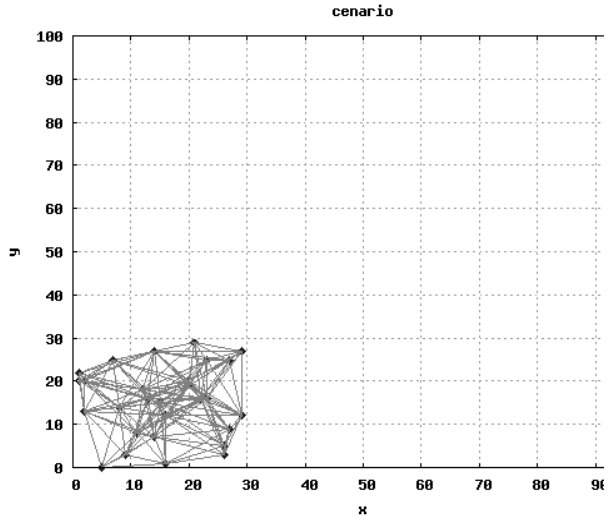


*Figura 152: G(V,E), N=30, conexo, aleatório uniformem distribuído*

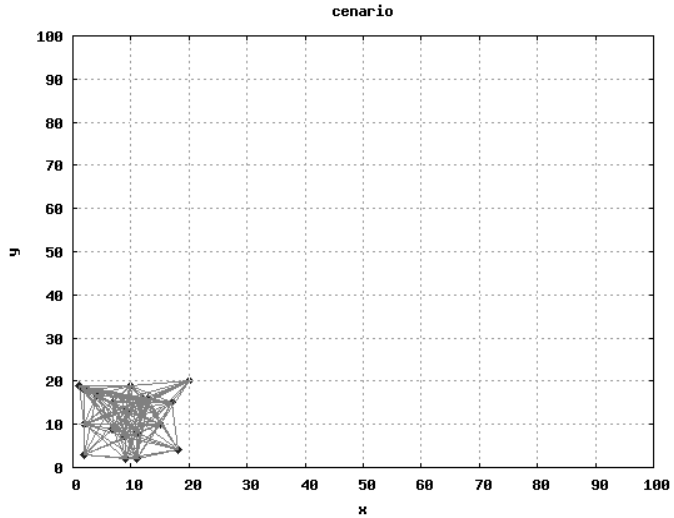


*Figura 153: G(V,E), N=20, conexo, aleatório uniformemente distribuído*

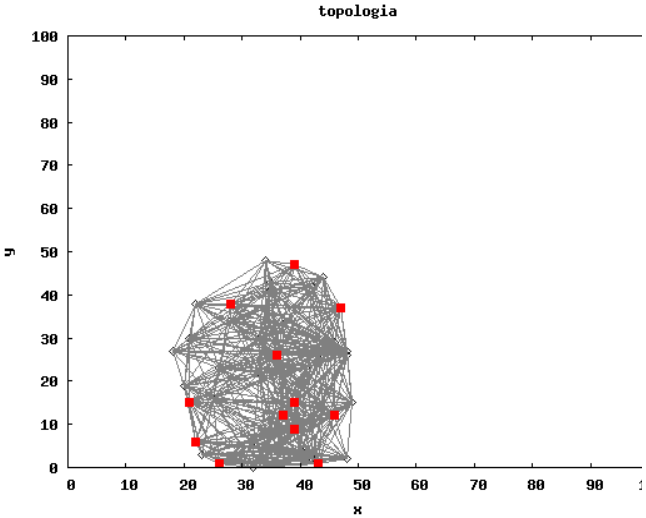


*Figura 155: Uma topologia formada a partir de G(V,E), N=50, conexo, aleatório uniformemente distribuído (apenas clusterheads)*

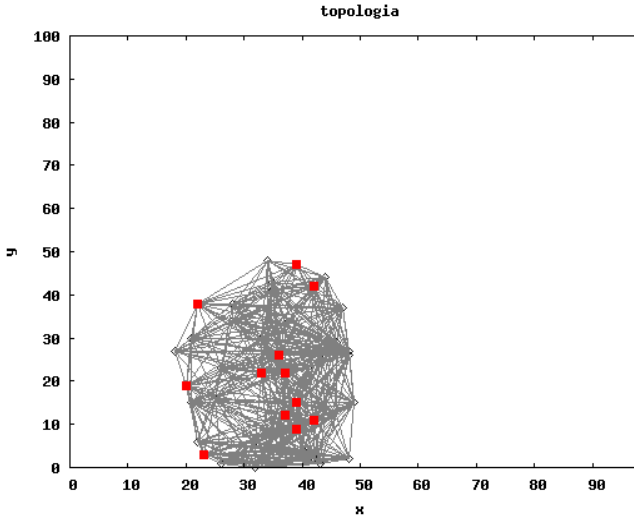


*Figura 156: Uma topologia formada a partir de G(V,E), N=50, conexo, aleatório uniformemente distribuído (apenas clusterheads)*

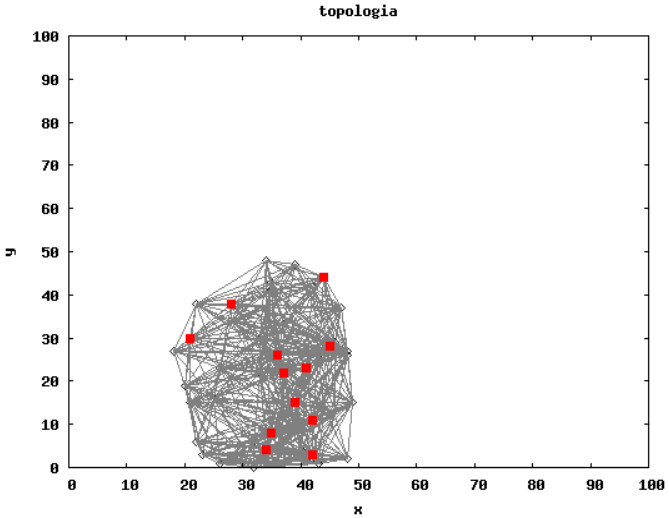


*Figura 154: Uma topologia formada a partir de G(V,E), N=50, conexo, aleatório uniformemente distribuído (apenas clusterheads)*

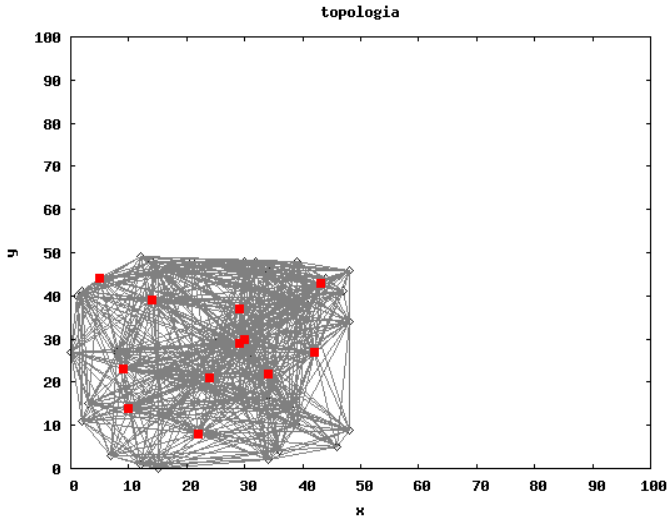


*Figura 158: Uma topologia formada a partir de G(V,E), N=50, conexo, aleatório uniformemente distribuído (apenas clusterheads)*

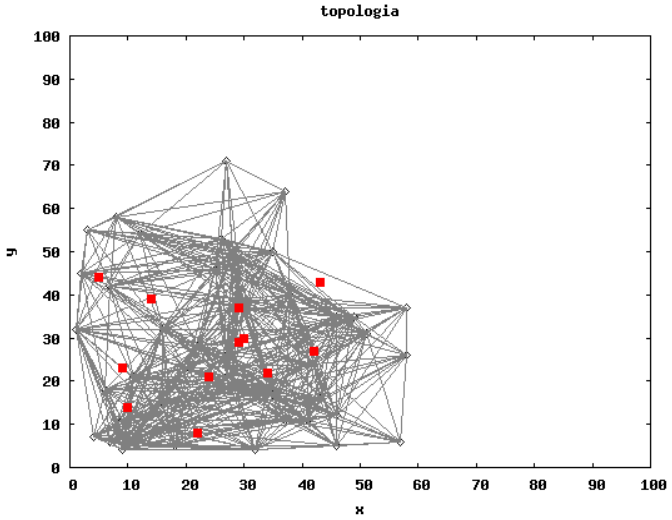


*Figura 157: Uma topologia formada a partir de G(V,E), N=50, conexo, aleatório uniformemente distribuído (apenas clusterheads)*

# E - EXPRESSÕES REGULARES PARA CONSULTAS DE LOGS

## E.1. INTRODUÇÃO

As informações obtidas durante o processo de experimentação foram coletadas por meio de *logs* de cada execução. Os *logs* são informações textuais. Além de fornecerem informações essenciais à análise e interpretação de dados, também servem de registro de todas as simulações, a fim de proporcionar histórico e aumentar a confiabilidade do presente estudo.

Foi armazenada a maior parte das simulações que forneceram dados para o presente trabalho, catalogados em 3.272.477 linhas de *logs*, a partir de 3.480 simulações.

A manipulação grandes textos é uma tarefa complexa. Particularmente quando se deseja extrair dados para estudo. O estabelecimento de filtros utilizando *Shell Script/Awk*, associados à Expressões Regulares (ER) é muito útil para a extração de informações relevantes de uma grande quantidade de arquivos muito extensos.

## E.2. FILTROS DE LOGS

Este Apêndice apresentará alguns filtros usando para manipulação de arquivos de *logs* (*Shell Script/Awk*/ER):

- Variáveis:

```
#MHAdHoc
keyphrase=' analisaFluxoInterIntraClusters'
#WCA
keyphrase=' analisaFluxoInterIntraClustersWca'
#HDA
keyphrase=' analisaFluxoInterIntraClustersHigh'
#Cenarios de Simulaccao (*)
filename=logCenario[7,8,9,10]Metodo[VA,BL,SLEP]
```

- Extração de informações sobre $\psi_{inter}$ :

*$ cat log.txt | egrep "$keyphrase\:' -A 4 | egrep fluxointer | awk '{print $3}'*

*$ cat log.txt | egrep "$keyphrase\:' -A 3 -B 8*

- Extração de informações sobre $\psi_{intra}$ :

  *$ cat log.txt | egrep '$keyphrase\:' -A 4 | egrep fluxointra | awk '{print $3}'*

  *$ cat log.txt | egrep "$keyphrase\:' -A 3 -B 8*

- Extração de informações sobre Atualizações do Dominant Set DS(G):

  *$ cat log.txt | egrep '$keyphrase\:' -A 3 | egrep 'EH'*

- Extração de informações sobre NReaf:

  *$ cat log.txt | egrep 'analisaReafiliacoes\:'*

  *$cat log.txt | egrep 'ehPossivelReafiliarNoh\:NAO'*

- Extração de dados sobre $F_0$ em diferentes cenários de simulação para o GAdHoc, SAdHoc e TSAdHoc:

  *$ cat log$filename.txt | egrep 'ga_melhork' | awk '{print $3}' | tee -a filtro.txt*

  *$ cat log$filename.txt | egrep 'sa_melhork' | awk '{print $3}' | tee -a filtro.txt*

  *$ cat log$filename.txt | egrep 'ts_melhork' | awk '{print $3}' | tee -a filtro.txt*

- Extração de dados sobre $T_{exec}$ em diferentes cenários de simulação para o GAdHoc, SAdHoc e TSAdHoc:

  *$ cat log$filename.txt | egrep 'Tempo' | cut -d\= -f2 | tee -a filtro.txt*

- Extração de dados sobre $F_0$ em diferentes cenários de simulação para os Métodos de Seleção de Vizinhos $s$ de $s_0$ SLEP, BL e VA:

  *$ cat log$filename.txt | egrep 'melhork' | awk '{print $3}' | tee -a filtro.txt*

- Extração de dados sobre $T_{exec}$ em diferentes cenários de simulação para os Métodos de Seleção de Vizinhos $s$ de $s_0$ SLEP, BL e VA:

  *$ cat log$filename.txt | egrep 'Tempo' | cut -d\= -f2 | tee -a filtro.txt*

- Localiza e faz backup de *logs*

*$ find $HOME/Simulacao -name \log*.txt -type f -exec cp -v { } /$HOME/$DIR_BACKUP \;*

- Contagem de linhas de arquivos do protótipo

*$ find $HOME/$DIR -name "log*.txt" -exec wc -l { } \; | awk '{print $1}' | tee -a filter.txt*

*$ find $HOME/$DIR "*.[c,h]*" -maxdepth 1 -exec wc -l { } \; | awk '{print $1}' | tee -a filter.txt*

*$ find $HOME/$DIR "*.sh" -maxdepth 1 -exec wc -l { } \; | awk '{print $1}' | tee -a filter.txt*

*$ find $HOME/$DIR "*.java" -maxdepth 1 -exec wc -l { } \; | awk '{print $1}' | tee -a filter.txt*

(*) refere-se à exportação de *logs* dos métodos VA,BL,SLEP para diferentes cenários de simulação 7,8,9,10. A nomenclatura de atribuição de *filename* apresenta uma simplificação de sintaxe.

# F - SHELL SCRIPTS PARA GERAÇÃO DE GRÁFICOS USANDO GNUPLOT

## F.1. INTRODUÇÃO

Todos os gráficos gerados no corpo principal deste trabalho foram gerados utilizando o *Gnuplot*, v4.0 for *Unix/Linux* [GNU06], a partir das informações armazenadas em *logs* de cada execução. Este aplicativo é destinado à visualização de gráficos e superfícies. Apresenta domínio público GNU/GPL.

Aliada a esta ferramenta, foram desenvolvidos procedimentos em *Shell Script/Awk* para plotagem de gráficos e automatização de tarefas, extremamente úteis no presente estudo, para realizar tarefas tais como: armazenamento em galerias de imagens com distinção de data-hora, juntamente com seus *logs*, configuração de ambiente para cada necessidade, criação de menus de tarefas para execução em mais alto nível, a partir de parâmetros pré-configurados, dispensando a entrada repetitiva de instruções para a plotagem de uma sucessão de gráficos.

Foram desenvolvidos no presente trabalho, 44 *Shell Scripts* totalizando 4.319 linhas de código, somente para automatização de tarefas.

## F.2. CONFIGURAÇÃO DE PARÂMETROS NO GNUPLOT

Um exemplo de configuração de parâmetros para o *Gnuplot* em *Shell Script* (comentários precedidos do símbolo '#'):

```
#reinicia aplicação
#-------------------------------------------------------------------
reset

#titulo
#-------------------------------------------------------------------
set title "graph_title"
```

```
#identifying axis
#-----------------------------------------------------------------------
set ylabel "axis_Y"
set xlabel "axis_X"

#enables a key (or legend) describing plots on a plot and describe its position
#-----------------------------------------------------------------------
#set key bottom right
#set key top right
#set nokey
set key top right

#insert grid
#-----------------------------------------------------------------------
set grid

#determines ranges and interval graphs
#coment 'set [?range,?tics] to enable automatic values
#-----------------------------------------------------------------------
set xrange [0:100]
set yrange [0..100]
set xtics 010
set ytics 010

#draw a line in '?zero axis'
#-----------------------------------------------------------------------
set xzeroaxis lt -1
set yzeroaxis lt -1

#picture size
#-----------------------------------------------------------------------
set size 1.3,1.3

#configuring outputs
#-----------------------------------------------------------------------
#set term post eps enhanced color "Times" 12
#set out 'output.eps'
set terminal png size 420,320  #  nocropenhanced size 420,320
set output 'output.png'

#shows arrows from one point to another one
#-----------------------------------------------------------------------
#set arrow from ($4),($5) to ($6),($7) nohead
#set arrow from 10,10 to 20,20 nohead
#show arrow

#enables several plots (to disable just do: unset multiplot
#-----------------------------------------------------------------------
set multiplot

#plotting multiplot graphs
#-----------------------------------------------------------------------
plot 'filename1.txt' using ($1):($2) t "graphname1" with linespoints linetype 40
pointtype 5
```

```
        replot 'filename2.txt' using ($1):($2) t "graphname2" with linespoints linetype 8
pointtype 9
        replot 'filename3.txt' using ($1):($3) t "graphsname3 " with linespoints linetype 0
pointtype 8

        replot 'filename3.txt' using ($1):($2) t "graphname4" with linespoints linetype 1
pointtype 11
        replot 'filename3.txt' using ($1):($3) t "graphsname5 " with linespoints linetype 0
pointtype 10


        #end of file
        #------------------------------------------------------------------------
```

## F.3. SHELL SCRIPT PARA PLOTAGEM DE GRÁFICOS

Este Apêndice apresentará alguns filtros usando para manip

```
#!/bin/bash
DIR_GALLERY=$HOME/Gallery
flag=0
until [-eq $flag 1]; do

#insert header
#------------------------------------------------------------------------
#header

#options
#------------------------------------------------------------------------
echo '(1)xxxxxxxxxxxxxxxxxxxx'
echo '(2)xxxxxxxxxxxxxxxxxxxx'
echo '(3)xxxxxxxxxxxxxxxxxxxx'
# (....)

#other options
#------------------------------------------------------------------------
echo '(21)save graph'
echo '(22)clear screen'
echo '(23)kill eog process'
echo' (0)exit script'
read option

case $opcao in
    1)name='option1'
      ;;
    2)name='option2'
      ;;
    3)name='option3'
      ;;
     #(...)
```

```
        21)cp $nome.png $DIR_GALLERY/$name`date+%Y-%m-%d-%M-%S`.png
            ;;
        22)killal eog
            ;;
        23) exit
            flag=1
            ;;
esac

#plotting graph
#------------------------------------------------------------------------
gnuplot < $name.sh

#showing graph with eog
#------------------------------------------------------------------------
eog $name.png &

#end of loop
#------------------------------------------------------------------------
done
```

## F.4. LINE TYPES E POINT TYPES NO GNUPLOT

Compatível com *Gnuplot* v4.0 *Unix/Linux*.

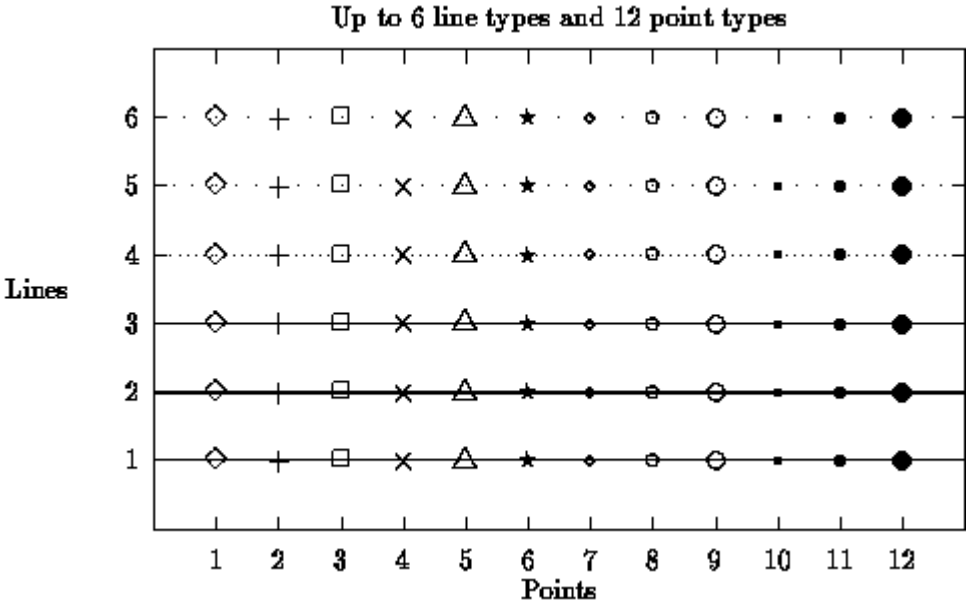


*Figura 159.: Point types e Line Types para Gnuplot*

# G - DETALHES DE CÁLCULO DA COMPLEXIDADE DE TEMPO E ESPAÇO

## G.1. INTRODUÇÃO

Neste apêndice, serão apresentados cálculos detalhados para a obtenção da complexidade de tempo do algoritmo básico Busca Tabu empregado no modelo proposto, complexidade de tempo de procedimentos adicionais utilizados na implementação da BT e complexidade de espaço da BT.

## G.2. COMPLEXIDADE DE TEMPO DO ALGORITMO BUSCA TABU

A Figura 160 apresenta o algoritmo básico BT, empregado no TSAdHoc, conforme Figura 35, descrito no Capítulo 2.

```
        TabuSearch(V,E,MAX_ITER)
        {
01.         s0 = y(); //initial solution
02.         s* ← s0 ;                          //best solution
03.         Iter ← 0 ;                         //iteration counter
04.         BestIter ← 0 ;
05.         LT ← Ø ;                           //tabu list is empty
06.         A = InitializeAspirationFunction(); //A(v)

07.         while (termination condition) do
08.         {   Iter++;
09.             Choose  s' ← s0 ∘ m | { ∀ s'' ∈ V ⊆ N_open(s0), f(s') ≤ f(s'') }
                        and  { f(s') ≤ A(f(s*)) } ;
10.             UpdateTabuList(LT, s0, s');
11.             s0 ← s' ;
12.             if  ( f(s0) < f(s*) )
13.             {     s* ← s0 ;
14.                   BestIter ← Iter ;
                }
15.             UpdateAspirationFunction(A);
            }
16.         return  s* ;
        }
```

*Figura 160: Algoritmo básico Busca Tabu empregado no TSAdHoc*

A Figura 161 apresenta procedimentos empregados no algoritmo básico Busca Tabu implementados no modelo proposto.

```
GenerateInitialSolution()
{       generate a random matrix g(n,k);
}

float InicializeAspirationFunction()
{       calculate A=A(f(s*));
        return A;
}

UpdateTabuList(LT,s,s')
{       if(quant<|T|)
        {   insert movement in the end of queue LT;
            quant++;
        } else{
            remove movement from the beginnig of queue LT;
            insert movement in the end of queue LT;}
}
float UpdateAspirationFunction(A)
{       calculate A=A(f(s*))
        return A;
}
```

*Figura 161: Procedimentos do algoritmo básico Busca Tabu empregado no TSAdHoc [MIL04]*

A partir das Figuras 160 e 161, a complexidade de tempo do algoritmo básico é calculada da seguinte forma:

Linha 01: gerar matriz n=|V| e k: $\Theta(nk)$ .

Linha 02 a05: atribuições e inicialização da LT (atribuição de NULL à LT): $\Theta(1)$ .

Linha 06: depende de *A(f(s*))*, que depende de *f*, dado por: $\Theta(n^2k)$ . Observamos que o parâmetro $|E|$ , de G=(V,E), foi utilizado no cálculo de *f*. A estrutura de dados que o representa é uma matriz *nxn*, sendo $n=|V|$ .

Linha 07: depende do número máximo de iterações $O(\text{MAX\_ITER})$ .

Linha 08: operação de soma e atribuição do resultado a *Iter*: $\Theta(1)$ .

Linha 09: dentro de V (i), para obtermos *s'*, precisamos realizar um movimento *m* (ii), verificar se *m* não está na LT (iii). Se estiver, é preciso verificar se supera o Critério de Aspiração para ser aceito (iv).

(i) $\forall s'' \in V \subseteq N_{open}(s_0), f(s') \leqslant f(s'')$ : a seleção de s' depende do número de vizinhos do subconjunto V, dado por MAX_V. Logo: $O(\text{MAX\_V})$

(ii) $s' \leftarrow s_0 \circ m$ : o movimento *m*, depende da seleção aleatória de uma linha i e do shift com query de i para a direita ou para a esquerda. O primeiro caso depende da seleção de um número aleatório. Para efeito de simplificação do problema, consideraremos $O(1)$ . Para o segundo caso, a seleção aleatória da direção do movimento também apresenta $O(1)$ e o shift na linha i, com k elementos, apresenta $O(k)$ .

(iii) A verificação se m está na LT depende apenas no tamanho da LT, |T| . Para o pior caso, $O(|T|)$ .

(iv) O Critério de Aspiração depende de *A(f(s*))*, que depende de $f$ , dado por: $\Theta(n^2 k)$ .

De (i), (ii), (iii) e (iv), temos:

$$O(\text{MAX\_V}(1+k+|T|+n^2 k))$$

Sendo $O(k+n^2 k)=O(n^2 k)$ , então:

$$O(\text{MAX\_V}(1+|T|+n^2 k))$$

Como $|T| \ll n^2 k$ , então:

$$O(\text{MAX\_V}(1+n^2 k))$$

Linha 10: a atualização da LT é feita pela inserção do movimento *m(s₀,s')* no final de uma fila de tamanho |T| . Se LT não estiver cheia, a inserção ocorre no fim da fila e a quantidade (*quant*) de elementos da LT é incrementada de uma unidade(i). Senão, o primeiro elemento a fila é retirado e a inserção de m é realizada, preservando o tamanho de LT (ii).

(i) a inserção é direta, no fim da fila $O(1)$ e o incremento de *quant* tem complexidade $O(1)$ . Logo: $O(1)$

(ii) a remoção do primeiro elemento é direta, com $O(1)$ e a inserção também. Logo: $O(1)$

De (i) e (ii), a partir do condicional *se (LT não estiver cheia) então....senão*, vale o pior caso:

$O(1)$

Linhas 11 a 14: $O(1)$

Linha 15: a atualização do Critério de Aspiração depende de *A(f(s*))*, que depende de $f$, dado por: $O(n^2 k)$ .

Linha 16: apenas o retorno do resultado encontrado: $O(1)$ .

A partir da complexidade de tempo obtidas nas Linhas 01 a 16, excetuando as linhas com $O(1)$ , temos:

$$\underbrace{O(nk)}_{linha\,1} + \underbrace{O(n^2 k)}_{linha\,6} + \underbrace{O(\text{MAX\_ITER}(\text{MAX\_V}(1+n^2 k)+n^2 k))}_{linhas\,7,9\,e\,15} =$$

Sendo $O(nk)+O(n^2 k)=O(n^2 k)$ , então:

$$O(n^2 k) + O(\text{MAX\_ITER}(\text{MAX\_V}(1+n^2 k)+n^2 k)) =$$

Sendo $O(\text{MAX\_V}(1+n^2 k)+n^2 k)=O(\text{MAX\_V}(1+n^2 k))$ , então:

$$O(n^2 k) + O(\text{MAX\_ITER}(\text{MAX\_V}(1+n^2 k))) =$$

$$O(\text{MAX\_ITER}(\text{MAX\_V}(1+n^2 k))) =$$

$$O(\text{MAX\_ITER}\,\text{MAX\_V}\,n^2 k) =$$

Admitindo MAX_V como valor constante no contexto do problema, então:

$$O(\text{MAX\_ITER}\,n^2 k)$$

## G.3. COMPLEXIDADE DE TEMPO DA IMPLEMENTAÇÃO DO ALGORITMO BUSCA TABU NO TSADHOC

Além da implementação do algoritmo básico, alguns procedimentos adicionais foram agregados para adequar o modelo TS ao problema em estudo e compor o modelo proposto TSAdHoc. A citar: método *TabuSearch::isValidMovement()* e *Solution::isValidSolution()*.

### G.3.1. Complexidade de Tempo do método *TabuSearh::isValidMovement()*

Houve a necessidade de verificar a validade do movimento *m*, $s' \leftarrow s_0 \circ m$ , para a

obtenção de um vizinho $s'$ de $s_0$. Observou-se experimentalmente que a escolha aleatória de uma linha de $s_0$ para realizar o *shift* para a direita ou para esquerda pode gerar uma solução absurda pelo ao ingresso de um nó não adjacente ao *cluster* de destino ou porque o nó movimentado tornou o *cluster* de origem desconexo.

A solução foi implementar o método *isValidMovement* na classe *TabuSearch*, como é mostrado na Figura 162. A complexidade de tempo é calculada neste método para verificar se exerce alguma influência no resultado da complexidade computada do algoritmo básico Busca Tabu aplicado no modelo proposto.

```
Class TabuSearch::isValidMovement(m, ClusterOrig clusternodes list, ClusterDest
clusternodes list)
{
        //Part one: Verify ClusterOrig connectivity without node moved to ClusterDest
01.     listA , queueB → Ø   ;
02.     ptrAuxQueueB → ptrHeadQueueB   ;
03.     copy ClusterOrig clusternodes list  to listA;
04.     remove one ID from listA and insert to queueB;
05.     ptrAuxQueueB++   ;
06.     do
07.     {     ptrAuxListA → ptrHeadListA   ;
08.          while(   ptrAuxListA ≠ NULL   )
             {
09.               if  (ptrAuxListA–> ID is adjacent to ptrAuxQueueB–> ID)
10.                    remove this ID from listA and insert to queueB;
11.                ptrAuxListA++   ;
             }

        }while(   ptrAuxQueueB ≠ NULL   )

        //Part two: Verify ClusterDest connectivity with new node came from ClusterOrig
12.     if(new node is adjacent to at least one node from ClusterDest nodes list)
13.          node is connected;
14.     else
15.          node is not connected;

        //The movement m is valid if ClusterOrig and ClusterDest are connected
16.     if((listA is not empty)&&(new node is connected))
17.          return TRUE;
18.     else
19.          return FALSE;
}
```

*Figura 162: Método isValidMovement da classe TabuSearch utilizado no modelo proposto*

A complexidade de tempo do algoritmo básico é calculada da seguinte forma:

Linhas 01 e 02: inicialização das estruturas auxiliares lista A (*listA*) e da fila B (*QueueB*) e atribuição do ponteiro auxiliar da fila B (*ptrAuxQueueB*), que aponta para o início da fila B: $\Theta(1)$

Linha 03: informação sobre lista de nós armazenada no *cluster* de origem é copiada para a lista auxiliar A (*listA*): $\Theta(n)$ .

Linhas 04 e 05: escolher um nó do cluster como ponto de partida. Remover um ID da lista A (i) e inserir na fila B (ii). Em seguida, incrementar o ponteiro auxiliar da fila B (*ptrAuxQueueB*) (iii).

(i) a remoção do elemento é direta, com $O(1)$

(ii) a inserção é direta, no fim da fila, com $O(1)$

(iii) incremento do ponteiro tem complexidade $O(1)$

De (i), (ii) e (iii), temos

$O(1)$

Linha 06: percorre os elementos da fila B. Se o *cluster* de Origem (*ClusterOrig*) for conexo, a lista terminará vazia e todos os elementos da lista A passarão para a fila B. Como a lista A tem complexidade $O(n)$ , então, $O(n)$ .

Linha 07: atribuição do ponteiro auxiliar da lista A (*ptrAuxListA*), que aponta para o início da lista A (listA): $O(1)$ .

Linha 08: percorre os elementos da lista A (listA). No pior caso, temos $O(n)$ .

Linhas 09 a 11: $O(1)$

Linha 12: o condicional é satisfeito se for encontrado, pelo menos, um nó do cluster de destino adjacente ao nó movimentado. Logo, $O(n)$ .

Linhas 13 a 15: para o escopo de *então....senão*, vale o pior caso do condicional. Como $O(1)$ para ambos os casos, então $O(1)$ .

Linhas 16 a 19: para o condicional *se....então....senão*, vale o pior caso do condicional. Como $O(1)$ para ambos os casos, então $O(1)$ .

A partir da complexidade de tempo obtidas nas Linhas 01 a 19, excetuando as linhas com $O(1)$ , temos:

$$\underbrace{O(n)}_{linha\,3} + \underbrace{O(n(1+n.(1+1+1)))}_{linhas\,6\,a\,11} + \underbrace{O(n)}_{linha\,12} =$$

Sendo $O(n) + O(n) = O(n)$ ,então:

$$O(n) + O(n(1+n.(1+1+1))) =$$

Sendo, $O(n(1+n.(1+1+1)))=O(n^2)$ então:

$$O(n) + O(n^2) =$$

$$O(n^2)$$

### G.3.2. Complexidade de Tempo do método *Solution::isValidSolution()*

Foi necessário verificar a validade da solução $s_0$, no procedimento de geração de uma solução inicial (*GenerateInitialSolution()*). Observou-se experimentalmente que a geração aleatória da matriz $G_{ik}(n,k)$, constituinte de $s_0$, pode conduzir a *clusters* disconexos, não correspondendo a uma solução realística.

A solução foi implementar o método *isValidSolution* na classe *Solution*, como é mostrado na Figura 163. A complexidade de tempo é calculada neste método para verificar se exerce alguma influência no resultado da complexidade computada do algoritmo básico Busca Tabu aplicado no modelo proposto.

Linha 01: percorre cada *cluster* e verifica se os seus nós constituintes são conexos. Logo, $\Theta(k)$ .

Linhas 02 e 03: inicialização das estruturas auxiliares lista A (*listA*) e da fila B (*QueueB*) e atribuição do ponteiro auxiliar da fila B (*ptrAuxQueueB*), que aponta para o início da fila B: $\Theta(1)$ .

Linha 04: informação sobre lista de nós armazenada no *cluster* atual é copiada para a lista auxiliar A (*listA*). Logo, $\Theta(n)$ .

Linha 05: percorre os elementos da fila B. Se o *cluster k* atual for conexo, a lista terminará vazia e todos os elementos da lista A passarão para a fila B. Como a lista A tem

complexidade $O(n)$ , então, para o pior caso temos $O(n)$ .

Linhas 06 e 07: escolher um nó do cluster como ponto de partida. Remover um ID da lista A (i) e inserir na fila B (ii). Em seguida, incrementar o ponteiro auxiliar da fila B (*ptrAuxQueueB*) (iii).

(i) a remoção do elemento é direta, com $O(1)$

(ii) a inserção é direta, no fim da fila, com $O(1)$

(iii) incremento do ponteiro tem complexidade $O(1)$

De (i), (ii) e (iii), temos

$O(1)$

Linha 08: atribuição do ponteiro auxiliar da lista A (*ptrAuxListA*), que aponta para o início da lista A (listA): $O(1)$ .

Linha 09: percorre os elementos da lista A (listA). No pior caso, temos $O(n)$ .

Linhas 10 a 12: $O(1)$

Linhas 13 a 14: para o condicional *se....então....*, vale o pior caso do condicional. Logo: $O(1)$ .

Linha 15: apenas o retorno do resultado encontrado: $O(1)$ .

A partir da complexidade de tempo obtidas nas Linhas 01 a 15, temos:

$$\underbrace{O(k(1+1+\underbrace{n}_{\text{linhas 1 a 14}}+n(1+1+1+n(1+1+1)+1+1)))}_{\text{linhas 1 a 14}} + \underbrace{O(1)}_{\text{linha 15}} =$$

$$O(k(n+n^2)) =$$

$$O(k(n^2)) =$$

$$O(n^2 k)$$

```
    Class SolutionS::isValidSolution
    {
          //Verify Cluster connectivity
01.       foreach Cluster k do
          {
02.            listA, queueB → Ø   ;
03.             ptrAuxQueueB → ptrHeadQueueB   ;
04.           copy Cluster-k nodes list  to listA;
05.           do
06.           {   remove one ID from listA and insert to queueB;
07.               ptrAuxQueueB++   ;
08.               ptrAuxListA → ptrHeadListA   ;
09.               while(  ptrAuxListA ≠ NULL   )
                  {
10.                   if(  ptrAuxListA -> ID is adjacent to ptrAuxQueueB -> ID)
11.                       remove this ID from listA and insert to queueB;
12.                    ptrAuxListA++   ;
                  }
              }while(  ptrAuxQueueB ≠ NULL   )

              //If at least one node is not connected, the solution is not valid
13.           if(listA is not empty)
14.               return FALSE;
          }
          //The solution is valid if all Clusters are connected
15.       return TRUE;
    }
```

*Figura 163: Método isValidSolutiont da classe Solution utilizado no modelo proposto*

## G.4. COMPLEXIDADE DE ESPAÇO DO ALGORITMO BUSCA TABU

Por não ser um algoritmo populacional, durante a execução do algoritmo Busca Tabu, são armazenados em memória a solução atual, solução vizinha da solução atual e a melhor solução encontrada. A complexidade de espaço de cada solução é $O(nk)$ . Outra estrutura armazenada é a Lista Tabu, de comprimento $|T|$. Logo, $O(|T|)$ . Demais estruturas apresentam complexidade constante ou unitária. Portanto, a complexidade de espaço do algoritmo básico da Busca Tabu é dada por $O(nk)+O(|T|)=O(nk+|T|)$ .

## G.5. COMPLEXIDADE DE ESPAÇO DA IMPLEMENTAÇÃO DO ALGORITMO BUSCA TABU NO TSADHOC

Detalhes acrescentados na implementação não influenciaram na complexidade de espaço computada teoricamente.