Leonardo Figueiredo Maia

O USO DO MÉTODO DO ENXAME DE PARTÍCULAS NA OTIMIZAÇÃO DA LATÊNCIA E CONSUMO DE ENERGIA DE UMA DE REDE-EM-CHIP

Dissertação apresentada ao Programa de Pós-Graduação Sistemas e Processos Industriais – Mestrado, Área de Concentração em Controle e Otimização de Processos Industriais, Universidade de Santa Cruz do Sul, como requisito parcial para obtenção do título de Mestre em Sistemas e Processos Industriais.

Orientador: Prof. Dr. João Carlos Furtado

Santa Cruz do Sul, abril de 2009

AGRADECIMENTOS

São tantos que tenho que agradecer que espero não esquecer de nenhum!
Agradeço a Deus, as entidades e aos espíritos de luz que sempre estão comigo.
Agradeço a minha família, em especial minha mãe e a minha filha pelo apoio.
Agradeço aos meus colegas da Unisc, em especial aos de Santa Maria pelas aventuras que vivemos e ao companheirismo.
Agradeço ao meu orientador, prof. Dr. João Carlos Furtado pelo apoio, puxões de orelhas e pela eterna paciência.
Agradeço ao Coordenador do Programa de Pós-Graduação em Sistemas e Processos Industriais, prof. PhD. Marco Flôres Ferrão, por sempre exigir o melhor de nós mestrandos, pela compreensão e pela paciência que teve conosco.
Agradeço aos demais professores, em especial ao prof. Dr. Ruben Edgardo Panta Pazos que ao longo desta jornada além de ensinar, também foi meu “terapeuta”.
Agradeço ao bolsista Fernando Konzen Kirch, pela imensa força que me deu com o simulador. Sei que tem um futuro promissor.
Agradeço a secretária Janaína Ramires pela atenção, eficiência e pelos bate-papos.

“Investir em educação é o melhor juros que se pode colher”

Abraham Lincoln

RESUMO

Com o desenvolvimento de aplicações para computadores pessoais, servidores e dispositivos eletrônicos, os quais exigem um poder computacional cada vez maior, surgiu a necessidade de tecnologias que aperfeiçoassem tanto o desempenho físico quanto o lógico destes. A rede-em-chip, que é um conjunto de roteadores e canais ponto-a-ponto que interconectam os núcleos de um sistema integrado de modo a suportar a comunicação entre esses núcleos, surgiu como opção de otimização arquitetural do *hardware*. Já em relação ao *software*, existem várias técnicas que utilizam algoritmos inteligentes, como o Enxame de Partículas, o qual foi proposto por Eberhart e Kennedy em 1995, sendo um algoritmo de otimização estocástico de conceito simples, de fácil implementação, robustez para controlar parâmetros e eficiência computacional durante o processo de otimização. Sua heurística evolutiva é baseada em uma população (enxame), formada por indivíduos (partículas), cuja evolução se dá por meio da velocidade. Cada partícula possui uma velocidade que a permite percorrer o espaço de busca e uma posição nesse espaço. Este trabalho teve como objetivo otimizar a latência e o consumo de energia de uma rede-em-chip, utilizando duas topologias diferentes, a malha e a toróide através de um simulador usando o método de Enxame de partículas o qual, através dos experimentos realizados, se mostrou eficiente neste tipo de otimização.

Palavras-chave: Rede-em-chip, Enxame de Partículas, Simulador, Otimização.

ABSTRACT

With the development of applications for personal computers, servers and electronic devices, which require increasing computational power, came the need for technologies that improving both the physical performance such as logical. The network-on-chip, which is a set of routers and channels point-to-point interconnect to the core of an integrated system to support communication between these cores, appeared as an option for optimization of architectural hardware. In relation to software, there are several techniques that use intelligent algorithms, such as Particle Swarm, which was proposed by Kennedy and Eberhart in 1995, and an algorithm for optimization of stochastic simple concept, easy implementation, robustness to control parameters and computational efficiency during the optimization. Their heuristic is based on evolving a population (swarm), formed by individuals (particles), whose evolution is given by the speed. Each particle has a speed that allows the search through the space and position in space. This work aimed to optimize the latency and energy consumption of a network-on-chip, using two different topologies, the mesh and Toroid through a simulation using the method of particle swarm which, through the experiments, was efficient in this type of optimization.

Keywords: Network-on-chip, Particle Swarm, Simulation, Optimization.

LISTA DE SÍMBOLOS E SIGLAS

NOC – (Network-on-Chip) – Rede-em-Chip

SOC (*System-on-Chip*) – Sistema em Chip

HDL – Linguagem de Descrição de *Hardware*

PHITS (*Physical Units*) – Unidade Física

FLITS (F*low Control Units*) – Unidade de Controle de Fluxo

i, *j* – facilidades/partículas

x_i – posição da partícula *i* em relação ao eixo x

y_i – posição da partícula *i* em relação ao eixo y

H_{ij} – razão de aproximação entre as facilidades *i* e *j*

nl – coeficiente de não linearidade

r_i – raio da facilidade *i*

N – número de facilidades

n – número de partículas

c_{ij} – custo entre as facilidades *i* e *j*

wi – largura da facilidade

he – altura da facilidade

q_i – centro da facilidade *i*

v_i – velocidade atual da partícula *i*

c_1, c_2 – parâmetros de confiança

rand() – função aleatória

$pbest_i$ – melhor posição que a partícula *i* já obteve durante a busca

gbest – melhor posição encontrada pelas partículas no enxame

c_{1ini} – valor inicial para o parâmetro de confiança cognitivo

c_{1fin} – valor final para o parâmetro de confiança cognitivo

c_{2ini} – valor inicial para o parâmetro de confiança social

c_{2fin} – valor final para o parâmetro de confiança social

R – número de iterações

it – iteração atual

V_{gbest} – velocidade próxima à melhor posição do enxame

V_{pbesti} – velocidade próxima à melhor posição da partícula

v_{max} – limitador de velocidade

w – componente inercial

w_{ini} = valor inicial para o coeficiente de inércia

w_{fin} = valor final para o coeficiente de inércia

k – fator de constrição

c_{3r} – parâmetros de confiança na iteração it

$gbest_i$ – posição da partícula i, quando obtida a melhor avaliação global do enxame

C – centróide do enxame

c_{3ini} – valor inicial para o parâmetro de atração da partícula pelo grupo

c_{3fin} – valor final para o parâmetro de atração da partícula pelo grupo

($pbest_i - x_i$) – distância entre a melhor posição já encontrada pela partícula i e a posição atual dessa mesma partícula

($gbest_i - x_i$) – distância entre a posição da partícula i quando o enxame atingiu o ótimo global e a posição atual dessa partícula.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

SUMÁRIO

INTRODUÇÃO

Com a diminuição do tempo de resposta das aplicações em computadores pessoais ou em servidores, houve a necessidade do surgimento de tecnologias que otimizassem a performance destes, iniciando assim, uma constante busca por artifícios de otimização de desempenho de arquiteturas e aplicações.

Atualmente a forma de otimização mais popular e economicamente viável utilizada em *hardwares*, é a replicação de recursos computacionais como processadores, memórias e outros dispositivos, tornando simples máquinas comerciais em máquinas de processamento paralelo. Entretanto, Zeferino (1999) relata que essa forma de otimização não depende apenas da performance de seus recursos, mas também de seu sistema de comunicação, o qual geralmente é projetado com o objetivo de atender a um grupo de requisitos que maioria das vezes, não é o mesmo de um computador paralelo. Deste modo, para obter um alto desempenho de comunicação, é necessário projetar redes de interconexão específicas de maneira a atender os requisitos alvo da máquina paralela, surgindo como opção, redes-em-chip. As quais segundo Fernandes (2006) são baseadas em redes de interconexão para computadores paralelos, sendo que os núcleos do sistema integrado são considerados como nodos de um multicomputador e são interconectados através de uma rede formada por roteadores e canais ponto-a-ponto.

Já em *softwares*, existem várias técnicas de programação que utilizam algoritmos “inteligentes” que visam melhorar o desempenho dos mais variados tipos

de aplicações, tendo como exemplos o algoritmo genético, colônia de formigas, algoritmo busca tabu e o enxame de partículas, este último será utilizado no presente trabalho, pois caracteriza-se por sua simplicidade, robustez e eficiência.

Com a finalidade de otimizar o consumo de energia e latência de uma arquitetura de rede-em-chip através da disposição dos nodos aos roteadores, será desenvolvido um algoritmo na linguagem de programação orientada a objetos C++, utilizando o método de enxame de partículas, o qual será acoplado a um simulador desenvolvido por Kreutz que conforme Bruch *et al* (2007), descreve a arquitetura dos componentes internos de roteadores de uma rede-em-chip no nível de transação, a precisão ocorre em nível de ciclos e a cada período do relógio todos os componentes da rede são executados.

1.1 Motivação

A pesquisa em redes-em-chip está progredindo cada vez mais, são inúmeras possibilidades de configurações de seus componentes com a finalidade de melhorar o desempenho de suas aplicações e dela mesma. Dentre vários exemplos de pesquisas em redes-em-chip, cita-se, o projeto de um roteador parametrizável para síntese de redes-em-chip (SANTO *et al*, 2004), estudar diferentes tipos de topologias (MELLO e MÖLLER, 2003), avaliação de desempenho de rede-em-chip (BRUCH *et al*, 2007), entre outros.

Em *software* também não é diferente, pois a cada dia surgem novas técnicas de programação que visam buscar e otimizar resultados nas mais diversas aplicações, tais como o uso de algoritmos genéticos, busca tabu, otimização através do enxame de partículas, de *layouts* industriais, encontrando de modo eficiente a melhor maneira de dispor das facilidades. (MÜLLER, 2007).

Kreutz uniu o bom desempenho das redes-em-chip e a eficiência dos algoritmos genéticos e o busca tabu em um simulador de arquitetura de rede-em-chip. Este visa otimizar o consumo de energia e a latência da mesma através do melhor posicionamento dos nodos aos roteadores. Para complementar a pesquisa

de otimização arquitetural, foi acoplado mais um algoritmo heurístico ao simulador, o enxame de partículas.

Com este trabalho, dados importantes sobre a eficiência do uso do enxame de partículas na otimização de redes-em-chip foram adquiridos. E em trabalhos futuros poderão ser feitas comparações entre os algoritmos que integram o simulador, objetivando qual a melhor técnica para determinada situação.

1.2 Objetivos

1.2.1 Objetivo Geral

Aplicação do Método Enxame de Partículas na otimização do consumo de energia e da latência em uma arquitetura de rede-em-chip, utilizando duas topologias diferentes, a malha e a toróide.

1.2.2 Objetivos Específicos

- Pesquisar sobre o enxame de partículas e redes-em-chip;
- Entender e saber utilizar o simulador de redes-em-chip;
- Realizar experimentos no simulador, utilizando o algoritmo de enxame de partículas;
- Coletar e analisar os resultados obtidos através da aplicação do método, viabilizando ou não a eficiência do algoritmo.

1.3 Metodologia

Para a realização deste trabalho, o mesmo foi dividido em 3 etapas: inicial, intermediária e conclusiva.

1.3.1 Etapa Inicial

Foi feito um levantamento teórico, pesquisando em livros, artigos, *home-pages*, monografias, dissertações e teses sobre o estado da arte de redes-em-chip e enxame de partículas.

1.3.2 Etapa Intermediária

Foram realizados estudos sobre a funcionalidade e usabilidade do *software* utilizado para simular a arquitetura de uma rede-em-chip. Logo em seguida começou a modelagem do problema, seguida da construção do algoritmo, o qual foi desenvolvido na linguagem orientada a objetos C++, no ambiente Neatbeans 6.0 em plataforma Linux *Debian*, a execução dos experimentos foram realizadas em um *Notebook* Sempron, de 1.8 GHz de processamento e 256 MB de memória RAM, com plataforma Debian ETCH.

1.3.3 Etapa Conclusiva

Depois de realizar experimentos e obter os resultados, foi realizada analise nos dados obtidos e a divulgação dos mesmos, focando a eficiência do método enxame de partículas na otimização arquitetural de uma rede-em-chip.

1.4 Estrutura da Dissertação

Nos capítulos 2 e 3 encontra-se a fundamentação teórica sobre redes-em-chip e enxame de partículas respectivamente, o objetivo da mesma é dar um forte alicerce para o desenvolvimento deste trabalho.

O capítulo 4 aborda apresentação da pesquisa, os resultados obtidos e a análise dos mesmos.

2 REDES-EM-CHIP

2.1 Introdução

Com o crescente avanço da tecnologia computacional no desenvolvimento de circuitos integrados, se tornou possível à criação de um sistema completo em um único chip, ou seja, processadores, memórias, controladores e outros recursos, em uma única pastilha, tal sistema é denominado *SoC* (*System-on-Chip*) conforme relata Zeferino(2003). Os *SoCs* têm sua metodologia de projeto fundamentada na reutilização de blocos previamente projetados e verificados, denominados núcleos (*cores*), os quais podem ser divididos em três categorias, (*i*) s*oft-core*, é a descrição de um núcleo em linguagem de descrição de *hardware* (HDL - *Hardware Description Language*) que pode ser mapeada para diferentes processos de fabricação, (*ii*) *firm-core*, o qual contém mais informação, normalmente um *netlist* ou até informações sobre o posicionamento e o roteamento, e (*iii*) *hard-core* que é geralmente a descrição de um *layout* e inclui informações referentes à temporização do circuito para uma determinada tecnologia e já está pronto para ser utilizado no sistema. Os núcleos são interligados através de uma arquitetura de comunicação, normalmente baseados em um ou mais barramentos compartilhados. Entretanto, a comunicação por barramento é adequada apenas para sistemas com poucas dezenas de núcleos, pois possui limitações quanto ao desempenho de comunicação e ao consumo de energia, sendo assim, não atenderá aos requisitos dos futuros *SoCs*, os quais integrarão dezenas a centenas de núcleos (KREUTZ *et al*, 2007).

Para solucionar as limitações do barramento, surgiu a rede-em-chip ou NOC (N*etwork-on-chip)*, a qual é definida como um conjunto de roteadores e canais ponto-a-ponto que interconectam os núcleos de um sistema integrado de modo a suportar a comunicação entre esses núcleos.

Uma NOC é caracterizada por sua topologia, que organiza a rede sob a forma de um grafo, onde os roteadores são os vértices do grafo e os canais são os arcos utilizados, e por seus mecanismos de comunicação, os quais definem a maneira como as mensagens são memorização seus principais mecanismos de comunicação.

Em uma rede-em-chip os núcleos dos *SoCs* são considerados como nodos de um multiprocessador que é constituído por nodos de processamento e nodos de chaveamento. Na figura 1 está descrito o modelo genérico de uma rede-em-chip.

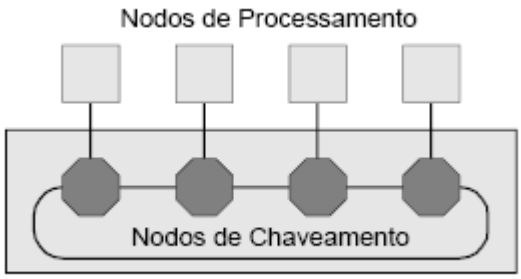

Figura 1 – Modelo Genérico de uma Rede-em-Chip.

2.1.1 Nodos

Os nodos de processamento ilustrados na figura abaixo executam as subtarefas do algoritmo paralelo e possuem no mínimo um processador e uma interface para a rede-em-chip, conhecida como *interface de rede*, havendo a possibilidade de ter ainda, memória local, discos e outros periféricos.

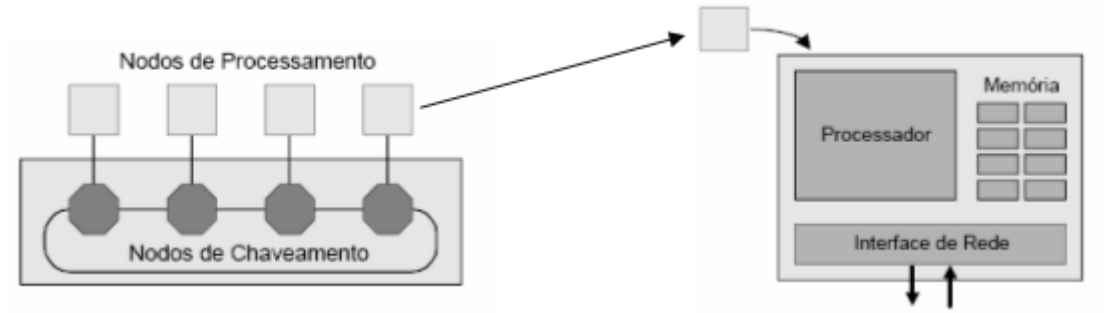

Figura 2 – Parte interna de um nodo de processamento.

Os nodos de chaveamento executam a transferência de mensagens entre os nodos de processamento. Na maioria das vezes, possuem um núcleo de chaveamento, uma lógica para roteamento e arbitragem, e portas de comunicação para outros nodos de chaveamento e as vezes para um nodo de processamento local.

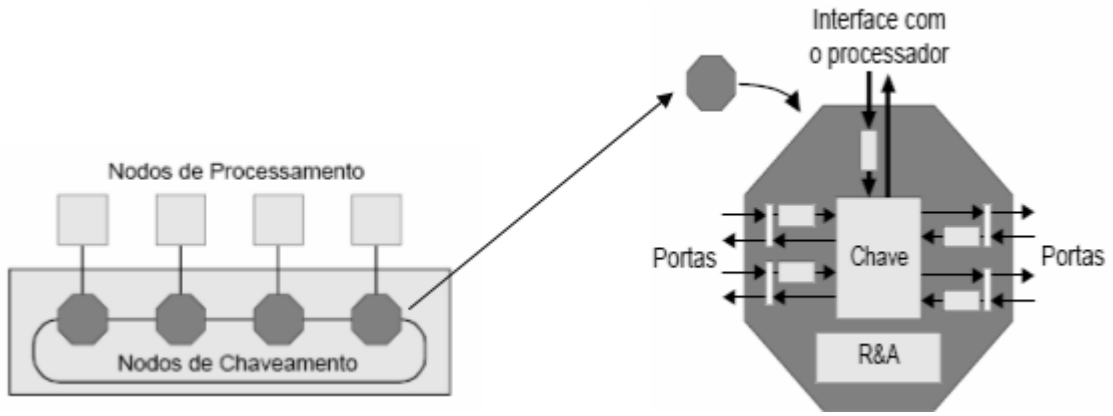

Figura 3 – Parte interna de um nodo de chaveamento.

2.1.2 Enlace

É a conexão física entre dois nodos de chaveamento, possuindo um ou dois canais físicos de comunicação, sendo implementado sob a forma de um cabo elétrico ou óptico, o qual transporta sinais analógicos. O transmissor e o receptor conectados em seus dois terminais devem efetuar as conversões adequadas para transferir a informação digital, função do controlador de enlace da porta de comunicação.

Figura 4 – Enlace entre dois nodos de chaveamento.

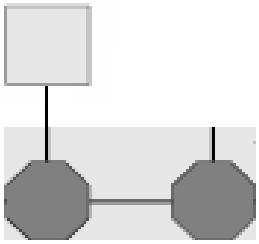

Figura 5 – Enlace entre dois nodos de chaveamento e um nodo local de processamento.

Para Culler e Singh (1999), um enlace caracteriza-se em relação a três propriedades distintas, o comprimento, pois um enlace é longo se diversos valores lógicos trafegam ao mesmo tempo pelo cabo, e curto se em um determinado instante largura e sincronismo, apenas um valor lógico trafega, em relação a largura, uma vez que um enlace é considerado fino, se as informações de dados e de controle transferidas na mensagem são multiplexadas em uma única linha serial e largo, se essas informações podem ser transferidas simultaneamente por meio de linhas paralelas e a última propriedade seria o sincronismo, porque um enlace só é considerado síncrono se os nodos origem e destino operam no mesmo relógio global e assíncrono se o nodo de origem codifica o seu relógio junto com o sinal analógico que é transmitido e o nodo receptor recupera o relógio fonte do sinal e transfere a informação segundo seu próprio domínio de relógio.

Segundo Zeferino (1999), a informação que é transferida por um enlace, trafega na forma *phits* (*physical units*) que são unidades de transferência física, sendo que um *phit* possui a largura do canal de dados. Em um enlace fino, o envio da informação pelo nodo transmissor se dá através de um pacote com valores que definem o início (*start-bit*) e o fim (*stop-bit*) do pacote, envolvendo a informação. Nos enlaces largos, podem ser utilizadas linhas de controle específicas com este fim.

Dependo do canal físico do enlace, este ainda é classificado como unidirecional, o qual tem apenas um canal físico origem-destino, tendo o fluxo de informação entre os dois nodos conectados ao enlace ocorrendo em uma única direção, ou bidirecional, quando o fluxo se dá nas duas direções e quaisquer dos nodos conectados ao enlace podem servir de origem ou de destino para uma comunicação. Os enlaces bidirecionais também se dividem em *half-duplex,* que possui dois canais unidirecionais opostos que compartilham um mesmo meio físico, e *full-duplex*, os dois canais opostos podem ser utilizados simultaneamente, sendo implementados através da atribuição de meios físicos separados para cada canal.

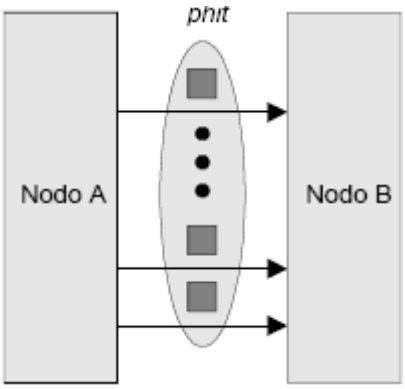

Figura 6 – Curto, largo e unidirecional.

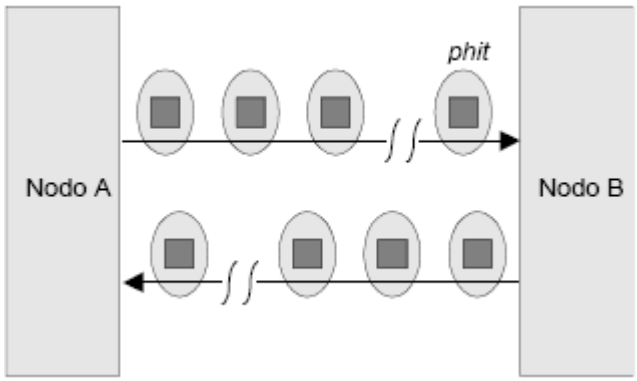

Figura 7 – Enlace, longo, fino e bidirecional.

2.2 Modelo de Comunicação

A comunicação entre os nodos é realizada sob a forma de mensagens que, tipicamente tem três partes, sendo um cabeçalho (*header*), onde são incluídas informações de roteamento e controle utilizadas pelos nodos de chaveamento para difundir a mensagem em direção ao nodo destino, a carga útil (*payload*), que seria a informação que está sendo transmitida, e um terminador (*trailer*), que por sua vez, inclui informações usadas para a detecção de erros e para a sinalizar o término da mensagem.

Figura 8 – Modelo das partes de uma mensagem.

Em geral, as mensagens são quebradas em pacotes para transmissão. Sendo o pacote a menor unidade de informação que contém detalhes sobre o roteamento e seqüenciamento dos dados, o pacote por sua vez é quebrado em *flits* (*flow control units*) que é a menor unidade sobre a qual é feita a regulação do tráfego, e este por fim se transforma em *phits*, os quais já foram abordados anteriormente.

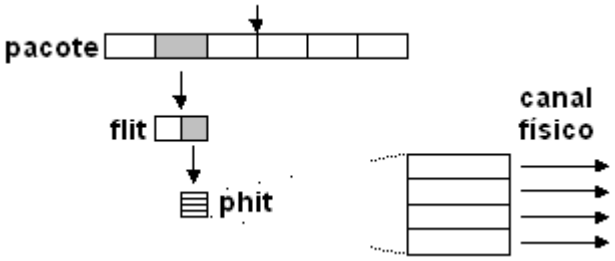

Figura 9 – Descrição da quebra de uma mensagem até *phit.*

2.3 Topologia

A topologia descreve como é o layout do meio através do qual há o tráfego de informações, e também como os recursos estão conectados. Está organizada sob a forma de um grafo G(N,C), onde *N* representa o conjunto de nodos da rede e *C* representa o conjunto de canais de comunicação.

Há uma divisão em duas classes principais nas redes-em-chip para multiprocessadores, as redes diretas e as redes indiretas.

2.3.1 Redes Diretas

Nas redes diretas, existe a integração de um nodo de chaveamento a um nodo de processamento, sendo que essa conexão, faz com que se torne apenas um elemento, ou simplesmente um único nodo, como está descrito na figura 10. O conjunto de nodos interconectados forma a topologia da rede, descrito figura 11.

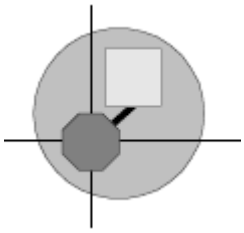

Figura 10 – Nodo formando pela integração de nodos de chaveamento e processamento.

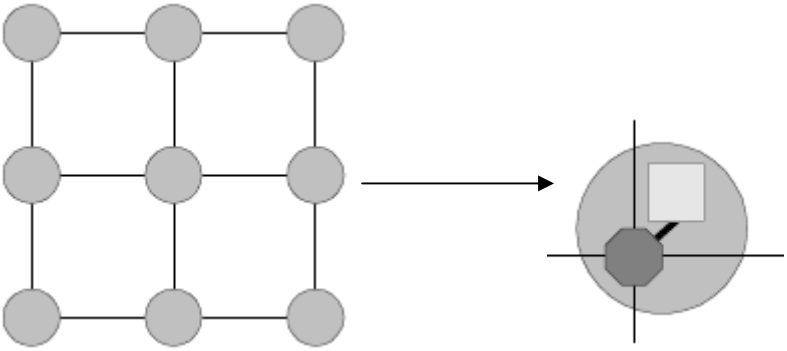

Figura 11 – Topologia formada pelo conjunto de nodos.

O trafego de mensagens pelos nodos se dá através de ligações ponto-a-ponto com seus nodos vizinhos e caso o destino de uma mensagem seja um nodo que não seja um vizinho, a mensagem é roteada e repassada para um determinado nodo intermediário até chegar ao destino.

Para melhor caracterização de cada topologia de rede direta, a qual é vista como um grafo de nodos e canais, existem alguma propriedades que são estudas que são, (i) grau do nodo, que é o número de ligações entre os nodos e seus vizinhos, (ii) diâmetro, que é o caminho mais curto entre dois nodos mais distantes na rede, (iii) regularidade, caso todos os nodos tiverem o mesmo grau, são regulares e (iv) simetria, se a rede for vista igualmente por todos os nodos, esta é simétrica. (ZEFERINO, 1999).

Algumas topologias de rede direta estão nas figuras abaixo.

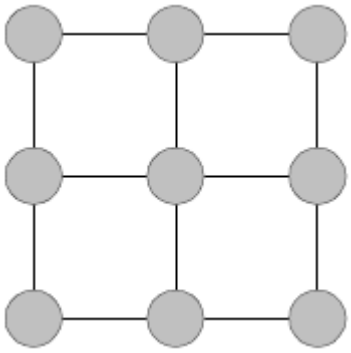

Figura 12 – Topologia Malha 3x3.

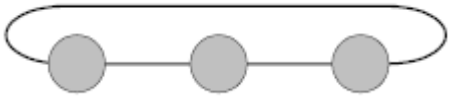

Figura 13 – Topologia Toróide 1D.

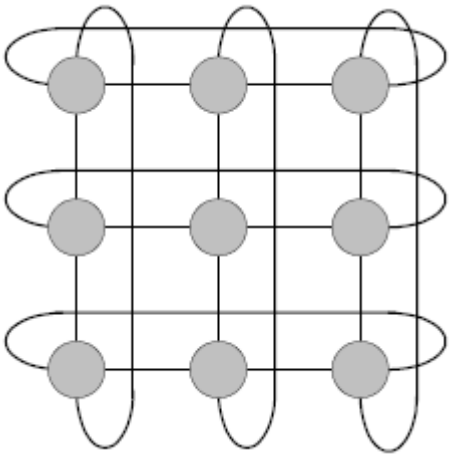

Figura 14 – Topologia Toróide 2D 3x3.

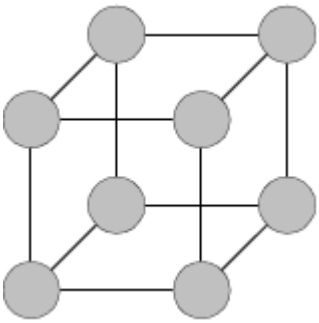

Figura 15 – Topologia Hipercubo 3D.

2.3.2 Redes Indiretas

Diferente do que acontece com as redes diretas, nas redes indiretas, o nodo de processamento não está conectado diretamente ao nodo de chaveamento, mas sim por uma interface para uma rede de nodos de chaveamento baseados em chaves, sendo que cada chave possui um conjunto de portas bidirecionais para ligações com outras chaves e/ou com os nodos de processamento. Para uma mensagem chegar ao seu destino, esta tem que atravessar algumas chaves. A topologia de uma rede indireta é tida como um grafo de nodos de chaveamento (ou chaves) e canais. Dentre as topologias desta rede, se destacam a matriz de chaveamento ou *crossbar* e as redes multiestágio.

Segundo Zeferino (1999), a topologia *crossbar* é ideal para conexão de N nodos de processamento, também provê maior largura de banda e capacidade de interconexão, pode ser vista como uma rede chaveada de estágio único. A chave

crossbar é composta por um conjunto de pontos de chaveamento que estabelecem conexões dinâmicas dedicadas entre pares fonte-destino.

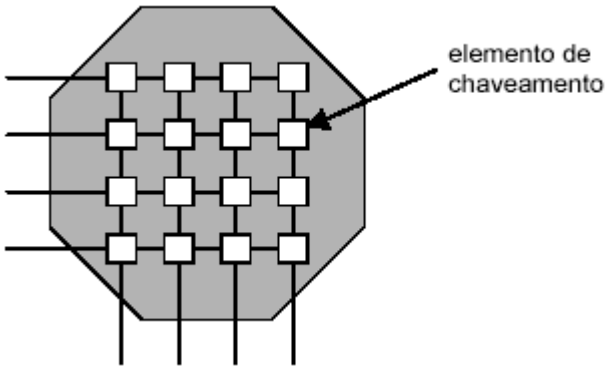

Figura 16 – Topologia Rede Crossbar 4x4.

Outra topologia já comentada é a multiestágios, as quais conectam entradas à saídas através de uma seqüência de estágio de chaves, onde cada chave é uma rede *crossbar*.

Algumas das permutações usadas no desenvolvimento de diversos tipos de redes multiestágios são a *perfect shuffle*, *digit reversal*, *cube*, entre outras. Podendo ser ainda de dois tipos, unidirecionais e bidirecionais.

2.4 *Starvation, Livelock e Deadlock*

A comunicação na rede-em-chip é realizada através da troca de mensagens entre seus componentes, no progresso das mesmas, algumas situações podem ocorrer que interfiram na funcionalidade da rede que devem ser consideradas no projeto, são elas: o *starvation* que conforme Mello e Möller (2003) acontece quando um pacote de uma fila requisita um canal de saída e é bloqueado porque o canal de saída está alocado a outro pacote; o *livelock* ocorre quando um pacote jamais chega a seu destino, ficando a trafegar pela rede, normalmente este problema afeta sistemas tolerantes a falhas (REGO, 2006), a última situação e a mais grave, é o *deadlock* que ocorre quando há uma dependência cíclica na rede, na qual cada mensagem garante a alocação de um canal e requer o uso de outro canal já alocado

a outra mensagem, causando a paralisação da rede. A solução mais indicada e mais econômica para esse tipo de problema é o uso do algoritmo XY, pois faz com que qualquer mensagem de um emissor para um receptor tenha sempre a mesma rota. (SANTO *et al*, 2004). A garantia da não ocorrência de *starvation*, em geral, é dada pelo mecanismo de arbitragem utilizado, enquanto que nos casos de *livelock* e *deadlock* depende exclusivamente do algoritmo de roteamento adotado. (ZEFERINO, 2003)

2.5 Mecanismos de Comunicação

Uma rede-em-chip é composta por mecanismos de comunicação que têm o objetivo de garantir que as mensagens enviadas cheguem a seu destino. Sendo estes o controle de fluxo, o roteamento, a arbitragem, o chaveamento e a memorização, além da topologia de rede, já abordada no tópico 2.2.

2.5.1 Controle de Fluxo

O controle de fluxo administra a distribuição dos *buffers* e canais entre diferentes pacotes. Como na maioria das vezes o número de pacotes em uma rede é maior que a vazão da mesma é necessário tomar decisões sobre o destino a ser dado a estes pacotes, como descarte, bloqueio no lugar onde estão, recebimento e armazenamento temporário, ou ainda desvio para outros caminhos.

Quando um pacote não pode prosseguir porque algum recurso que ele necessita já está sendo utilizado por um outro pacote, é dito que ocorreu uma colisão de recurso. O mecanismo de controle de fluxo deve idealmente evitar a colisão de recursos, o congestionamento dos canais e minimizar a latência das mensagens. As técnicas mais utilizadas são a de canais virtuais, *slack buffer* e a baseada em créditos (ZEFERINO, 1999).

- Controle de fluxo Canais Virtuais é a divisão do *buffer* físico logicamente em diversos canais de prioridades menores, tornando-se um *buffer* multi-via,

eliminando o problema conhecido como *Head od Line* (HOL), o qual é causado pela memorização baseada em FIFO (o primeiro pacote recebido é o primeiro a ser transmitido), já que se o primeiro pacote não puder ser transmitido, este ficará bloqueado, impedindo o tráfego dos demais pacotes subseqüentes.

- Controle de fluxo *slack buffer* pode ser visto como o controle de nível em um reservatório com marcas de nível baixo e de nível alto, onde o conteúdo do mesmo deve ser mantido entre esses limites. O *buffer* possui canais de entrada e de saída de dados e três regiões internas dimensionadas (superior, histerese e inferior). O controle do fluxo de entrada de dados no nodo receptor é realizado através do envio de dois tipos de símbolos ao nodo fonte, *Stop, que* é utilizado para solicitar ao nodo fonte a interrupção do envio de dados, e *Go,* que serve para requisitar o reestabelecimento desse fluxo.

- Controle de fluxo baseada em créditos é de simples implementação. O módulo receptor envia um sinal ao módulo transmissor informando o espaço disponível no buffer de entrada. Esta informação é vista como crédito. A transmissão de dados ocorre quando existem créditos disponíveis. Esta é uma técnica de controle de fluxo onde não há descarte de dados. (REGO, 2006).

2.5.2 Roteamento

O roteamento é usado por um pacote ou mensagem na escolher de um caminho a ser utilizado para atingir o seu destino, sendo fortemente dependente de seu algoritmo de roteamento. Alguns objetivos específicos precisam ser atendidos pelo algoritmo como a (i) conectividade: capacidade de rotear pacotes de qualquer nodo fonte para qualquer nodo destino; (ii) liberdade de *deadlock* e *livelock*: capacidade de garantir que nenhum pacote ficará bloqueado ou trafegando pela rede; (iii) adaptatividade: capacidade de rotear pacotes através de caminhos alternativos quando ocorre congestionamento ou falha em algum componente do caminho em uso; e (iv) tolerância a falhas: capacidade de rotear pacotes na presença de falhas em componentes.

Para Zeferino (1999 e 2003), as técnicas de roteamento recebem classificações com base em critérios adotados, tais como:

Local onde as decisões de roteamento são tomadas

- Centralizado: um controlador central estabelece os caminhos;
- Fonte: o nodo fonte define o caminho do pacote antes de inseri-lo na rede;
- Distribuído: o roteamento é realizado enquanto o pacote percorre a rede;
- Multifase: o nodo fonte define alguns nodos destinos, mas o caminho é definido de forma distribuída;

Adaptatividade

- Determinístico: é percorrido sempre o mesmo caminho entre um determinado par fonte-destino;
- Adaptativo: evita congestionamentos e falhas utilizando informações sobre o tráfego da rede;

Os algoritmos adaptativos ainda podem ser classificados quanto:

- à progressividade: progressivo, se o cabeçalho sempre avança pela rede, reservando um novo canal a cada passo de roteamento, ou regressivo, se o cabeçalho pode retornar pela rede, liberando canais previamente reservados.

- à minimalidade: mínimo, se o algoritmo de roteamento pode selecionar apenas canais de saída que aproximem cada vez mais o pacote do seu destino, ou não-mínimo, se o algoritmo de roteamento pode selecionar canais que levem o pacote a se afastar do seu destino.

- ao número de caminhos: completo, se o algoritmo de roteamento pode utilizar todos os caminhos disponíveis, ou parcial, se apenas um subconjunto desses caminhos pode ser usado.

Número de destinatários

- Unicast: os pacotes possuem apenas um destino;
- Multicast: os pacotes são roteados para múltiplos destinos;

Implementação

- Tabela: o roteamento é realizado a partir de uma consulta a uma tabela em memória;
- Máquina de estados: o roteamento é realizado a partir da execução de um algoritmo desenvolvido em *software* ou *hardware*;

2.5.3 Arbitragem

Para evitar *starvation,* ou seja, que pacotes fiquem aguardando de modo indefinido uma porta de saída para chegar ao seu destino, o mecanismo de arbitragem deve definir através de um algoritmo, qual porta de entrada será escolhida para sair por uma determinada porta de saída, resolvendo assim, conflitos internos na rede, como múltiplas requisições simultâneas para uma mesma porta de saída.

A arbitragem pode ser centralizada, onde as requisições das portas de entrada são realizadas por um único módulo (arbitragem e roteador), identificando a saída mais adequada para cada requisição e ele mesmo determina quais serão servidas, configurando o *crossbar*. Outro tipo de arbitragem é a distribuída, na qual a arbitragem e o roteamento são realizados em módulos separados e as requisições das portas de entrada serão encaminhadas ao seu destino pelo arbitro após serem escalonadas, por uma política de escalonamento.

A arbitragem pode ser baseada em diferentes critérios. São alguns exemplos os esquemas de prioridades estáticas, prioridades dinâmicas, escalonamentos por idade (ou *deadline*), FCFS (*First-Come-First-Served*), LRS (*Least Recently Served*), *round-robin*, entre outros.

2.5.4 Chaveamento

O chaveamento define como uma mensagem é transferida de um canal de entrada de um roteador para um de seus canais de saídas, ou seja, enquanto o roteador determina por onde os dados serão transmitidos, o chaveamento determina *como* isto acontece (SANTO *et al*, 2004).

Para Mello e Möller (2003) algumas das técnicas mais utilizadas de chaveamento, por circuito, por pacote, *store-and-forward*, *virtual cut-through* e *wormhole*.

- Chaveamento por circuito: um caminho é estabelecido antes do envio da mensagem entre a fonte e o destino. Se houver outra requisição, essa será negada, pois o caminho já está reservado.

- Chaveamento por pacote: a mensagem que será transmitida é quebrada em vários pacotes. Cada pacote possui um cabeçalho, o qual é verificado na chegada em cada chave intermediária, que por sua vez, decide para qual porta de saída ela deve enviar o pacote.

- Chaveamento *store-and-forward*: o pacote inteiro é recebido e armazenado, para depois ser enviado. Para ser armazenado integralmente, há necessidade de uma fila capaz de conter o pacote inteiro, fato que acarreta em uma alta latência em cada chave intermediária.

- Chaveamento *virtual-cut-through*: é um aperfeiçoamento do *store-and-forward*, tendo como diferença o armazenamento do pacote inteiro somente se a chave destino esteja ocupada.

- Chaveamento por *wormhole*: o pacote é dividido em *flits*, sendo transmitidos entre as chaves intermediárias até o destino.

2.5.5 Memorização

Determina a utilização de filas para armazenar uma mensagem bloqueada na rede quando o canal de saída requisitado já está alocado para uma outra mensagem (ZEFERINO, 2003). O esquema de memorização tem grande impacto no desempenho da rede como um todo, e na área ocupada em chip por seus roteadores (REGO, 2006).

A memorização dos pacotes pode ser feita na entrada ou na saída do roteador, podendo ser de forma centralizada ou distribuída. A forma mais utilizada é a memorização distribuída na entrada, onde existe um *buffer* para cada uma das portas de entrada, sendo o FIFO a estratégia mais adotada onde os pacotes vão sendo armazenados à medida que são recebidos em uma fila, e vão sendo transmitidos os primeiros pacotes da fila.

3 ENXAME DE PARTÍCULAS – EP

3.1 Introdução

Proposto por Eberhart e Kennedy em 1995, o método de otimização por exame de partículas foi desenvolvido a partir do trabalho do biólogo Frank Heppner, que ao analisar o comportamento de um grupo de pássaros a procura de alimento ou de um lugar para construir o ninho, percebeu que quando um pássaro encontra o alimento, todos os demais passam a encontrá-lo também, mais rapidamente. O que acontece é um aprendizado por parte do bando no momento que um dos pássaros adquire determinado conhecimento, conforme relata Gomes (2004). O mesmo comportamento também pode ser observado em outros grupos de animais, tal como os cardumes de peixes. Sendo assim, o método faz uma simulação do "comportamento social" dos pássaros e dos peixes, ou seja, o grupo é influenciado pela experiência individual acumulada por cada individuo, bem como, pelo resultado da experiência acumulada pelo próprio grupo, fazendo com que atinjam o objetivo.

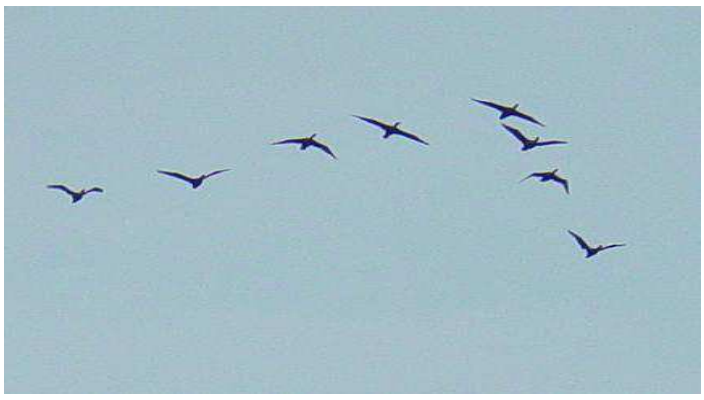

Figura 17 – Representação do comportamento de um bando de pássaros.

Figura 18 – Representação do comportamento de um cardume de peixes.

O algoritmo de otimização por enxame de partículas enquadra-se na família dos algoritmos evolutivos, os quais são baseados em uma gama de mecanismos da evolução biológica que serviram para originar outros algoritmos, como os algoritmos genéticos, colônia de formigas, entre outros (WIKIPÉDIA, 2007).

Segundo Biscaia, Schwaab e Pinto (2004), o EP consiste na otimização de uma função objetivo através da troca de informações entre as partículas ou agentes do grupo que mudam sua posição (estado) no espaço de busca, de acordo com a própria experiência e a influência das partículas vizinhas que constituem o enxame, tornando-se assim um algoritmo de otimização não-determinístico robusto, de fácil implementação e eficiente. Para Furtado, Crossetti e Pazos (2006), o EP é uma metaheurística evolutiva, ou seja, é uma aproximação algorítmica para encontrar soluções de problemas de otimização combinatória, baseada em uma população, cuja evolução se dá por meio da velocidade, controlando parâmetros e eficiência computacional.

3.2 Aplicabilidade do Método

Desde que foi desenvolvido, o método é utilizado nas mais diversas áreas, tais como otimização de processos químicos em minimizações de funções testes (BISCAIA e PINTO, 2004), otimização do projeto do núcleo de um reator nuclear em que se procura obter a melhor combinação de dimensões e constituição dos elementos do núcleo (MEDEIROS, 2005), simulação do desempenho de alocações de conversores de comprimento de ondas em nós de redes ópticas parciais para

ratificar os benefícios de conversores nas arquiteturas de redes (LOMONACO, 2006), otimização de *layouts* industriais, encontrando de modo eficiente a melhor maneira de dispor das facilidades (departamentos, máquinas, etc) dentro da indústria, (MÜLLER, 2007).

3.3 Algoritmo do Enxame de Partículas

No algoritmo de otimização por enxame de partículas, primeiramente, gera-se uma população inicial (enxame), onde cada indivíduo (partícula) é candidato a uma possível solução para o problema, ocupam uma posição no espaço de busca bidimensional (x,y) e uma velocidade aleatória.

As partículas locomovem-se pelo espaço de busca, analisando a melhor posição já visitada por elas mesmas e a melhor posição visitada pelo enxame, tais dados junto com os dados de suas velocidades, são armazenados em vetores que são consultados no processo de atualização de suas velocidades e posições, a procura da melhor solução para otimização do problema (CARVALHO e LUDERMIR, 2006).

A velocidade (1) e a nova posição (2) de cada partícula podem ser modeladas através das equações a seguir:

$$\boldsymbol{v_i^{it+1}} = v_i^{it} + (c_1{}^{it} * \text{rand}()^{it} * (pbest_i{}^{it} - x_i{}^{it}) + (c_2{}^{it} * \text{rand}()^{it} * (gbest^{it} - x_i{}^{it}) \quad (1)$$

$$\boldsymbol{x_i}^{it+1} = x_i^{it} + \boldsymbol{v_i}^{it+1} \quad (2)$$

desfragmentando a equação da velocidade:

$\boldsymbol{v_i^{it+1}}$ = representa a velocidade atualizada

v_i^{it} = representa a velocidade antiga

$(c_1{}^{it} * \text{rand}()^{it} * (pbest_i{}^{it} - x_i{}^{it})$ = representa o conhecimento da partícula

$(c_2{}^{it} * \text{rand}()^{it} * (gbest^{it} - x_i{}^{it})$ = representa a colaboração do enxame

desfragmentando a equação da posição:

$\boldsymbol{x_i}^{it+1}$ = representa a posição atualizada da partícula

x_i^{it} = representa a posição antiga

$\boldsymbol{v_i}^{it+1}$ = representa a velocidade atualizada, resultado da equação (1)

descrição das variáveis utilizadas nas equações (1) e (2):

v_i = velocidade da partícula *i*

c_1 e c_2 = coeficientes de aceleração

rand() = função randômica

pbest = melhor posição que a partícula *i* já obteve durante a busca

gbest = melhor posição encontrada pelas partículas no enxame

x_i = posição da partícula *i*

it = iteração atual

3.3.1 Coeficientes de Aceleração

Carvalho e Ludermir (2006) informam que os coeficientes são fatores de aceleração individual e global que aceleram o deslocamento das partículas. Para Esmin (2005) os coeficientes de aceleração c_1 e c_2 exercem influência no tamanho máximo do passo que uma partícula pode dar em uma única interação, fazendo com que na equação da velocidade o coeficiente c_1 ajuste o tamanho do passo na direção da melhor posição individual daquela partícula e o coeficiente c_2 ajuste de forma objetiva o tamanho máximo do passo na direção da melhor partícula do enxame.

Muller (2007), comenta que em alguns trabalhos, os valores utilizados pelos coeficientes de aceleração variam entre 0,5 e 2, sendo que o primeiro valor produziu melhores resultados.

As equações abaixo fazem com que os coeficientes c_1 e c_2 variem a cada interação do algoritmo:

$$c_1^{it} = \left(c_{1\,fin} - c_{1ini}\right)\frac{it}{R} + c_{1ini} \qquad (3)$$

$$c_2^{it} = \left(c_{2\,fin} - c_{2\,ini}\right)\frac{it}{R} + c2_{ini} \quad (4)$$

descrição das variáveis utilizadas nas equações (3) e (4):

c_{1ini} = valor inicial para o parâmetro de confiança cognitivo

c_{1fin} = valor final para o parâmetro de confiança cognitivo

c_{2ini} = valor inicial para o parâmetro de confiança social

c_{2fin} = valor final para o parâmetro de confiança social

R = número de iterações

it = iteração atual

3.3.2 Função Randômica

Segundo Müller (2007) a função randômica tem o objetivo de manter a diversidade do enxame gerando números aleatórios entre 0 e 1. Já Silva (2006) segue em outro enfoque, não a trata como uma função, mas sim, como uma variável aleatória de distribuição uniforme que pode tomar qualquer valor entre 0 e 1.

3.3.3 Algoritmo Original do Enxame de Partículas

O algoritmo original de otimização por enxame de partículas está descrito abaixo (ESMIN, 2005):

Algoritmo
Inicializar o enxame
Repetir
    Para cada partícula i = [1..T] faça
        Se f(T.x_i) < f(T.*pbest*)
            Então T.*pbest* = T.x_i
        Se f(T. $pbest_i$) < f(T.*gbest*)
            Então T.*gbest* = *pbest*
    Fim Para

Atualização da velocidade e da posição da partícula

Até que o critério de parada seja satisfeito

Fim Algoritmo

Descrição das etapas do algoritmo:

1. Inicializar o algoritmo fazendo com que as partículas recebam suas posições e velocidades iniciais de modo aleatório;
2. A variável T representa o tamanho do enxame;
3. Para cada partícula é calculada a função objetivo, representada pela variável f;
4. A primeira tomada de decisão, é uma comparação do valor obtido pela partícula com o seu *pbest*, caso o valor seja melhor, é atualizado o *pbest*;
5. A segunda tomada de decisão, é uma comparação do *pbest* da partícula *i* com a melhor posição obtida pelo o enxame, o *gbest*, caso o *pbest* seja melhor, o *gbest* é atualizado;
6. É atualizada a velocidade e a posição da partícula através das equações (1) e (2);
7. O fim da execução do algoritmo se dá a partir do momento que é alcançado o critério de parada, normalmente acentuado pelo número de interações.

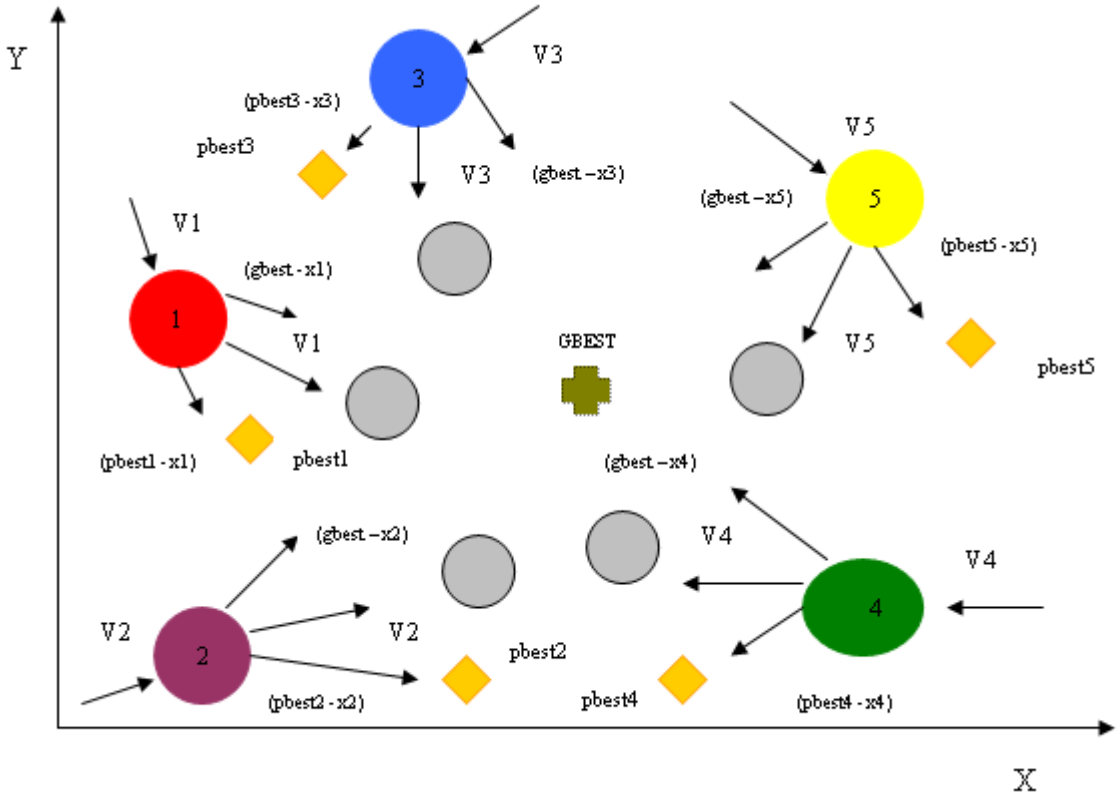

Figura 19 – Movimentação das partículas no espaço de busca (x,y).

Descrição:

$pbest_i$ = melhor posição da partícula

gbest = melhor posição das partículas no enxame

$(pbest_i - x_i)$ = distância entre a melhor posição da partícula e sua posição atual.

$(gbest - x_i)$ = distância entre a melhor posição das partículas no enxame e sua posição atual

= posições atuais

= próximas posições

Na figura 3, as partículas se movimentam pelo espaço de busca e atualizam suas posições utilizando as equações (1) e (2), e as atualizações de suas novas posições, estão indicadas pelos círculos cinzas. As equações, levam em consideração a velocidade anterior da partícula, *v*, a sua posição atual (x, y), a melhor posição que essa partícula já ocupou até o momento, *pbest* e a melhor posição de todas as partículas no enxame, *gbest*.

3.4 Principais alterações propostas ao algoritmo Enxame de Partículas

Ao longo de sua utilização, o método de otimização por enxame de partículas recebeu algumas sugestões de melhorias em relação ao algoritmo original.

3.4.1 Limitador de velocidade

Com a finalidade de controlar a velocidade da partícula para que a mesma não ultrapasse a melhor posição no espaço de busca devido a um alto valor de velocidade, ou que nem alcance a melhor posição devido a um valor insuficiente de velocidade, foi estabelecido o parâmetro v_{max} como limite da velocidade. De acordo com o problema de otimização v_{max} é estipulado pelo usuário, normalmente este valor fica entre -4 e 4, ou seja, a cada atualização da velocidade (v_i), se esta for maior que 4, o valor é substituído por 4 e caso a velocidade encontrada for menor que -4, seu valor será -4 segundo Kennedy *et al* (2001).

Shi e Eberhart (1998) relatam que apesar de localizar a melhor área mais rápido que os demais métodos, uma vez na região próxima da melhor posição, o algoritmo de otimização por enxame de partículas pode não continuar ajustando sua velocidade na tentativa de encontrar uma solução melhor. Para solucionar esse problema dois novos parâmetros, foram adicionados à equação da velocidade e utilizados separadamente, o componente inercial ou pesa da inércia e o coeficiente de enxugamento ou fator de constrição.

3.4.2 Componente inercial ou peso da inércia

O peso da inércia (*w*) é um fator escalar associado com a velocidade durante o passo de tempo anterior, resultando numa nova equação de atualização da velocidade:

$$v_i^{it+1} = w^{it} * v_i^{it} + c_1^{it} * \text{rand}()^{it} * (pbest_i^{it} - x_i^{it}) + c_2^{it} * \text{rand}()^{it} * (gbest^{it} - x_i^{it}) \quad (5)$$

O ideal é que *w* iniciasse com um valor alto e fosse decrementado a cada iteração para dar equilíbrio entre exploração global e local. Sendo assim, é possível encontrar soluções mais apuradas, suficientemente ótimas, em um menor número de iterações.

Eberhart e Shi (1998) utilizaram valores de *w* na faixa de 0 à 1.4, também variando com o passar do tempo. Parsopoulos e Vrahatis (2002) relatam que um valor um inicial em torno de 1.2 e gradual declínio para 0, seria uma boa escolha.

Para que o componente inercial (*w*) variasse a cada iteração do algoritmo Eberhart e Shi (2000) desenvolveram a seguinte equação:

$$w^{it} = \left(w_{ini} - w_{fin}\right)\frac{\left(R - it\right)}{R} + w_{fin} \qquad (6)$$

onde:
w_{ini} = valor inicial para o coeficiente de inércia
w_{fin} = valor final para o coeficiente de inércia
R = número de iterações

Para facilitar a convergência das partículas em suas últimas interações, Chatterjee e Siarry (2006) propuseram a equação de variação não-linear para o componente inercial.

$$w^{it} = \left\{\frac{\left(R - it\right)^{nl}}{R^{nl}}\right\} * \left(w_{ini} - w_{fin}\right) + w_{fin} \qquad (7)$$

Onde:
nl = coeficiente de não linearidade
w_{ini} e w_{fin} = são definidos previamente
R = número de iterações

Para testar o enxame de partículas, Prado e Saramago (2005) utilizaram c_1 = c_2 = 2 e w_0 = 0.729, a cada iteração, atualizando a inércia utilizando a seguinte equação:

$$w_{new} = f_w \, w_{old} \tag{8}$$

onde:

f_w = fator de redução, constante entre 0 e 1.

3.4.3 Coeficiente de enxugamento ou fator de constrição

Desempenhando função semelhante ao parâmetro v_{max} , o coeficiente é utilizado para controlar a grandeza das velocidades, ajudando a assegurar a convergência, segundo Clerc e Kennedy (2002).

$$\boldsymbol{v_i}^{\boldsymbol{it+1}} = k^{it} [v_i^{it} + c_1^{it} * \text{rand}()^{it} * (pbest_i^{it} - x_i^{it}) + c_2^{it} * \text{rand}()^{it} * (gbest^{it} - x_i^{it})] \tag{9}$$

$$k^{it} = \frac{2}{|\, 2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\,|} \tag{10}$$

$$\varphi = c_1 + c_2, \qquad \varphi > 4. \tag{11}$$

3.4.4 Centróide do enxame

A “Atração Social” ou centróide do enxame (c) como definida por Albrecht (2004), tem como objetivo evitar a dispersão das partículas. Acrescentado o centróide na equação da velocidade, faz com que as partículas concentrem-se próximas ao centro do enxame, ou seja, são influenciadas pelo grupo.

O centróide do enxame é calculado da seguinte forma:

$$C = \frac{\sum_{i=1}^{N} x_i}{N} \tag{12}$$

3.4.5 Coeficiente de aceleração c_3

Em 2004 um novo coeficiente de aceleração (c_3) foi introduzido em função do centróide.

$$c_3{}^{it} = \left(c_{3\,fin} - c_{3ini}\right)\frac{it}{R} + c_{3ini} \tag{13}$$

Onde:
c_{3ini} = valor inicial para o coeficiente de atração da partícula pelo enxame
c_{3fin} = valor final para o coeficiente de atração da partícula pelo enxame

Os valores iniciais e finais dos coeficientes de aceleração dependem muito do problema em questão. Contudo, deve-se observar para que φ seja maior que 4, quando utilizado o fator de constrição. De tal modo que o termo proposto que deverá ser adicionado à equação da velocidade será c_3 * rand() * (C - x_i) (MÜLLER, 2007).

3.5 Outras Versões do Enxame de Partículas

Sempre com o objetivo de melhorar o desempenho do método de otimização por enxame de partículas, novas versões do algoritmo vão surgindo tais como, enxame de partículas por vizinhanças, binário, cooperativo, entre outras.

A seguir uma simples conceituação sobre algumas versões do EP, as quais poderão ser objetos de estudo em trabalhos futuros.

3.5.1 Enxame de Partículas por Vizinhança

Com a finalidade de aumentar o potencial de exploração da partícula no espaço de busca, Eberhart e Shi (2001) desenvolveram o conceito de vizinhança, o qual refere-se à topologia dos relacionamentos entre as partículas, ou seja, o enxame é dividido em vizinhanças, as quais agem de modo independente e seus indivíduos interagem com sua melhor posição encontrada (*pbest*) e com a melhor posição encontrada por sua vizinhança (*lbest*), que sua vez, substitui o *gbest* da versão global do algoritmo.

Para verificar qual a melhor posição da vizinhança, a partícula *i* compara o valor encontrado pela função objetivo das partículas *i*-1 e *i*+1 a sua melhor posição, caso algum vizinho esteja melhor posicionado, esta atualiza sua posição.

As figuras X e Y abaixo, mostram como as partículas utilizam o g*best* e o *lbest* em busca da melhor posição, tanto no algoritmo global quanto no algoritmo local.

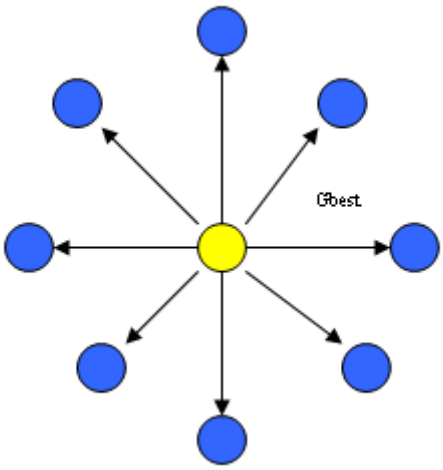

Figura 20 – *Gbest* sendo compartilhado com o enxame.

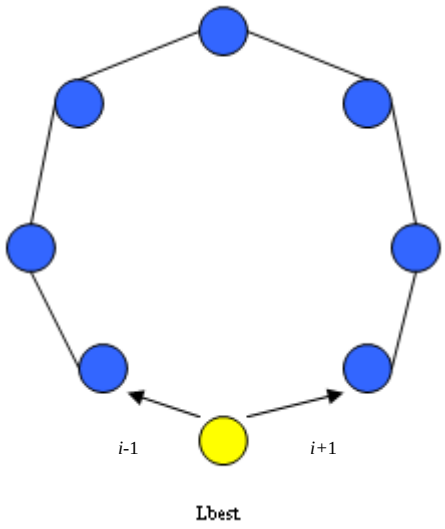

Figura 21 – *Lbest* sendo compartilhado com a vizinhança.

3.5.2 Enxame de Partículas Binário

Para Eberhart e Kennedy (2001), o enxame de partículas binário funciona como um jogo de decisões a fazer, sim/não ou verdadeiro/falso, ou seja, duas opções ou dois estados em que as partículas podem estar.

Neste método, Carvalho e Ludemir (2004) informam que os vetores da velocidade e da posição são compostos por 0 e 1, sendo que o da velocidade é tratado como uma probabilidade.

A equação (1) continua sendo utilizada para atualizar a velocidade e o v_{max}, o qual é o limitador de velocidade, fica entre -4 – 4 conforme aconselha Kennedy *et al* (2001).

Resultados mostraram que o enxame de partículas binário chega a resultados mais rápidos e lida melhor com alta dimensionalidade, comparado a outros métodos de otimização, como o algoritmo genético.

3.5.3 Enxame de Partículas Cooperativo

Segundo Carvalho e Ludemir (2006), o cooperativismo entre as partículas envolve uma coleção de agentes que interagem pelo compartilhamento de informação durante a resolução do problema no espaço de busca que ao invés de utilizar apenas um enxame de *n*-dimensões com *s*-indivíduos, utiliza mais de um enxame com *s*-indivíduos, trabalhando cooperativamente entre si em busca da melhor otimização.

Bergh e Engelbrecht (2004) relatam que o enxame de partículas cooperativo melhora significantemente o desempenho do algoritmo original.

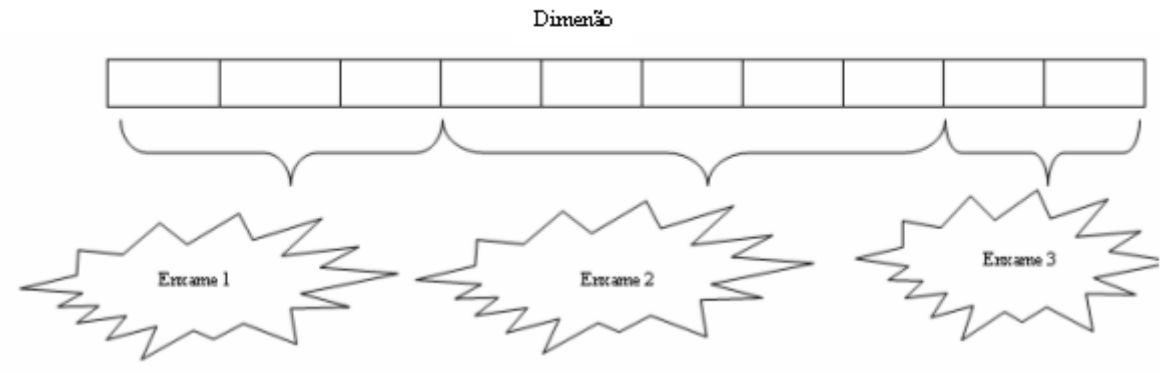

Figura 22 – Representação do enxame de partículas cooperativo.

4 O USO DO MÉTODO DO ENXAME DE PARTÍCULAS NA OTIMIZAÇÃO DA LATÊNCIA E CONSUMO DE ENERGIA DE UMA DE REDES-EM-CHIP

4.1 Simulador

Segundo Bruch *et al* (2007), o simulador de redes-em-chip desenvolvido por Kreutz através da linguagem orientada a objetos C++, descreve a arquitetura dos componentes internos dos roteadores de uma *noc* no nível de transação, a precisão ocorre em nível de ciclos e a cada período do relógio todos os componentes da rede são executados. Atualmente tem a capacidade de simular a otimização da latência e do consumo de energia através da execução de algoritmos meta-heurísticos que são utilizados como ferramentas de avaliação. Além do algoritmo do enxame de partículas, o algoritmo genético e o busca tabu, também estão implementados no simulador.

Para a simulação, alguns parâmetros necessitam ser configurados, tais como o tipo de roteador, o tipo de arbitragem, o que se deseja otimizar (latência ou energia), o algoritmo que será utilizado, a topologia da rede-em-chip, entre outros.

O simulador está em constante desenvolvimento, carecendo de uma interface gráfica, sendo as configurações dos parâmetros realizadas mediante a digitação de linhas de comando no próprio código da classe, fato que exige atenção e conhecimento, sobre o projeto, de quem estiver configurando o simulador.

4.2 Enxame de Partículas no Simulador

Os algoritmos meta-heurísticos podem e são utilizados nas mais diferentes soluções de problemas de modelagem computacional, ou seja, por não serem usados num problema específico, precisam ser adaptados a cada situação, através do entendimento do problema em questão, no caso, a otimização da latência e consumo de energia através das disposições dos nodos pelos roteadores de uma determinada rede-em-chip. O algoritmo de otimização busca, através da sua heurística, encontrar o menor valor para uma função objetivo, no caso de minização. A execução do simulador é a Função Objetivo, passando como parâmetro para a simulação a partícula que se deseja avaliar. Após a execução da simulação, é retornado uma avaliação desta simulação, por exemplo, a latência média gerada. O valor retornado como avaliação da simulação é o que estará sendo otimizado na *Noc*.

Cada partícula do enxame é um vetor de inteiros que representam a disposição dos nodos associando-os aos roteadores. Observe a tabela abaixo:

Tabela 1 – Exemplo de disposição dos nodos aos roteadores.

Partícula (Nodos)	5	10	13	7	9	0	15	11	12	2	14	1	8	3	6	4
Roteadores	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

A primeira linha é composta pelos nodos, os quais foram criados de forma aleatória e inseridos no vetor que representa uma partícula. A segunda linha representa os roteadores, os quais possuem sua ordem definida de 0 a 15, totalizando 16 roteadores. A explicação para que hajam 16 roteadores se dá devido ao fato das topologias malha e toróide que são utilizadas nos experimentos deste trabalho serem matrizes com dimensões 4x4.

Para o cálculo da atualização da partícula no espaço de busca, houve a necessidade de adaptações, devido a representação da solução, sendo que cada iteração da partícula é calculado do seguinte modo:

p = p + pBest e p = p + gBest, onde o pBest representa a posição da melhor solução local a ser copiada e gBest a posição da melhor solução global a ser copiada.

4.3 Pesquisa

A pesquisa para avaliação da otimização da latência e do consumo de energia tiveram cinco momentos: configuração da rede-em-chip, configuração do algoritmo do enxame de partículas, coleta dos dados, organização dos dados e análise dos resultados. Abaixo, encontram-se os parâmetros utilizados na pesquisa.

Configuração da rede-em-chip: roteador, arbitragem, memorização, controle de fluxo, topologia são configurados na classe *main* e a otimização (latência ou consumo de energia) é configurada na classe proxy_ctrl.

- Roteador: XY
- Arbitragem: Round Robin
- Memorização: Fifo
- Controle de Fluxo: Baseado em crédito
- Chaveamento: Pacote
- Topologia: Malha e Torpoide
- Otimização: Latência e Consumo de Energia

Configuração do algoritmo do enxame de partículas: tipo de algoritmo (básico, componente inercial e fator de constrição), quantidade de partículas, quantas iterações, quantidade de execuções para determinada quantidade de partículas, parâmetros de confiança (c1 e c2) e valores mínimos e máximos, todos são configurados na classe pso.

- Tipo de Algoritmo: Básico
- Quantidade de partículas: iniciando com 10 indo até 100
- Quantas iterações: iniciando com 10 indo até 100
- Quantidade de execuções: 10

- C1 e C2: 2 ambos
- Valores mínimos e máximos: 16 ambos

Coleta dos dados: os experimentos começaram com 10 partículas, 10 iterações, as quais foram variando de 10 em 10 até chegarem a 100 para a avaliação da latência e para o consumo de energia. As melhores avaliações foram coletadas.

Organização dos dados: os melhores resultados obtidos nos experimentos, foram inseridos em tabelas, as quais se encontram em anexo, e visando uma melhor análise dos dados, os melhores resultados também foram organizados em gráficos.

Análise dos resultados: depois de cada gráfico, foi descrito o melhor resultado obtido para cada quantidade de partículas utilizadas. Ao final de cada otimização com determinada topologia são apresentadas duas tabelas, uma com os melhores resultados da otimização, e a outra, com o melhor resultado com a sua respectiva disposição dos nodos aos roteadores, entre os melhores resultados obtidos, entretanto, é na conclusão deste trabalho que estão as análises da melhor topologia para a otimização da latência e consumo com o uso do enxame de partículas.

As execuções dos experimentos foram realizadas em um *Notebook* Sempron, de 1.8 GHz de processamento e 1 GB de memória RAM, com plataforma *Debian* ETCH.

4.4 Resultados

4.4.1 Resultados obtidos com a topologia malha para o consumo de energia

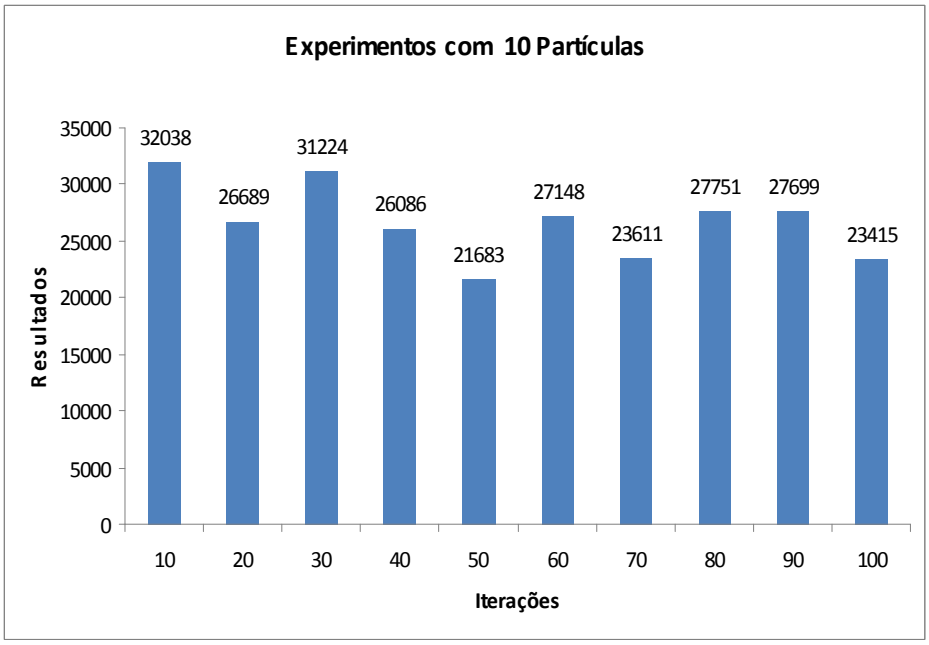

Gráfico 1 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 10 partículas.

A melhor otimização foi de 21683 com 50 iterações.

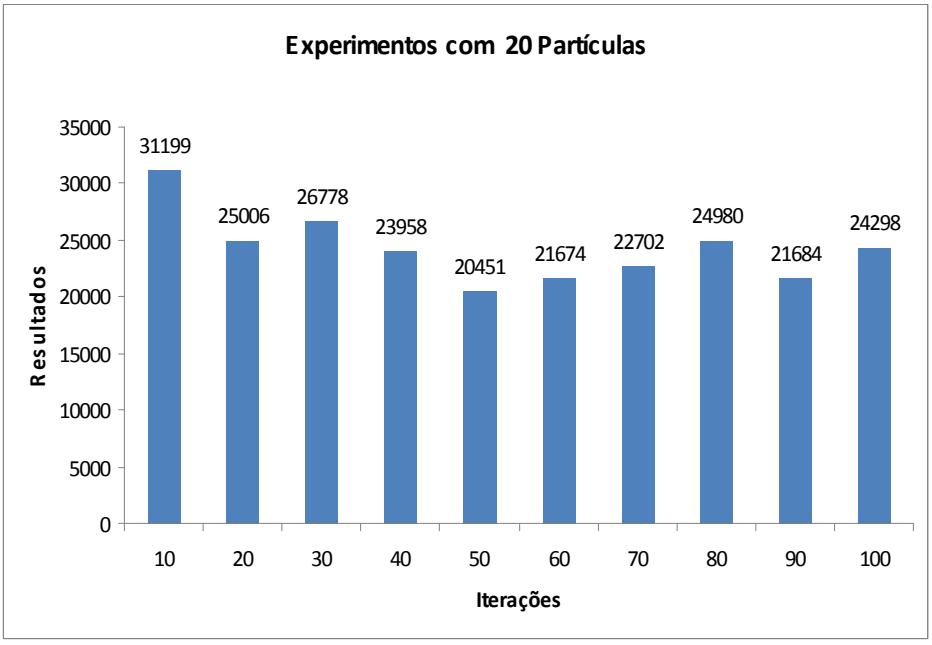

Gráfico 2 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 20 partículas.

A melhor otimização foi de 20451 com 50 iterações.

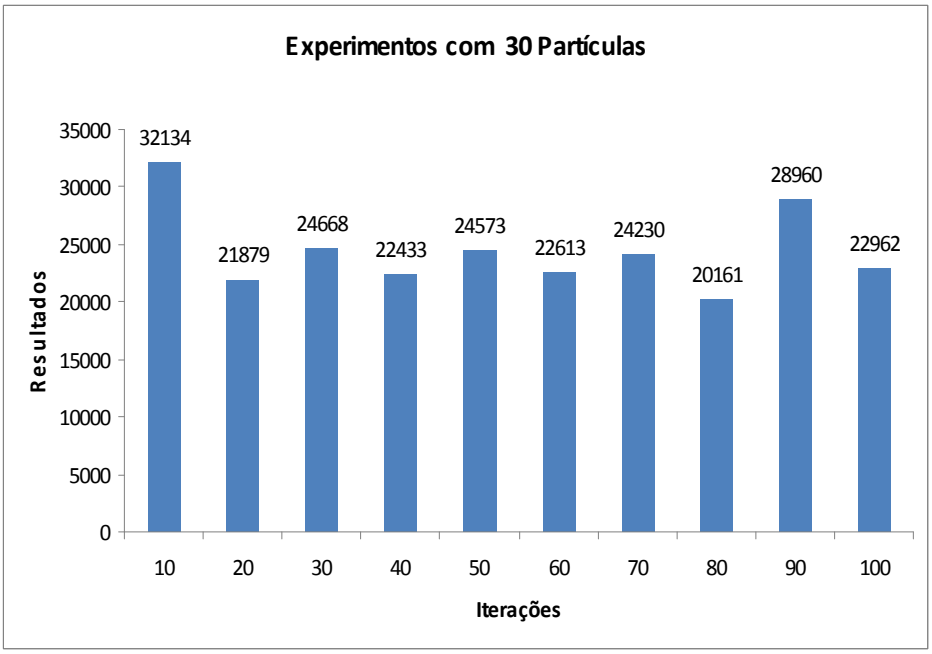

Gráfico 3 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 30 partículas.

A melhor otimização foi de 20161 com 80 iterações.

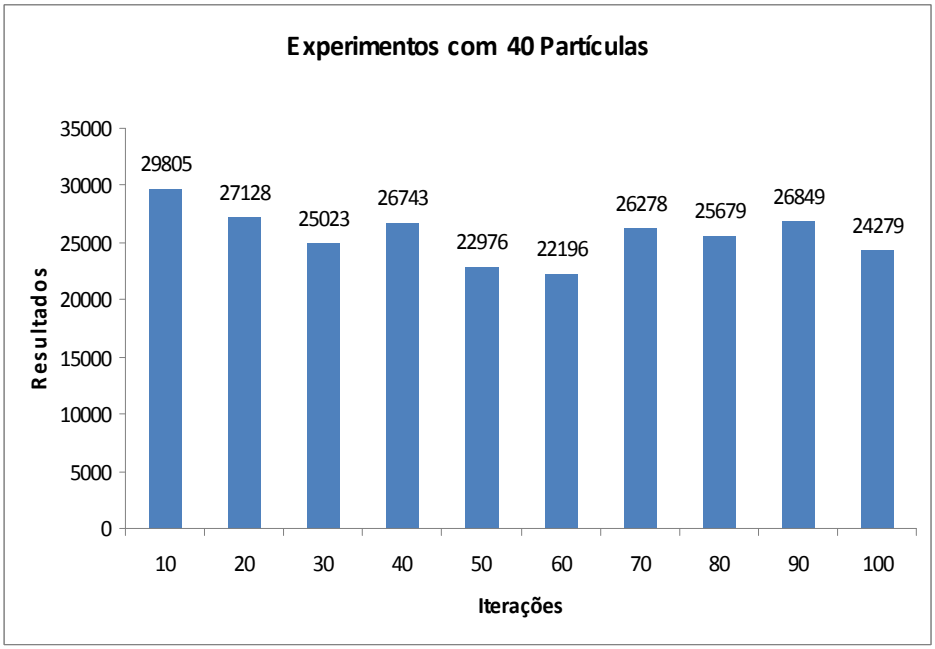

Gráfico 4 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 40 partículas.

A melhor otimização foi de 22196 com 60 iterações.

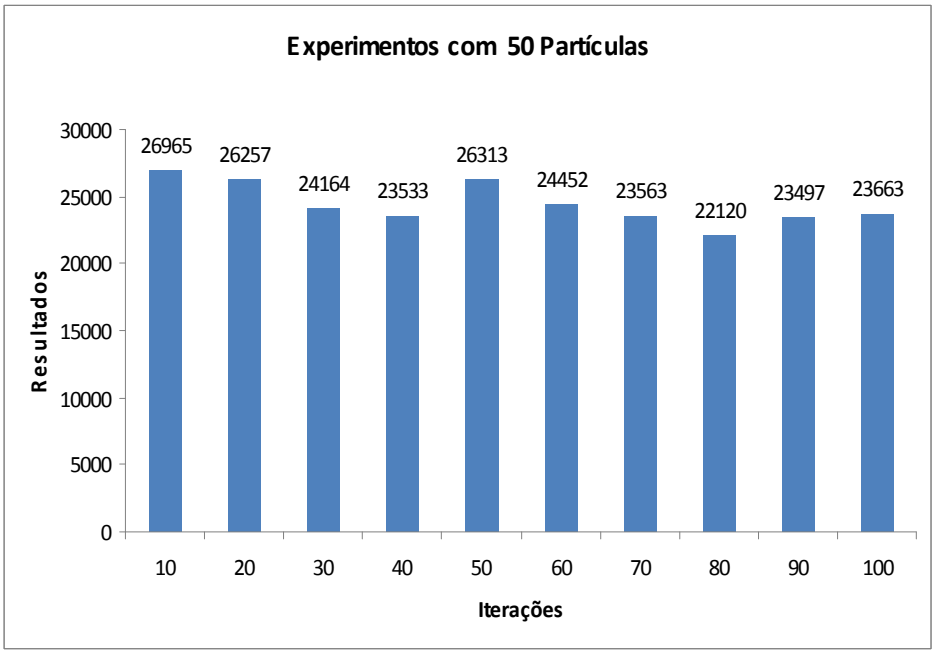

Gráfico 5 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 50 partículas.

A melhor otimização foi de 22120 com 80 iterações.

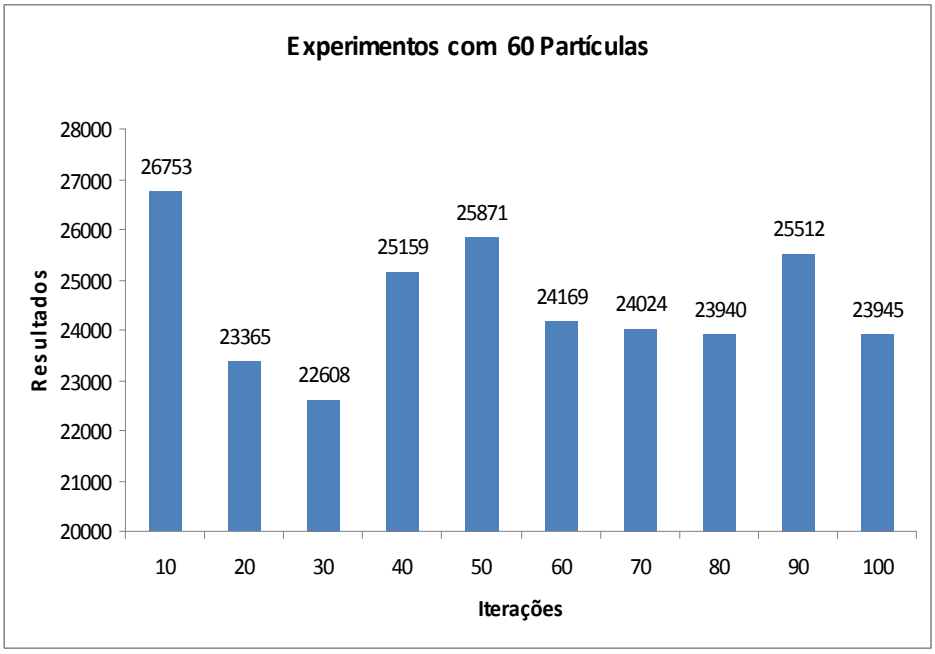

Gráfico 6 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 60 partículas.

A melhor otimização foi de 22608 com 30 iterações.

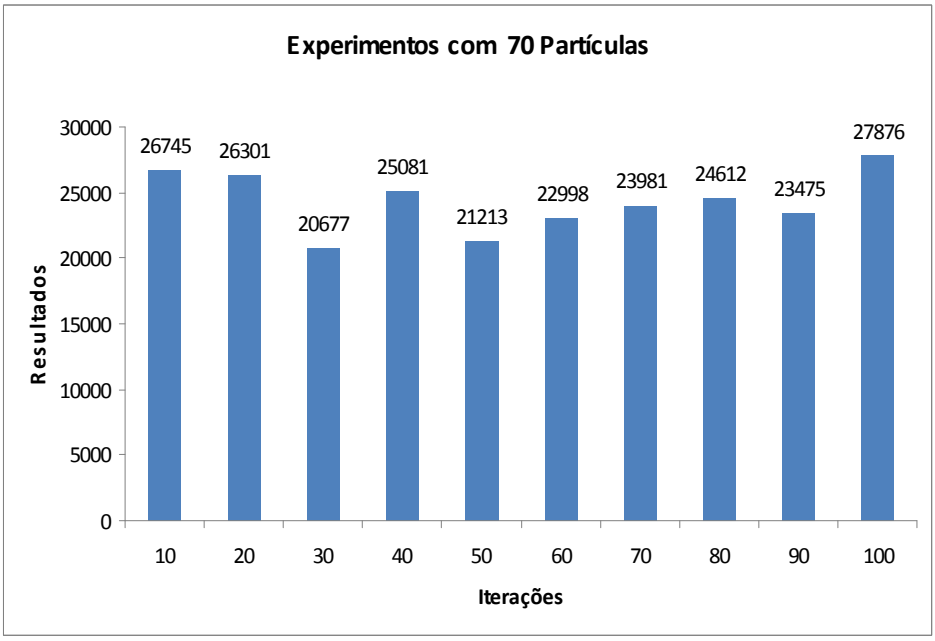

Gráfico 7 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 70 partículas.

A melhor otimização foi de 20677 com 30 iterações.

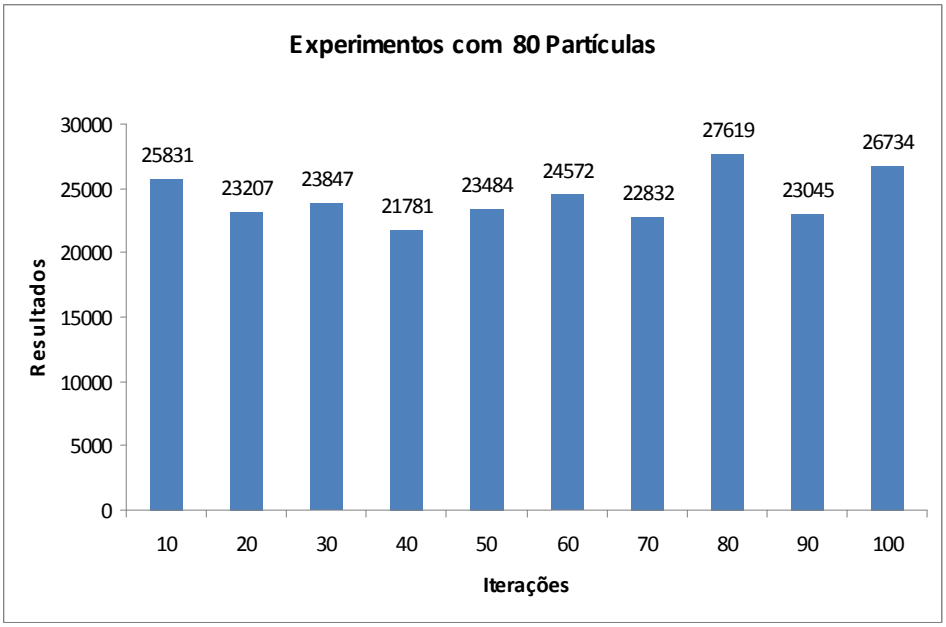

Gráfico 8 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 80 partículas.

A melhor otimização foi de 21781 com 40 iterações.

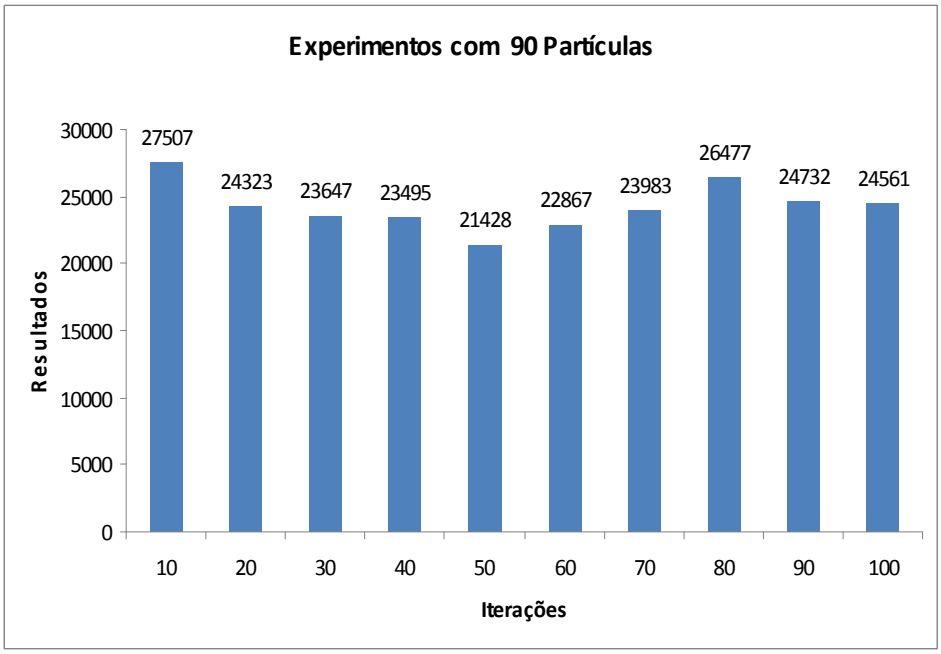

Gráfico 9 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 90 partículas.

A melhor otimização foi de 21428 com 50 iterações.

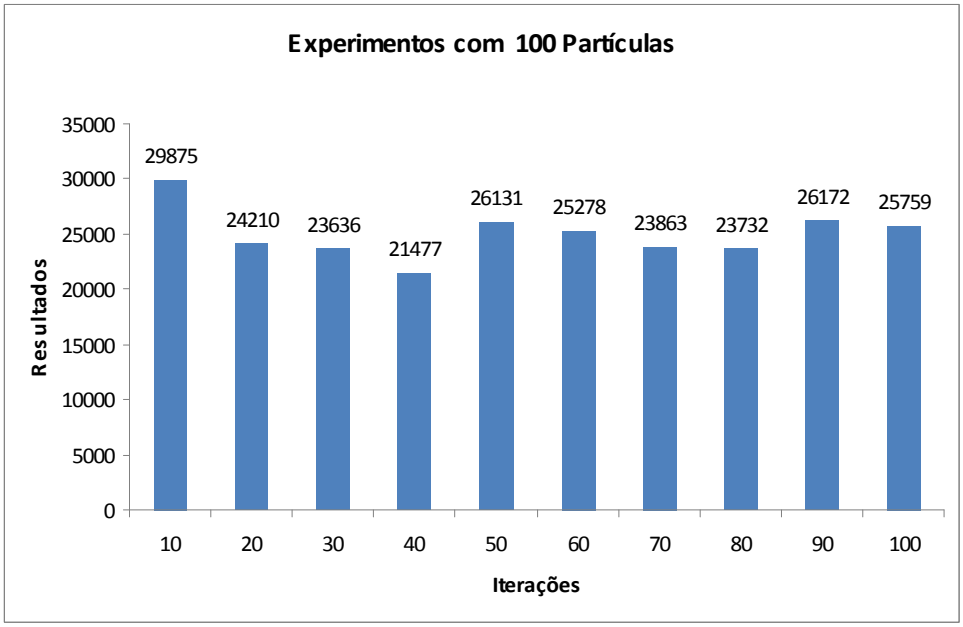

Gráfico 10 – Resultados obtidos com a topologia Malha otimizando o consumo de energia da rede-em-chip utilizando 100 partículas.

A melhor otimização foi de 21477 com 40 iterações.

Abaixo, encontram-se os melhores resultados obtidos pelas partículas com as suas respectivas iterações.

Tabela 2 – Melhores resultados da pesquisa entre 10 a 100 partículas com a topologia malha otimizando o consumo de energia.

Partículas	Iterações	Resultado
10	50	21683
20	50	20451
30	80	20161
40	60	22196
50	80	22120
60	30	22608
70	30	20677
80	40	21781
90	50	21428
100	40	21477

Tabela 3 – Melhor resultado da otimização com a topologia malha com a seqüência dos nodos dispostos aos roteadores.

Partículas	Iterações	Resultado	Disposição dos nodos aos roteadores
30	80	20161	4-6-13-1-7-2-10-14-3-5-15-11-8-0-9-12

4.4.2 Resultados obtidos com a topologia toróide para o consumo de energia

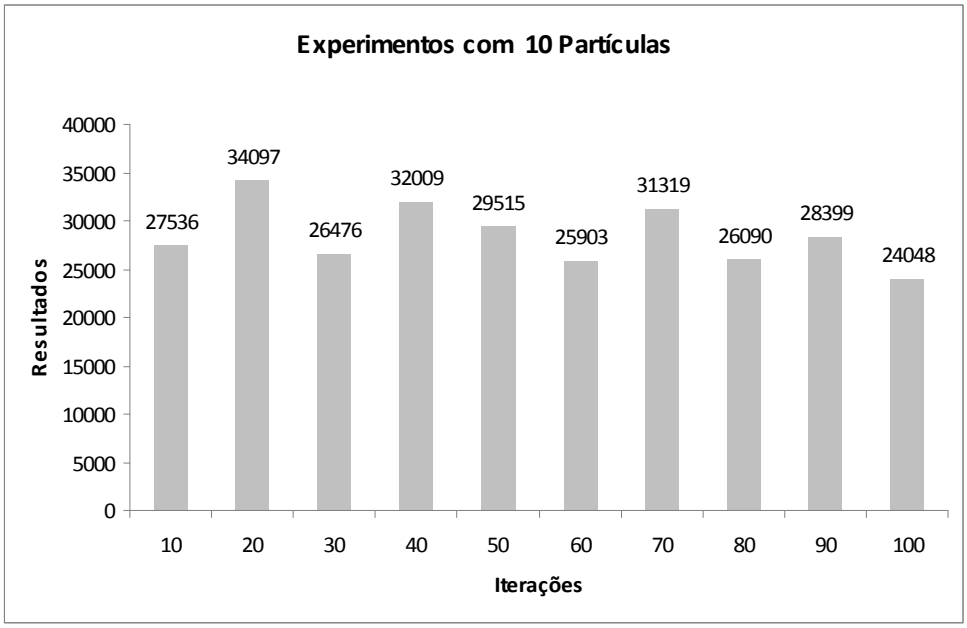

Gráfico 11 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 10 partículas.

A melhor otimização foi de 24048 com 100 iterações.

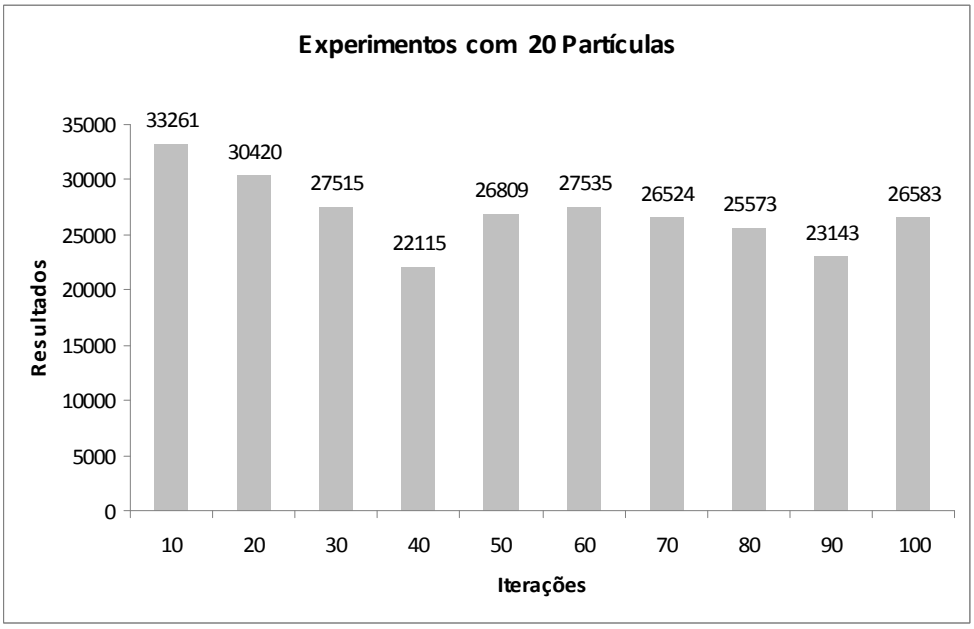

Gráfico 12 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 20 partículas.

A melhor otimização foi de 22115 (unidade) com 40 iterações.

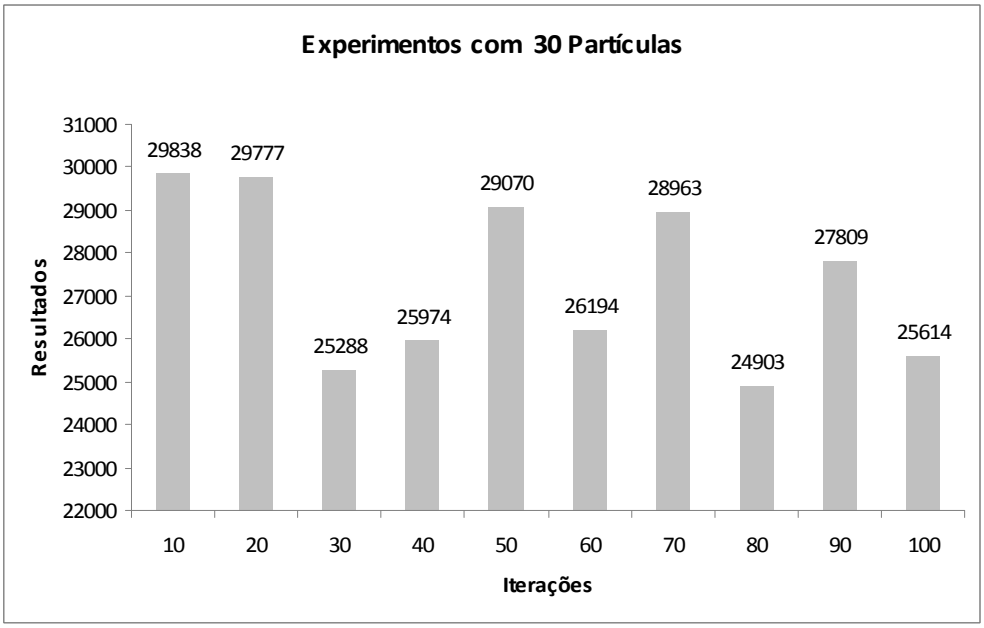

Gráfico 13 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 30 partículas.

A melhor otimização foi de 24903 com 80 iterações.

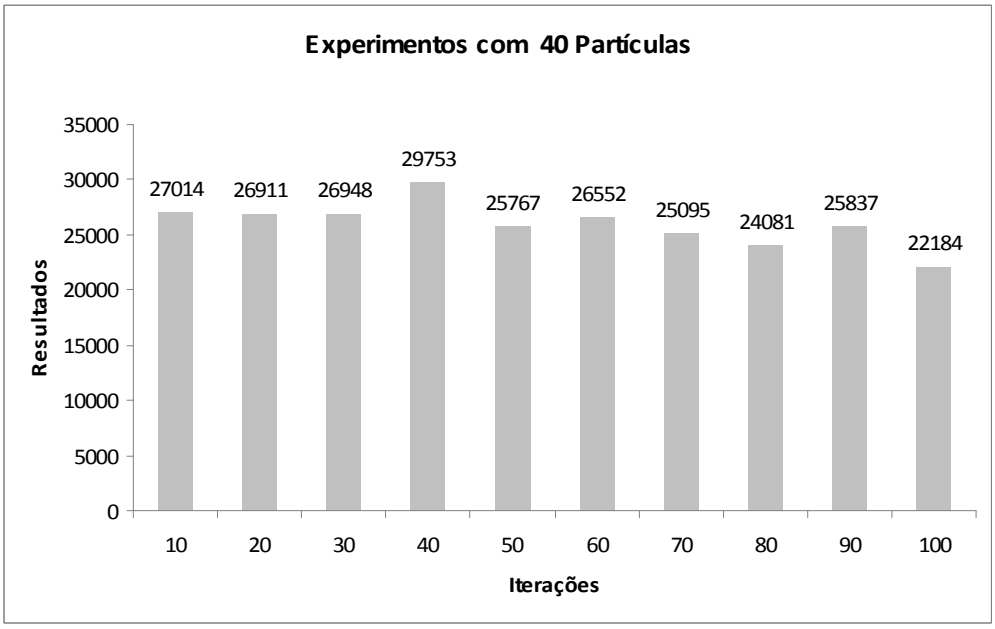

Gráfico 14 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 40 partículas.

A melhor otimização foi de 22184 (unidade) com 100 iterações.

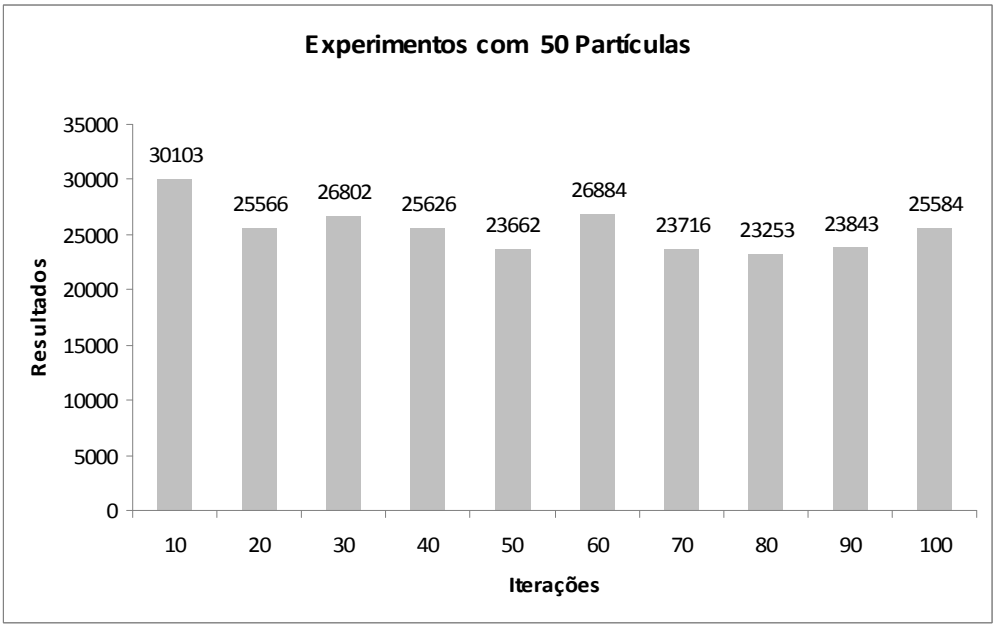

Gráfico 15 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 50 partículas.

A melhor otimização foi de 23253 com 80 iterações.

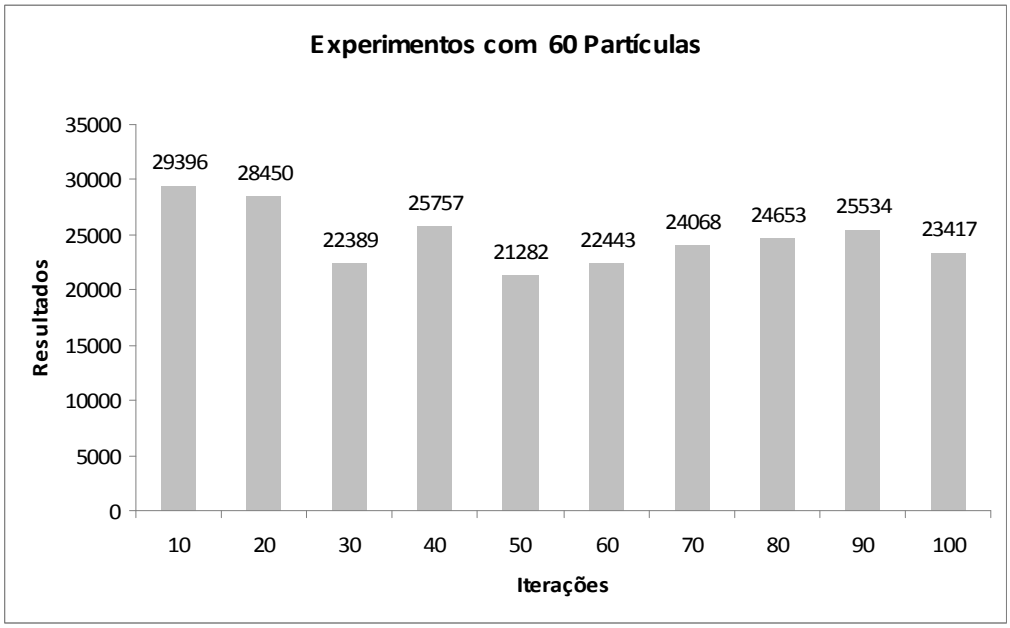

Gráfico 16 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 60 partículas.

A melhor otimização foi de 21282 (unidade) com 50 iterações.

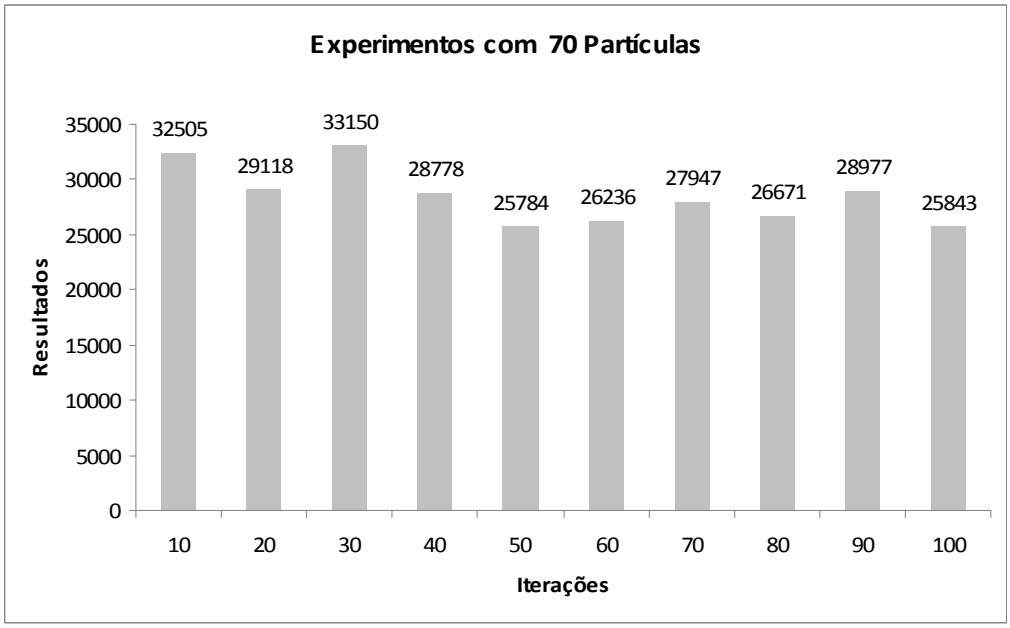

Gráfico 17 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 70 partículas.

A melhor otimização foi de 25784 com 50 iterações.

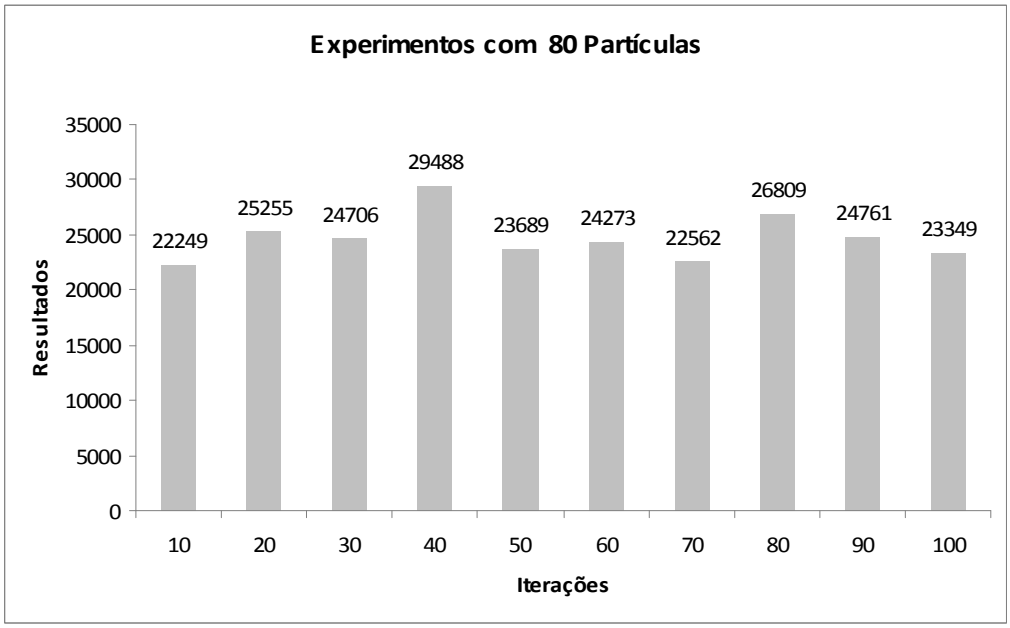

Gráfico 18 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 80 partículas.

A melhor otimização foi de 22249 (unidade) com 10 iterações.

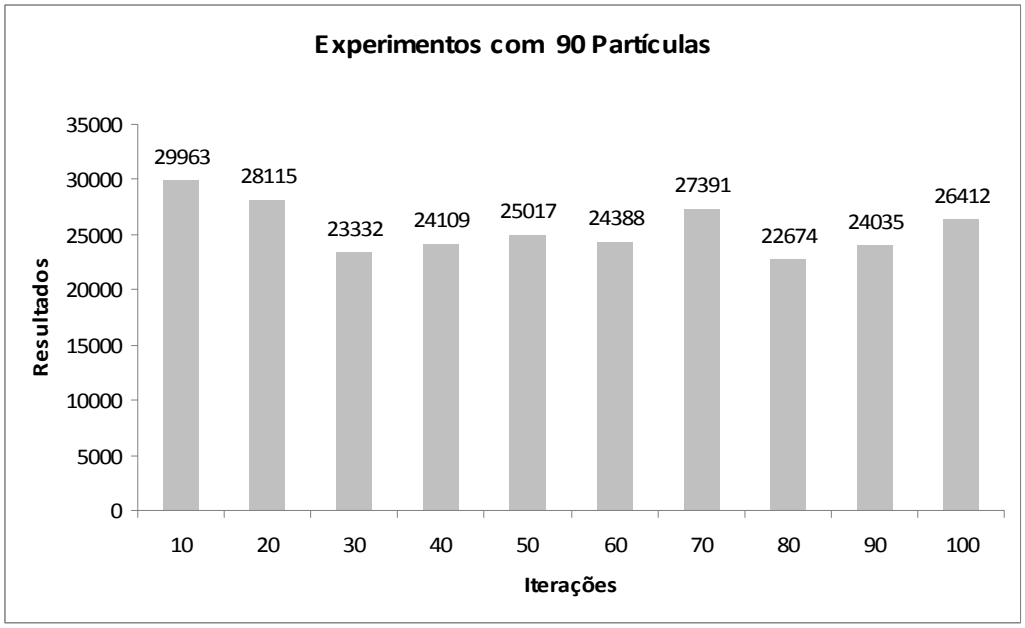

Gráfico 19 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 90 partículas.

A melhor otimização foi de 22674 com 80 iterações.

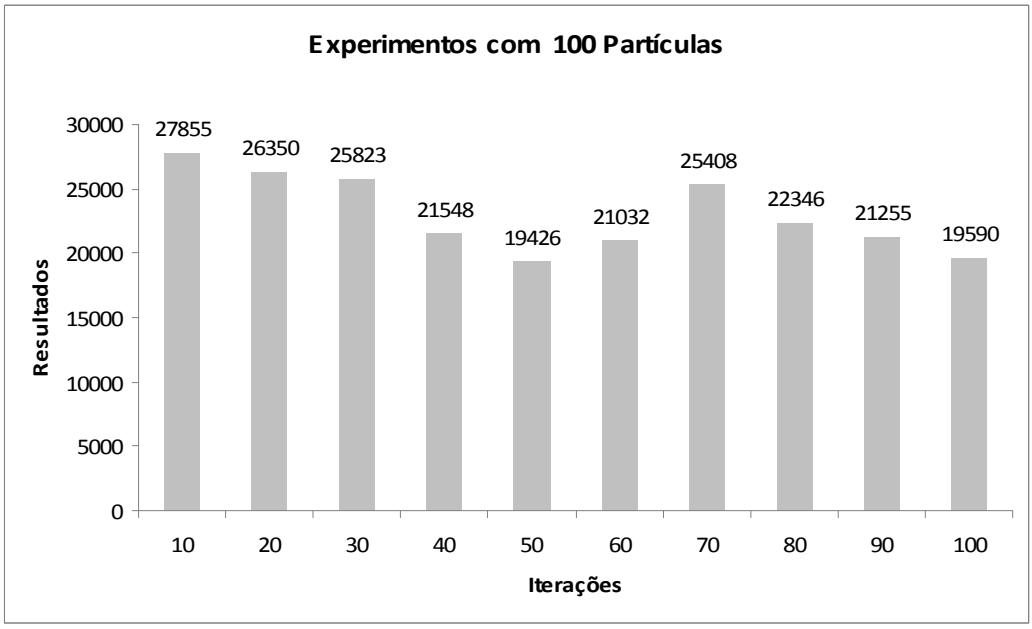

Gráfico 20 – Resultados obtidos com a topologia Toróide otimizando o consumo de energia da rede-em-chip utilizando 100 partículas.

A melhor otimização foi de 19426 com 50 iterações.

Abaixo, encontram-se os melhores resultados obtidos pelas partículas com as suas respectivas iterações.

Tabela 4 – Melhores resultados da pesquisa entre 10 a 100 partículas com a topologia toróide otimizando o consumo de energia.

Partículas	Iterações	Resultado
10	100	24048
20	40	22115
30	80	24903
40	100	22184
50	80	23253
60	50	21282
70	50	25784
80	10	22249
90	80	22674
100	50	19426

Tabela 5 – Melhor resultado da otimização com a topologia toróide com a seqüência dos nodos dispostos aos roteadores.

Partículas	Iterações	Resultado	Disposição dos nodos aos roteadores
100	50	19426	12-10-15-11-8-13-1-4-5-2-6-0-14-7-3-9

4.4.3 Resultados obtidos com a topologia malha para a latência

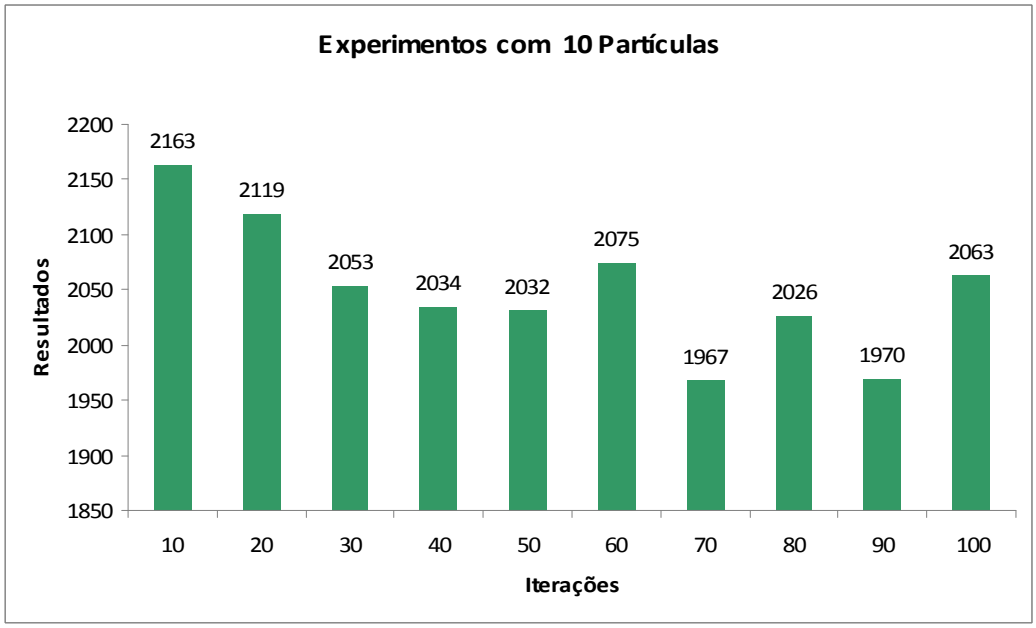

Gráfico 21 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 10 partículas.

A melhor otimização foi de 1967 com 70 iterações.

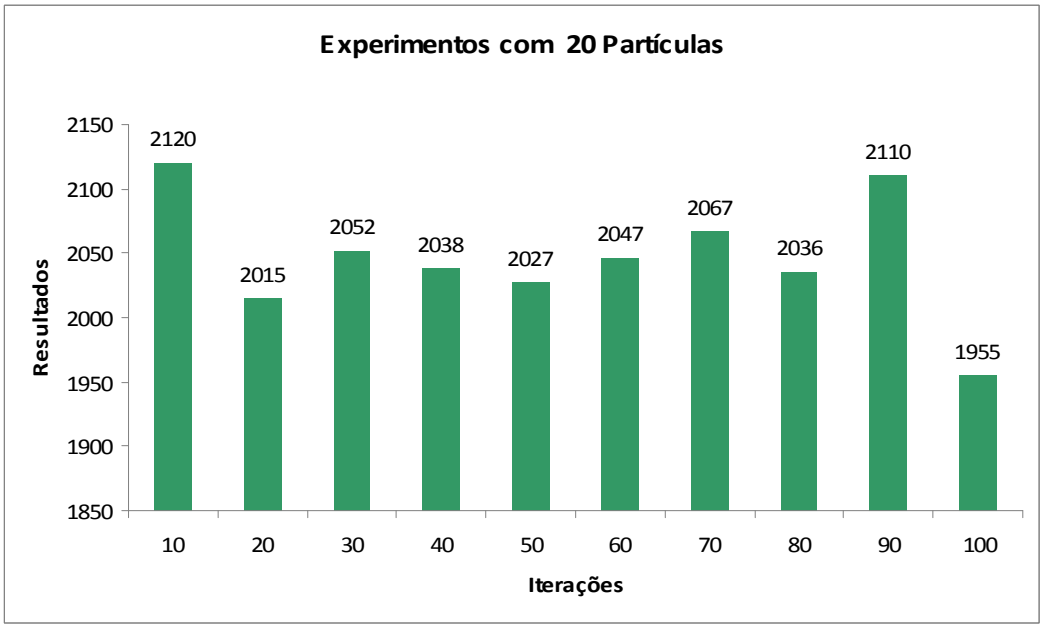

Gráfico 22 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 20 partículas.

A melhor otimização foi de 1955 com 100 iterações.

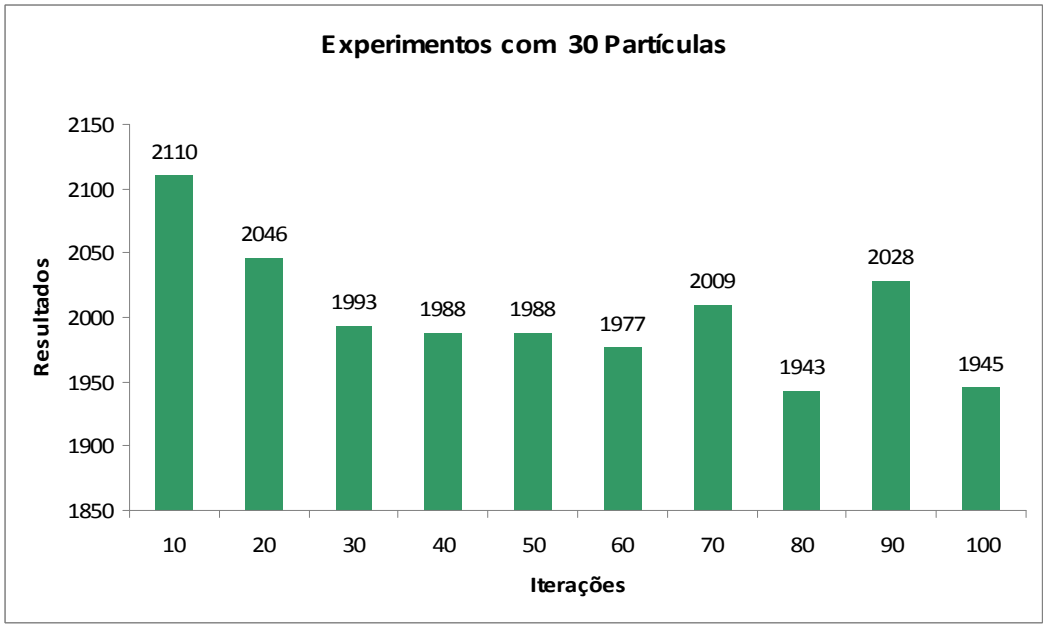

Gráfico 23 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 30 partículas.

A melhor otimização foi de 1943 com 80 iterações.

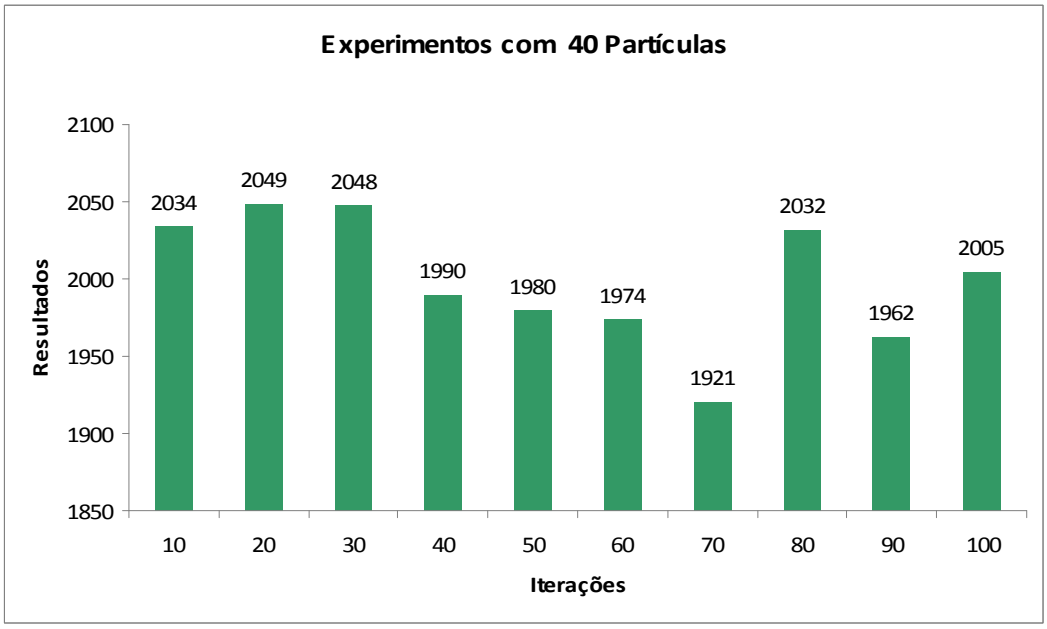

Gráfico 24 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 40 partículas.

A melhor otimização foi de 1921 com 70 iterações.

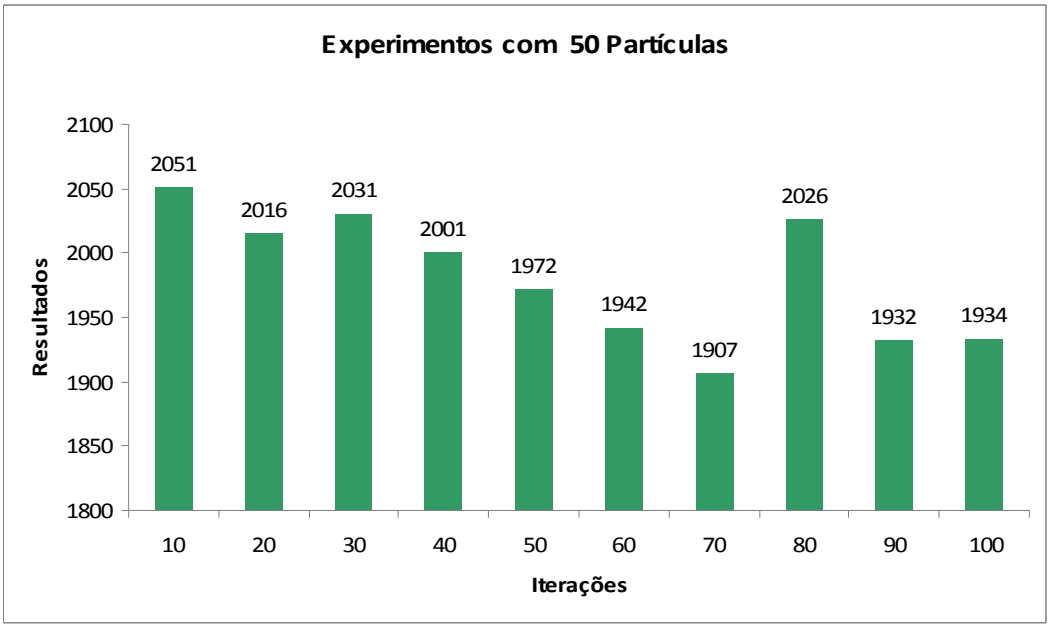

Gráfico 25 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 50 partículas.

A melhor otimização foi de 1907 com 70 iterações.

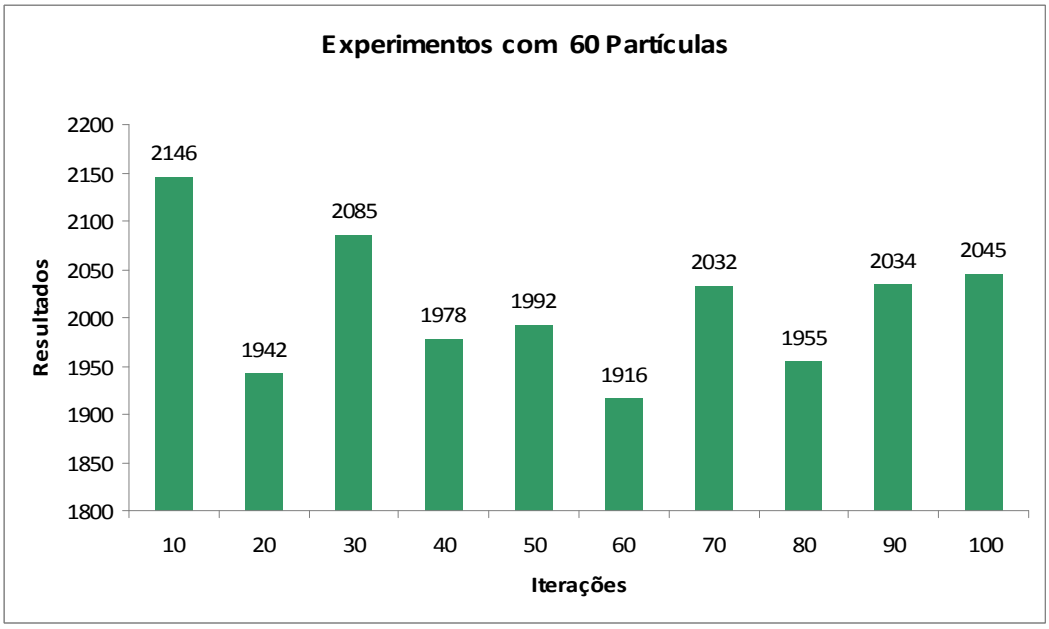

Gráfico 26 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 60 partículas.

A melhor otimização foi de 1916 com 60 iterações.

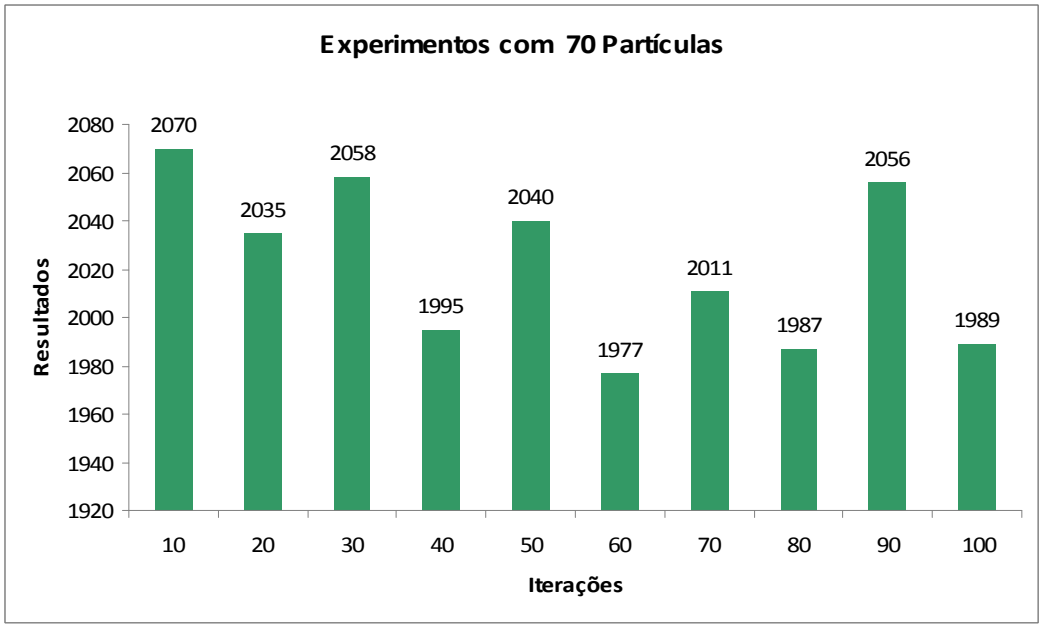

Gráfico 27 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 70 partículas.

A melhor otimização foi de 1977 com 60 iterações.

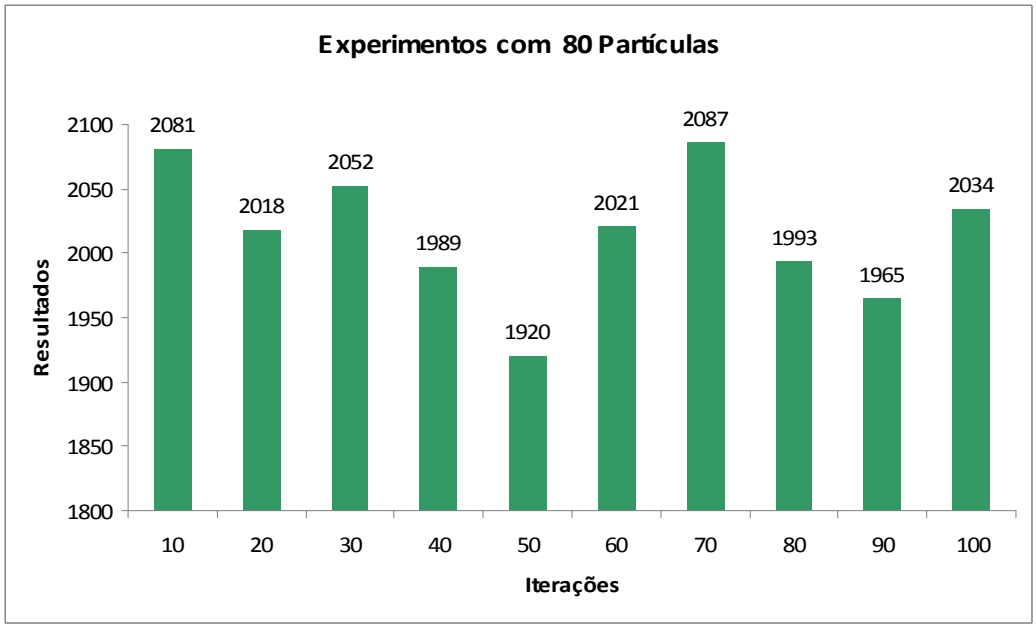

Gráfico 28 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 80 partículas.

A melhor otimização foi de 1920 com 20 iterações.

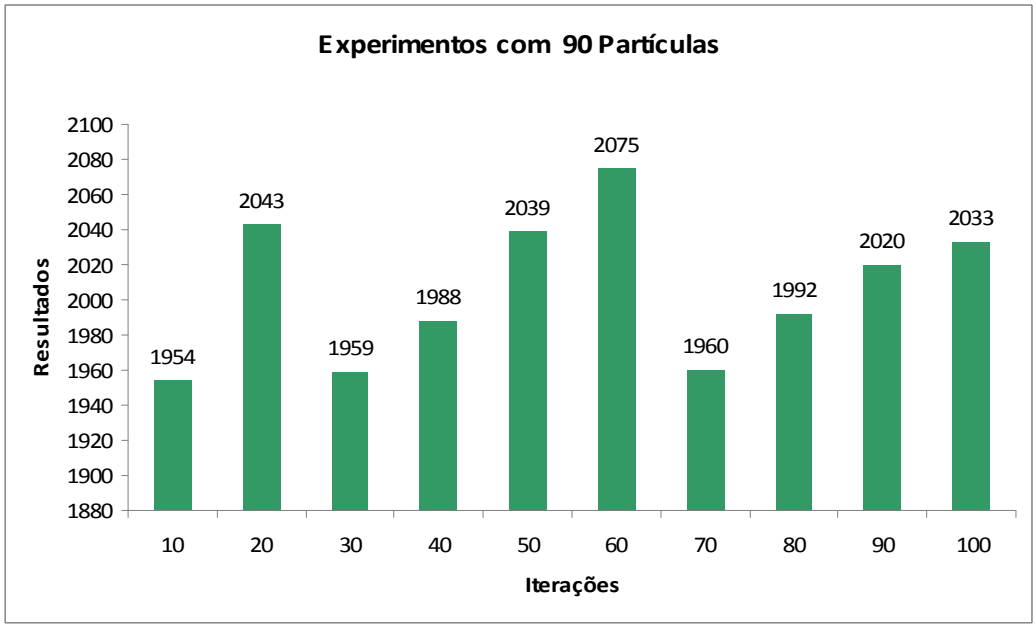

Gráfico 29 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 90 partículas.

A melhor otimização foi de 1954 com 10 iterações.

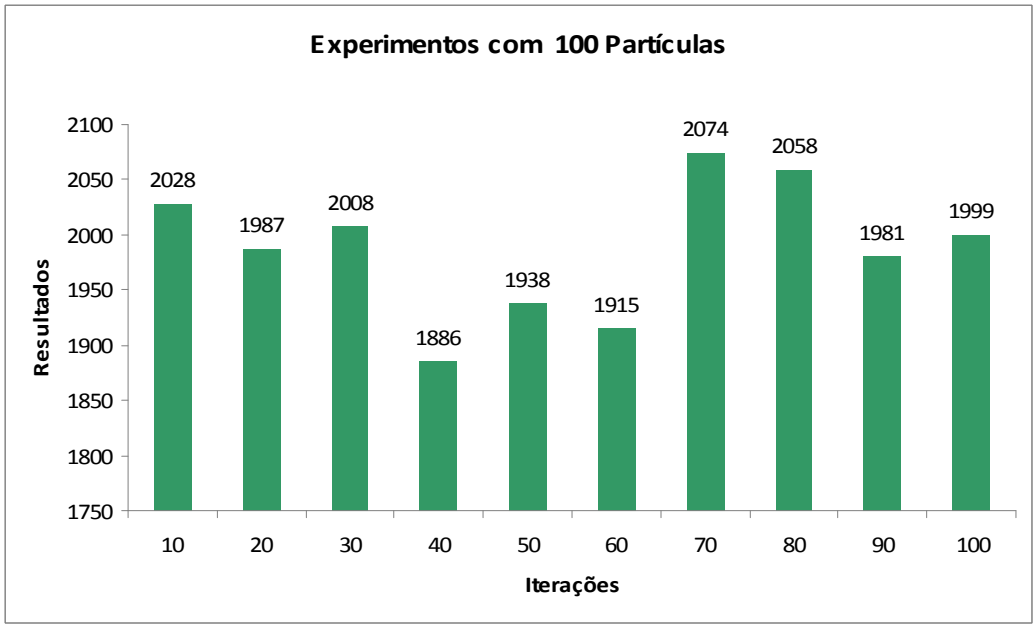

Gráfico 30 – Resultados obtidos com a topologia Malha otimizando a latência da rede-em-chip utilizando 100 partículas.

A melhor otimização foi de 1886 com 40 iterações.

Tabela 6 – Melhores resultados da pesquisa entre 10 a 100 partículas com a topologia malha otimizando a latência.

Partículas	Iterações	Resultado
10	70	1967
20	100	1955
30	80	1943
40	70	1921
50	70	1907
60	60	1916
70	60	1977
80	20	1920
90	10	1954
100	40	1886

Tabela 7 – Melhor resultado da otimização com a topologia malha com a seqüência dos nodos dispostos aos roteadores.

Partículas	Iterações	Resultado	Disposição dos nodos aos roteadores
100	40	1886	1-0-8-12-6-4-5-9-14-7-13-10-2-3-15-11

4.4.4 Resultados obtidos com a topologia toróide para a latência.

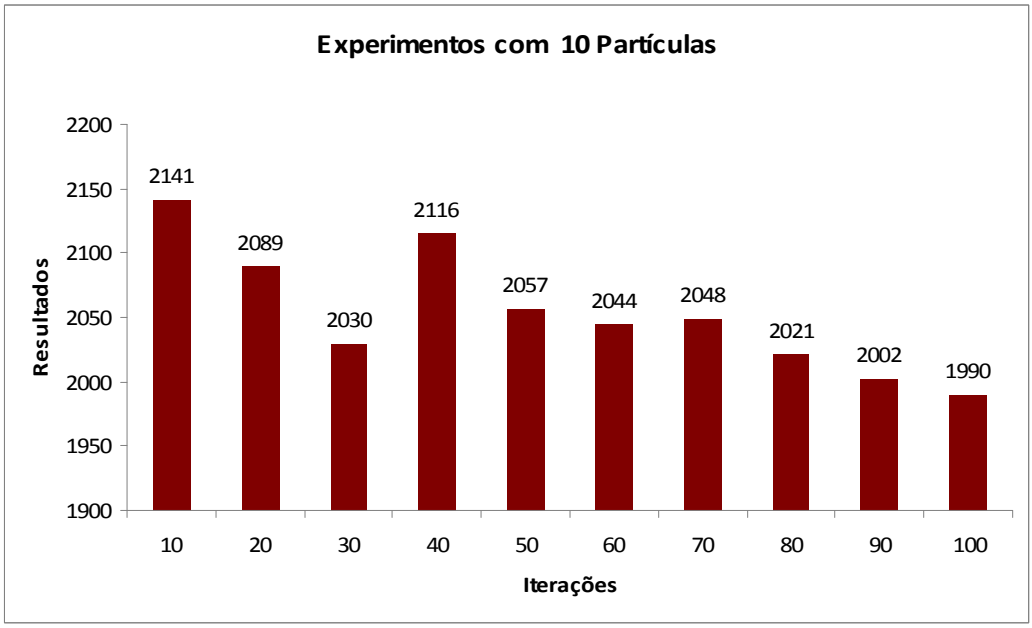

Gráfico 31 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 10 partículas.

A melhor otimização foi de 1990 com 100 iterações.

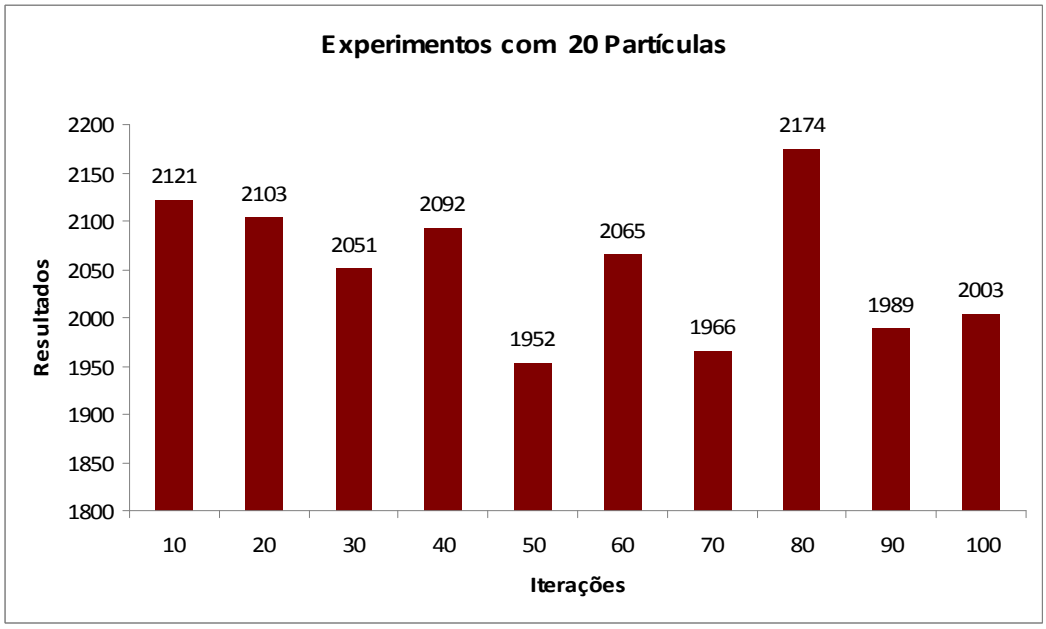

Gráfico 32 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 20 partículas.

A melhor otimização foi de 1952 com 50 iterações.

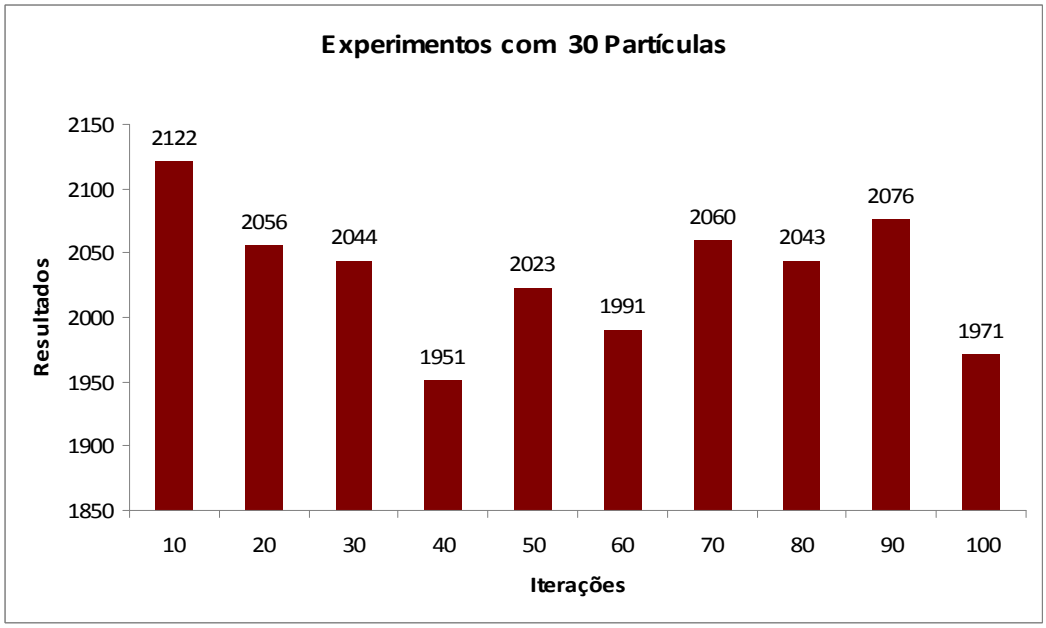

Gráfico 33 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 30 partículas.

A melhor otimização foi de 1951 com 40 iterações.

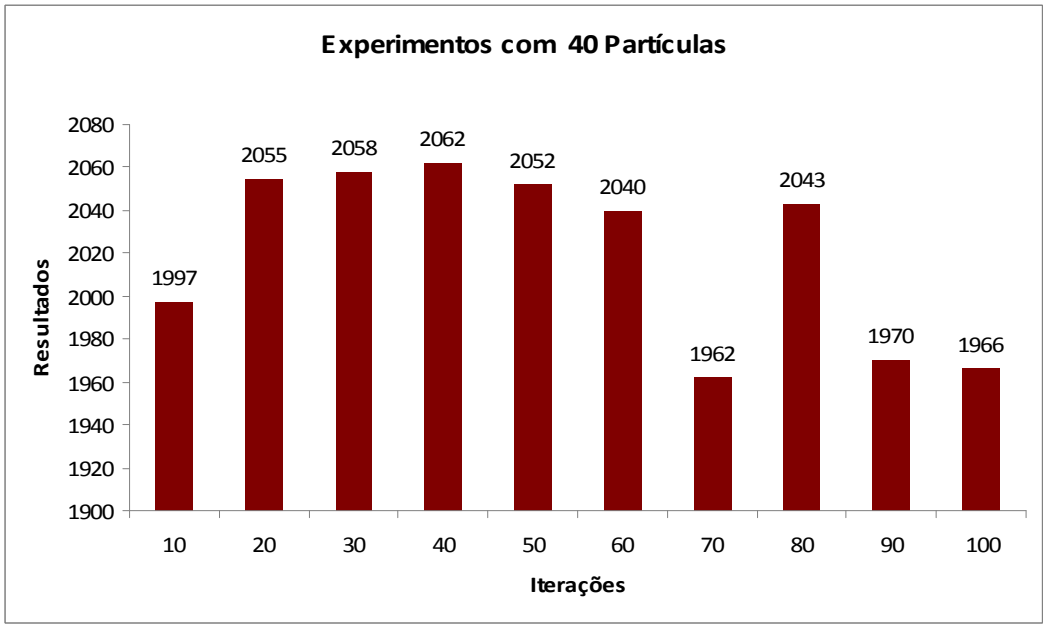

Gráfico 34 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 40 partículas.

A melhor otimização foi de 1962 com 70 iterações.

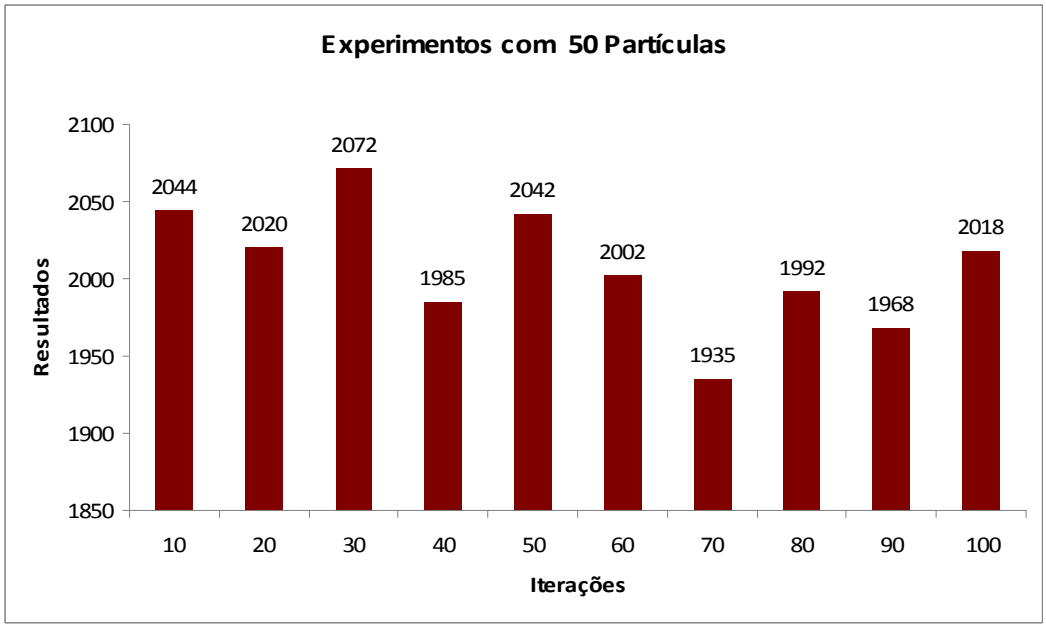

Gráfico 35 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 50 partículas.

A melhor otimização foi de 1935 com 70 iterações.

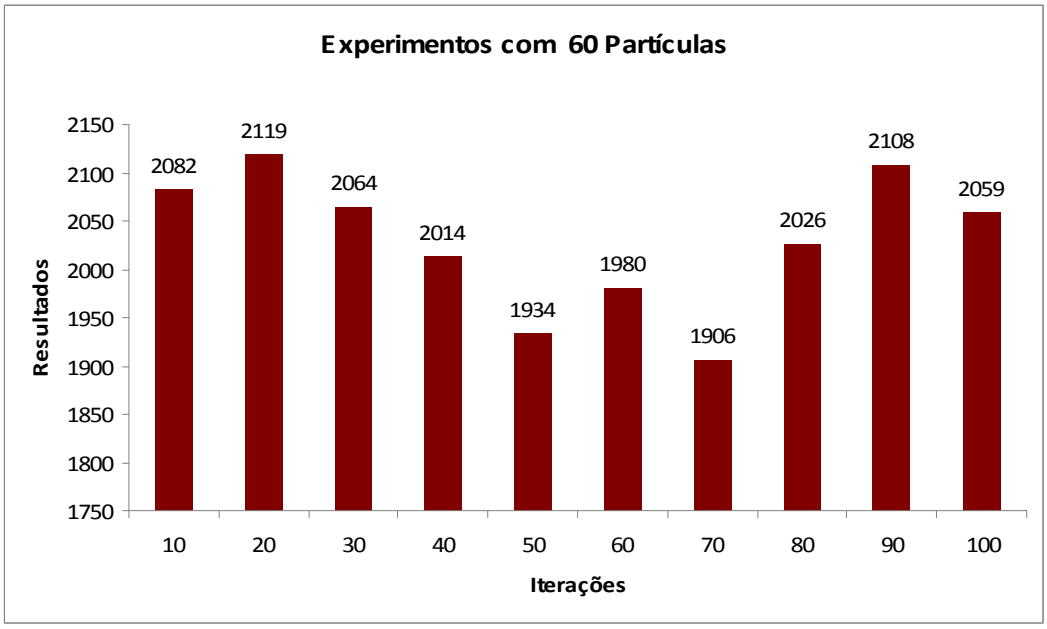

Gráfico 36 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 60 partículas.

A melhor otimização foi de 1906 com 70 iterações.

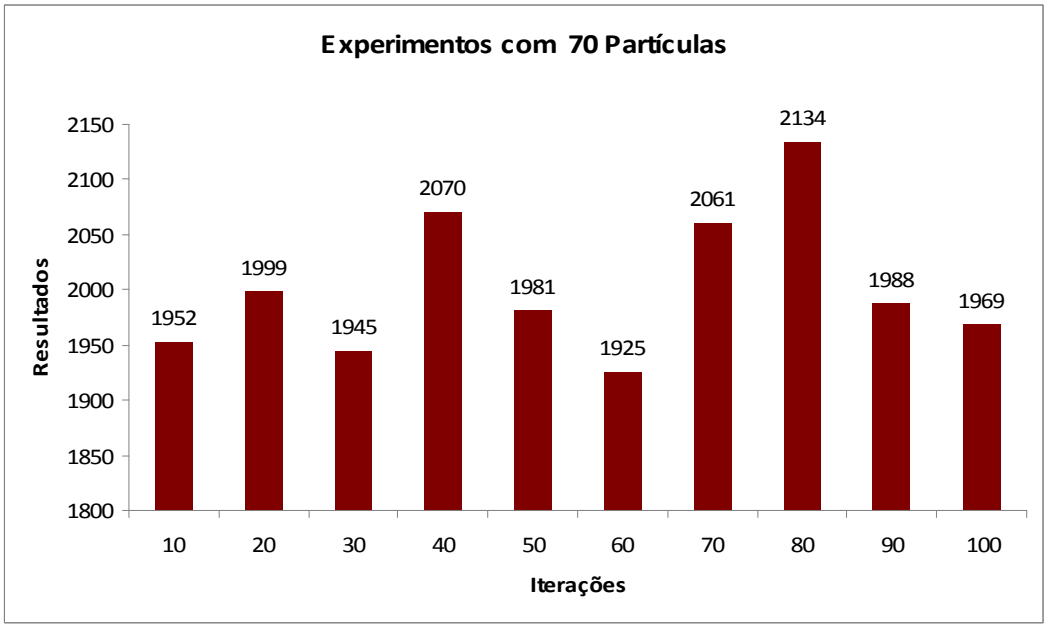

Gráfico 37 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 70 partículas.

A melhor otimização foi de 1925 com 60 iterações.

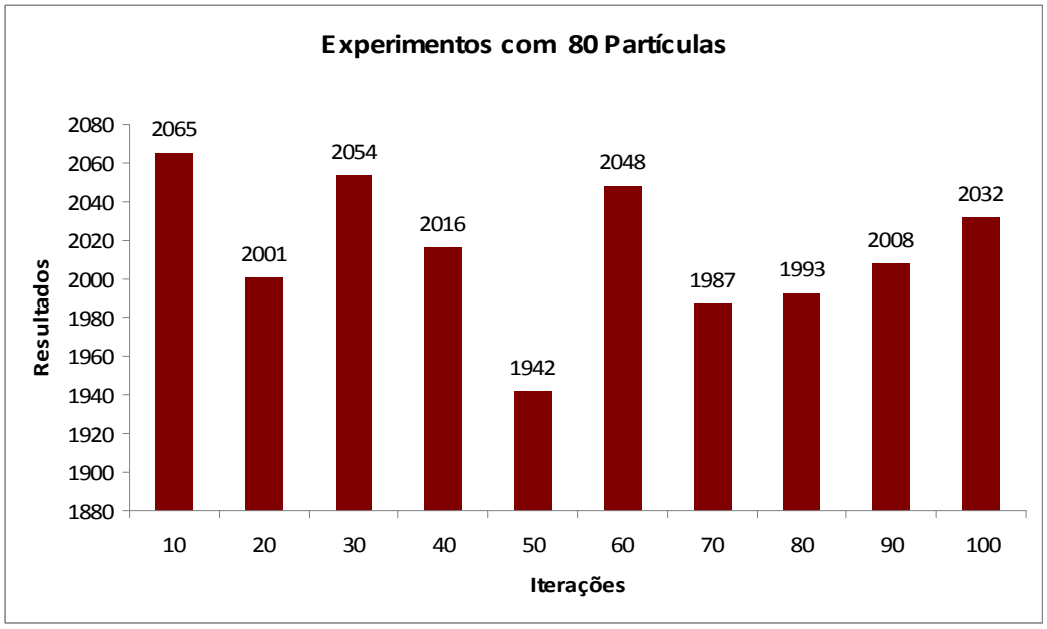

Gráfico 38 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 80 partículas.

A melhor otimização foi de 1942 com 50 iterações.

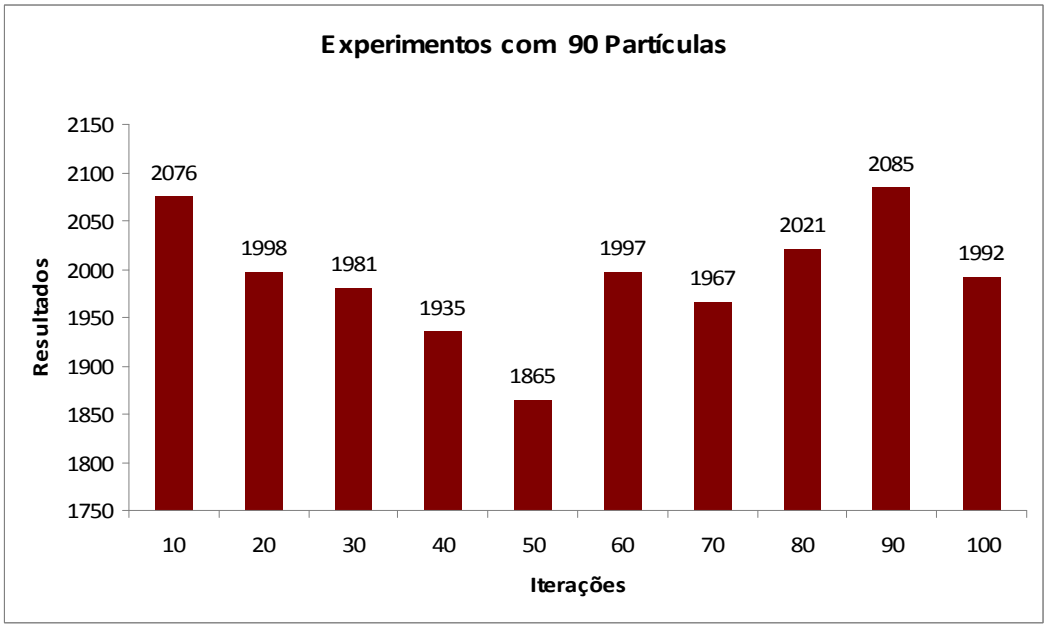

Gráfico 39 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 90 partículas.

A melhor otimização foi de 1865 com 50 iterações.

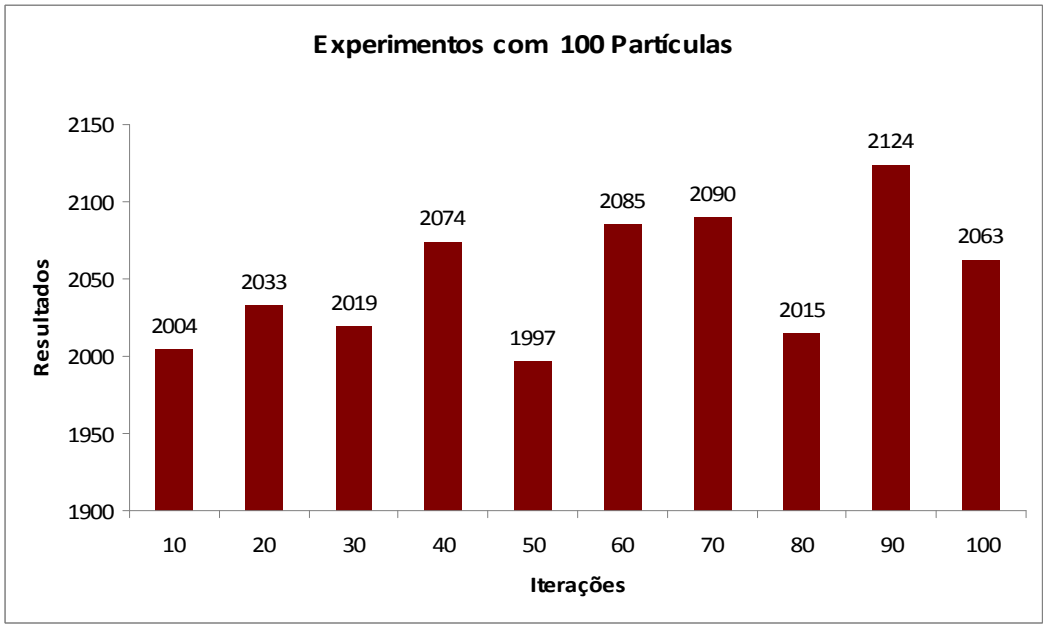

Gráfico 40 – Resultados obtidos com a topologia Toróide otimizando a latência da rede-em-chip utilizando 100 partículas.

A melhor otimização foi de 1997 com 50 iterações.

Tabela 8 – Melhores resultados da pesquisa entre 10 a 100 partículas com a topologia toróide otimizando a latência.

Partículas	Iterações	Resultado
10	100	1990
20	50	1952
30	40	1951
40	70	1962
50	70	1935
60	70	1906
70	60	1925
80	50	1942
90	50	1865
100	50	1997

Tabela 9 – Melhor resultado da otimização com a topologia toróide com a seqüência dos nodos dispostos aos roteadores.

Partículas	Iterações	Resultado	Disposição dos nodos aos roteadores
90	50	1865	0-8-9-13-1-4-12-14-6-7-15-11-2-3-10-5

CONCLUSÃO

Inicialmente, é importante destacar que a utilização da simulação de arquiteturas de uma rede-em-chip via *software*, veio a calhar, pois devido ao alto custo financeiro para se testar a eficiência de determinadas técnicas de programação diretamente em *hardware,* tornaria inviável esta prática.

Este trabalho teve como objetivo principal verificar a eficiência do método do enxame de partículas na otimização da latência e do consumo de energia de uma rede-em-chip, posicionando os nodos aos roteadores, utilizando duas topologias diferentes, a malha e a toróide. Para a realização dos experimentos, foi utilizado um simulador de *Nocs*, o qual foi desenvolvido por Kreutz em C++, no qual foi acoplado o algoritmo meta-heurístico do enxame.

Na otimização do consumo de energia, verificou-se que a topologia toróide obteve como melhor resultado o valor de 19426 e a topologia malha teve como melhor resultado 20161. Tal resultado se dá devido ao fato da toróide gastar menos energia para que os nodos das extremidades se comunicarem sem a necessidade de nodos intermediários, entretanto há um maior custo nesta topologia, pois é necessário um maior número de enlaces entre os nodos.

Na otimização da latência gerada pela rede, a topologia toróide obteve como melhor resultado o valor de 1865, sendo o melhor resultado da topologia malha o

valor de 1886. Novamente verificou-se o melhor desempenho da toróide tendo a menor latência na comunicação entre os nodos da rede.

Em vista a todas as considerações, o uso do método enxame de partículas é indicado na otimização da latência e consumo de energia de redes-em-chip, podendo o enxame ter um melhor desempenho se empregado outras opções de configuração, como o componente inercial e o fator de constrição que não foram utilizados neste trabalho.

Como o simulador de *Nocs* também possui os algoritmo de busca tabu e algoritmo genético, já implementados, em uma nova oportunidade poderá ser feito uma análise do desempenho de otimização entre os três algoritmos.

REFERÊNCIAS

ALBRECHT, C.H. *Algoritmos evolutivos aplicados a síntese e otimização de sistemas de ancoragem.* COPPE/Ufrj, 2004.

BISCAIA, E. PINTO, J. SCHWAAB, M.; *Um Novo Enfoque do Método do Enxame de Partículas*. Rio de Janeiro, Rio de Janeiro, Brasil, Universidade Federal do Rio de Janeiro. 2004.

BRUCH, Jaison, PEREIRA, Tiago, ZEFERINO, César, KREUTZ, Marcio, SUSIN, Altamiro; *Avaliação de Desempenho de Rede-em-chip Modelada em System C*, Rio de Janeiro, Rio de Janeiro, Brasil. V Workshop em Desempenho de Sistemas Computacionais e de Comunicação.2004.

CARVALHO, Marcio, LUDEMIR, Teresa; *Otimização Por Enxame de Partículas* (PSO), Recife, Pernambuco, Brasil, Universidade Federal de Pernambuco. 2006.

CHATTERJEE, A.; SIARRY, P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & OR*, n. 33, 2006, p. 859-871.

CLERC, M.; KENNEDY, J. *The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Somplex space*. IEEE Transaction on Evolutionary Computation, v. 6, 2002, p. 58-73.

CULLER, D.; SINGH, J. P. *“Parallel Computer Architecture: a Hardware Software Approach”*. Los Altos, California, Morgan Kaufmann, 1998, 1100 p.

EBERHART R C, SHI, Y. Particle swarm optimization: development, applications and resources. In: *Proceedings of Congress on Evolutionary Computation*, Seoul, Korea, 2001, p.81-86.

EBERHART, R.C.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings *of Congress on Evolutionary Computation*, San Diego, 2000, p. 84-88.

ESMIN, Ahmed; *Estudo de Aplicação do Algoritmo de Otimização por Enxame de Partícula na Resolução de Problemas de Otimização Ligados ao SEP*. Itajubá, Minas Gerais, Brasil. Universidade Federal de Itajubá. 2005.

GOMES, A.; CREÃO, D. Enxame de Partículas Evolucionários. In: MIRANDA, Vladimiro. *Computação Evolucionária Fenotípica*. Faculdade de Engenharia do Porto, 2004.

KENNEDY, J.; EBERHART, R.C.; SHI, Y. *Swarm intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.

KENNEDY, J.; EBERHART, R.C.; SHI, Y. *Swarm intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.

KREUTZ, M. ZEFERINO, Cesar; *Redes-em-chip*: *Definições básicas*
LOMONACO, Marcelo; *Alocação de Conversores de Comprimento de Onda em Redes Parciais*. Rio de Janeiro, Rio de Janeiro, Brasil. Pontifícia Universidade Católica do Rio de Janeiro. 2006.

MEDEIROS, José; *Enxame de Partículas Como Ferramenta de Otimização em Problemas Complexos de Engenharia nuclear*. Rio de Janeiro, Rio de Janeiro, Brasil. Universidade Federal do Rio de Janeiro. 2005.

MELLO, Aline, MÖLLER, Leandro; *Arquitetura Multiprocessada em SoCs: Estudo de Diferentes Topologias de Conexão*. Porto Alegre, Rio Grande do Sul, Brasil, Pontifícia Universidade Católica do Rio Grande do Sul. 2003.

MÜLLER, Viviane; *Otimização de Layouts Industriais Através Do Método Enxame de Partículas.* Santa Cruz do Sul, Brasil, Universidade de Santa Cruz do Sul, 2007.

PARSOPOULO, K.E; VRAHATIS, M.N, *Particle Swarm Optimization Method in Multiobjective Problems*. In: Proceedings of the 2002, University of Patras, Greece, 2002.

PRADO, J.R.do; SARAMAGO, S.F.P. Otimização por Colônia de Partículas. *FAMAT em Revista*, n. 04, abr. 2005.

REGO, Rodrigo; *Projeto e Implementação de uma Plataforma MP-SoC usando System C*, Natal, Rio Grande do Norte, Brasil. Universidade Federal do Rio Grande do Norte. 2006.

SANTO, Frederico, ZEFERINO, César, SUSIN, Altamiro; Uma Arquitetura de Roteador Parametrizável para a Síntese de Redes-em-chip. IV Congresso de Brasileiro de Computação, Sistemas Embarcados. 2004.

SHI, Y.; EBERHART, R.C. A modified particle swarm optimizer. *Proceedings of the IEEE International Conference on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1998, p. 69-73.

SILVA, Elina; *Investigação do Comportamento Dinâmico e Avaliação de Estratégias de Identificação, Controle e Otimização de Um Reator FCC*. Curitiba, Paraná, Brasil. Universidade Federal do Paraná. 2006.

ZEFERINO, Cesar; *Introdução às Redes-Em-Chip.* Itajaí, Santa Catarina, Brasil, Universidade do Vale do Itajaí. 2003.

ZEFERINO, Cesar; *Redes de Interconexão para multiprocessadores.* Porto Alegre, Rio Grande do Sul, Brasil, Universidade Federal do Rio Grande do Sul. 1999.

Anexos

Tabelas com os resultados das disposições dos nodos aos roteadores.

Topologia malha otimizando o consumo de energia.

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**															
10	10	32038	0	10	4	7	9	15	5	14	12	11	8	6	2	13	1	3
10	20	26689	7	0	12	1	11	13	5	8	10	3	15	6	9	14	2	4
10	30	31224	1	15	14	12	7	13	5	10	6	4	11	3	0	9	2	8
10	40	26086	5	15	12	13	14	2	7	1	10	9	0	4	8	6	11	3
10	50	21683	11	9	8	12	13	7	14	1	4	0	6	10	3	5	15	2
10	60	27148	8	13	4	9	10	7	14	3	2	5	11	15	6	0	12	1
10	70	23611	13	4	8	3	11	12	0	5	15	7	14	10	2	6	9	1
10	80	27751	12	5	8	10	3	4	2	15	13	1	11	6	9	14	7	0
10	90	27699	7	10	0	11	2	6	12	3	13	15	1	4	5	8	9	14
10	100	23415	13	6	14	8	1	9	3	10	5	2	7	12	0	4	15	11

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**															
20	10	31199	9	0	5	2	11	10	1	15	13	7	8	6	4	14	3	12
20	20	25006	15	9	0	6	11	14	7	2	13	8	3	5	1	4	12	10
20	30	26778	7	6	13	0	5	15	3	12	9	1	11	10	4	8	2	14
20	40	23958	3	15	6	14	7	11	13	9	2	1	5	8	10	4	12	0
20	50	**20451**	4	8	2	7	13	1	11	3	10	14	5	6	15	9	12	0
20	60	21674	14	11	1	8	10	6	4	0	13	9	3	5	12	15	7	2
20	70	22702	3	11	7	9	6	1	15	8	0	4	2	13	14	10	12	5
20	80	24980	1	4	8	0	2	13	3	14	6	10	15	7	9	12	5	11
20	90	21684	8	12	0	5	2	9	14	6	3	11	13	1	4	7	15	10
20	100	24298	4	12	9	1	13	2	15	8	9	10	5	6	14	7	11	3

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**															
30	10	32134	13	2	12	11	6	15	8	5	0	3	4	10	7	9	14	1
30	20	21879	9	4	14	11	5	12	10	15	8	0	13	3	1	7	2	6
30	30	24668	2	9	5	8	3	14	0	4	11	10	12	1	7	15	6	13
30	40	22433	0	6	3	12	5	2	4	11	1	9	15	7	8	13	10	14
30	50	24573	7	14	11	5	6	10	1	12	8	15	3	0	9	13	2	4
30	60	22613	9	5	13	1	2	15	8	4	10	3	12	0	11	14	7	6
30	70	24230	9	4	11	13	1	6	14	12	8	5	2	15	0	10	3	7
30	80	20161	4	6	13	1	7	2	10	14	3	5	15	11	8	0	9	12
30	90	28960	3	10	8	13	0	4	12	11	9	7	2	14	1	15	5	6
30	100	22962	0	5	2	15	9	1	14	10	8	12	3	6	13	11	7	4

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
40	10	29805	8	4	9	1	5	15	11	12	14	2	13	0	7	10	6	3
40	20	27128	5	7	0	13	1	6	12	10	4	15	2	9	3	11	14	8
40	30	25023	10	2	14	6	4	9	0	7	11	5	13	3	1	12	8	15
40	40	26743	2	10	15	9	7	1	5	14	4	12	3	6	8	11	13	0
40	50	22976	10	3	7	8	11	14	1	12	0	5	13	4	2	15	6	9
40	60	22196	13	5	10	7	4	9	15	3	8	12	2	11	14	1	6	0
40	70	26278	5	3	10	12	4	0	14	11	13	7	9	15	1	8	2	6
40	80	25679	1	9	14	13	11	5	10	15	7	8	0	12	4	2	6	3
40	90	26849	4	8	3	14	15	7	12	13	5	1	11	2	0	9	6	10
40	100	24279	12	14	15	7	2	13	1	11	10	5	8	3	4	9	0	6

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
50	10	26965	4	14	10	13	7	3	15	8	5	12	1	0	9	2	6	11
50	20	26257	11	6	4	14	3	9	2	15	10	0	13	1	7	8	12	5
50	30	24164	7	3	4	13	15	11	9	0	6	14	1	8	2	10	5	12
50	40	23533	9	2	8	6	7	0	14	11	3	4	1	12	15	5	13	10
50	50	26313	5	3	15	7	4	10	0	9	1	11	8	12	13	2	14	6
50	60	24452	11	4	8	6	7	9	1	12	10	5	0	15	2	3	14	13
50	70	23563	2	5	0	10	11	1	9	7	15	14	6	13	3	4	12	8
50	80	22120	7	8	10	15	12	11	14	9	0	5	2	13	6	3	4	1
50	90	23497	15	6	1	0	9	2	10	4	5	13	3	11	12	8	14	7
50	100	23663	10	13	1	6	12	5	9	2	11	15	0	4	8	3	14	7

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
60	10	26753	11	12	9	14	3	1	15	2	5	8	0	4	10	7	6	13
60	20	23365	13	6	4	1	11	14	0	12	10	2	9	5	3	15	7	8
60	30	22608	12	8	15	9	6	13	3	14	5	1	10	2	11	7	0	4
60	40	25159	5	0	4	13	15	10	12	1	6	14	9	11	3	7	8	2
60	50	25871	3	7	0	14	5	8	13	11	9	15	4	12	1	2	10	6
60	60	24169	10	13	11	3	7	8	1	15	9	4	12	6	5	0	14	2
60	70	24024	6	8	3	7	15	0	10	2	1	13	5	9	4	14	11	12
60	80	23940	3	10	1	4	8	0	5	12	7	13	2	9	15	6	11	14
60	90	25512	9	4	10	11	5	12	14	7	6	0	13	3	1	15	2	8
60	100	23945	13	1	8	3	11	12	9	14	15	7	5	10	2	6	0	4

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
70	10	26745	14	3	10	2	8	15	0	5	13	7	11	1	4	12	6	9
70	20	26301	8	7	12	0	13	2	6	10	1	5	3	14	11	4	15	9
70	30	20677	0	15	10	12	8	9	13	5	7	2	6	1	3	11	14	4
70	40	25081	10	8	15	2	9	11	3	5	14	0	4	13	6	12	1	7
70	50	21213	3	14	9	15	7	2	13	11	4	0	10	5	12	8	1	6
70	60	22998	15	10	12	4	9	3	8	7	5	11	1	6	14	2	13	0
70	70	23981	2	15	3	11	1	4	10	6	12	0	14	8	5	9	7	13
70	80	24612	14	7	1	8	2	13	4	0	6	9	5	3	12	15	11	10
70	90	23475	9	4	6	13	5	7	15	10	8	0	11	14	1	12	2	3
70	100	27876	4	7	10	2	12	11	1	9	0	5	15	13	6	3	8	14

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
80	10	25831	15	6	1	8	9	3	12	10	13	2	4	14	0	5	11	7
80	20	23207	15	7	9	12	5	8	0	4	6	1	13	11	3	14	2	10
80	30	23847	13	14	3	7	4	12	6	2	0	8	15	5	11	10	1	9
80	40	21781	15	3	5	2	7	11	0	12	8	4	13	9	1	10	6	14
80	50	23484	0	13	5	12	4	2	10	14	1	6	15	7	9	11	3	8
80	60	24572	0	3	11	6	4	5	15	1	8	12	7	14	13	9	10	2
80	70	22832	8	1	4	11	0	3	14	7	2	5	9	15	6	12	10	13
80	80	27619	11	1	2	0	13	8	4	15	10	5	6	12	9	3	14	7
80	90	23045	2	9	7	13	3	11	0	6	14	10	12	1	5	15	4	8
80	100	26734	10	15	12	6	7	4	11	5	13	8	3	2	1	14	0	9

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
90	10	27507	8	11	15	7	2	14	6	0	5	9	13	3	12	10	1	4
90	20	24323	9	14	2	11	12	7	6	3	4	10	0	5	13	1	15	8
90	30	23647	0	12	8	15	4	2	14	7	1	11	10	13	9	5	3	6
90	40	23495	1	4	8	3	11	10	13	2	7	12	5	14	15	6	9	0
90	50	21428	3	5	9	7	6	2	8	13	1	14	10	4	15	11	12	0
90	60	22867	9	5	14	11	4	7	10	15	6	3	12	0	1	13	2	8
90	70	23983	4	3	8	14	7	15	5	13	1	12	6	2	9	0	11	10
90	80	26477	7	8	13	0	12	6	2	11	14	5	3	4	10	9	15	1
90	90	24732	14	1	4	8	5	15	11	0	13	9	3	10	12	6	7	2
90	100	24561	12	5	3	7	2	11	8	15	0	1	4	6	9	14	10	13

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
100	10	29875	10	5	0	1	11	9	3	14	8	4	12	2	7	15	13	6
100	20	24210	6	0	13	8	2	14	10	5	9	7	12	3	11	15	1	4
100	30	23636	10	13	8	4	12	0	14	6	5	1	11	2	3	15	7	9
100	40	21477	3	15	11	8	7	0	14	9	2	6	5	12	4	1	13	10
100	50	26131	11	9	3	15	6	14	0	5	8	1	10	13	2	7	12	4
100	60	25278	7	0	13	8	3	15	5	9	12	1	11	10	4	6	2	14
100	70	23863	2	10	9	15	7	13	4	14	5	8	3	6	12	11	1	0
100	80	23732	9	5	10	7	12	15	8	4	13	3	2	0	14	11	1	6
100	90	26172	11	4	15	3	7	13	1	14	10	5	0	8	2	6	12	9
100	100	25759	5	9	1	7	6	0	4	11	13	12	3	15	10	8	2	14

Topologia toróide otimizando o consumo de energia.

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
10	10	27536	12	13	0	6	9	5	11	1	7	14	3	8	15	10	4	2
10	20	34097	15	3	0	10	6	13	9	12	11	4	14	8	5	1	7	2
10	30	26476	7	3	9	12	11	5	0	4	6	14	10	13	15	1	2	8
10	40	32009	5	12	4	14	15	0	7	9	1	6	10	8	3	13	2	11
10	50	29515	1	6	15	11	0	9	3	13	8	14	4	12	4	10	2	7
10	60	25903	5	7	14	4	13	8	3	15	8	11	10	0	6	2	12	1
10	70	31319	7	12	3	11	9	4	10	0	2	13	6	15	5	1	14	8
10	80	26090	9	7	11	2	14	10	13	3	6	15	5	4	8	1	12	0
10	90	28399	13	7	3	14	1	5	10	9	6	15	0	12	11	4	8	2
10	100	24048	2	7	8	13	11	3	14	6	1	12	5	9	4	0	10	15

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
20	10	33261	15	6	2	3	8	12	13	9	10	0	11	14	7	5	1	4
20	20	30420	15	7	12	11	3	10	2	8	0	13	6	5	9	4	14	1
20	30	27515	4	2	5	3	14	11	0	10	8	13	7	1	12	9	15	6
20	40	22115	6	0	15	11	3	8	9	2	4	1	14	7	5	13	10	12
20	50	26809	11	2	6	13	4	14	10	3	0	5	12	9	8	1	7	15
20	60	27535	13	2	10	5	15	6	1	14	3	12	0	8	9	7	11	4
20	70	26524	1	5	10	13	14	11	3	7	2	12	9	15	4	0	8	6
20	80	25573	7	6	1	9	15	11	13	4	5	0	10	3	12	8	2	14
20	90	23143	6	10	2	15	11	7	14	3	5	13	8	9	1	4	0	12
20	100	26583	11	6	15	3	0	14	5	2	13	7	9	4	10	1	12	8

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
30	10	29838	8	10	0	9	1	5	11	3	4	13	6	15	12	7	14	2
30	20	29777	14	8	2	7	1	10	6	3	13	0	11	15	9	5	12	4
30	30	25288	13	6	14	5	7	15	0	8	12	11	10	4	1	9	3	2
30	40	25974	8	6	1	9	12	0	11	2	5	4	3	13	15	10	14	7
30	50	29070	2	1	5	15	11	14	0	10	3	9	13	8	12	6	4	7
30	60	26194	8	0	9	1	10	13	4	12	3	7	11	2	5	14	6	15
30	70	28963	12	15	3	11	2	6	8	4	7	0	13	10	14	9	1	5
30	80	24903	0	1	7	15	4	9	3	8	11	6	14	5	10	12	2	13
30	90	27809	6	14	1	7	13	8	15	2	10	0	5	11	3	9	12	4
30	100	25614	11	14	8	4	3	1	15	0	5	9	7	10	2	12	13	6

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
40	10	27014	1	0	13	11	14	12	2	15	9	4	10	5	3	6	7	8
40	20	26911	9	15	10	1	13	5	0	4	7	8	12	6	14	11	2	3
40	30	26948	2	7	1	13	5	14	10	15	6	0	12	3	11	8	4	9
40	40	29753	14	9	6	13	0	7	2	8	3	15	11	12	5	1	10	4
40	50	25767	12	8	3	10	6	4	7	0	14	5	2	13	1	15	9	11
40	60	26552	6	3	11	12	14	5	9	2	1	13	0	10	7	4	8	15
40	70	25095	0	6	1	4	14	2	5	11	10	8	15	3	9	12	7	13
40	80	24081	15	6	13	11	9	14	0	5	12	10	8	3	1	4	2	7
40	90	25837	10	4	13	6	3	7	0	11	5	2	12	1	8	9	14	15
40	100	22184	0	4	2	10	12	1	5	14	6	8	13	9	11	7	3	15

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
50	10	30103	5	11	6	12	9	8	2	4	13	3	7	14	1	0	15	10
50	20	25566	14	3	6	1	4	15	2	9	12	5	11	7	0	13	10	8
50	30	26802	11	7	15	1	12	8	0	2	9	13	6	3	14	5	10	4
50	40	25626	12	0	14	10	2	13	11	15	8	1	5	3	4	6	9	7
50	50	23662	7	3	14	0	1	6	2	8	13	11	12	4	9	5	10	15
50	60	26884	5	1	10	3	12	14	7	0	15	6	13	4	11	9	2	8
50	70	23716	13	5	14	7	1	9	6	11	4	3	10	0	2	15	8	12
50	80	23253	9	0	8	6	5	10	4	1	12	14	2	13	3	7	15	11
50	90	23843	7	2	5	14	0	6	13	10	11	15	9	3	4	8	12	1
50	100	25584	8	0	15	6	14	4	3	10	11	7	5	1	12	13	2	9

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
60	10	29396	3	9	0	11	5	13	4	10	8	7	12	1	6	14	2	15
60	20	28450	5	11	15	6	8	0	2	13	14	7	12	9	1	10	4	3
60	30	22389	8	12	1	9	13	2	6	3	10	15	0	5	14	11	4	7
60	40	25757	2	15	6	0	14	4	13	5	3	12	1	8	11	10	9	7
60	50	21282	8	4	12	11	9	13	0	5	10	1	6	2	14	7	3	5
60	60	22443	10	14	5	8	15	2	6	1	3	13	0	4	11	9	12	7
60	70	24068	3	10	5	6	7	4	8	2	11	13	0	14	15	1	9	12
60	80	24653	2	9	10	12	13	6	14	3	0	8	5	11	7	1	4	15
60	90	25534	1	4	3	8	10	11	13	2	12	7	5	14	15	6	0	9
60	100	23417	14	9	1	11	5	4	8	6	3	15	2	10	7	12	0	13

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
70	10	32505	5	10	15	0	3	8	2	12	1	6	14	9	4	11	7	13
70	20	29118	6	7	4	11	14	0	5	2	13	10	3	8	1	15	9	12
70	30	33150	4	14	11	0	15	13	1	12	5	3	10	9	7	6	8	2
70	40	28778	8	3	5	11	9	10	12	15	14	1	6	2	0	4	7	13
70	50	25784	3	5	2	4	7	12	10	1	15	11	14	6	8	9	0	13
70	60	26236	5	11	3	4	14	9	15	8	0	13	2	1	12	7	10	6
70	70	27947	13	4	0	1	12	9	8	11	6	14	5	15	10	7	3	2
70	80	26671	12	10	3	8	4	6	7	0	14	2	5	13	1	15	9	11
70	90	28977	10	2	5	13	7	9	3	11	15	8	14	12	4	6	1	0
70	100	25843	13	14	9	8	1	0	5	11	15	12	4	6	10	3	2	7

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
80	10	22249	6	9	2	7	4	1	15	11	0	10	3	14	12	5	13	8
80	20	25255	11	3	5	6	7	1	4	0	8	12	2	15	14	10	13	9
80	30	24706	12	4	8	6	15	14	3	10	13	5	11	1	7	9	2	0
80	40	29488	9	6	1	3	0	8	14	2	11	12	13	7	5	15	4	10
80	50	23689	3	4	8	0	11	5	13	12	7	10	1	9	15	6	14	2
80	60	24273	12	2	10	0	8	14	3	15	13	4	11	7	5	9	1	6
80	70	22562	14	13	1	8	10	6	4	9	11	0	5	3	12	15	2	7
80	80	26809	10	13	11	3	7	8	1	15	9	4	12	6	5	0	14	2
80	90	24761	11	13	9	12	10	15	14	8	3	1	5	0	6	2	7	4
80	100	23349	15	5	1	4	11	12	9	0	10	6	8	14	2	3	7	13

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
90	10	29963	14	13	6	15	12	7	1	9	5	2	10	11	8	4	0	3
90	20	28115	0	4	7	13	9	3	2	11	6	14	12	15	8	5	1	10
90	30	23332	7	4	12	9	6	10	0	8	15	3	5	1	11	13	2	14
90	40	24109	15	2	7	3	10	13	9	4	12	11	6	0	1	5	14	8
90	50	25017	11	2	8	5	1	10	0	9	14	3	6	4	13	12	15	7
90	60	24388	0	13	5	12	4	2	10	14	1	6	15	7	9	11	8	3
90	70	27391	11	4	8	6	7	9	1	12	10	5	0	15	3	2	14	13
90	80	22674	4	9	10	15	0	11	1	14	5	12	13	2	8	3	7	6
90	90	24035	12	5	2	11	8	15	4	9	6	3	13	1	7	0	14	10
90	100	26412	13	8	4	3	11	12	0	5	15	7	14	10	2	6	9	1

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
100	10	27855	11	5	13	1	7	2	12	0	6	14	10	4	8	3	15	9
100	20	26350	6	2	9	8	10	15	1	12	0	7	14	11	13	5	4	3
100	30	25823	5	4	11	15	9	12	14	10	6	1	8	13	3	0	2	7
100	40	21548	10	6	13	2	4	11	7	3	8	14	1	5	15	9	12	0
100	50	19426	12	10	15	11	8	13	1	4	5	2	6	0	14	7	3	9
100	60	21032	13	14	6	9	1	8	3	10	5	2	7	0	12	4	11	15
100	70	25408	10	14	4	8	12	6	0	1	11	9	7	5	13	15	2	3
100	80	22346	7	4	3	0	2	8	5	9	1	11	14	15	6	12	10	13
100	90	21255	9	5	1	13	2	15	8	4	10	3	12	0	14	11	6	7
100	100	19590	12	0	1	2	4	8	15	3	9	6	13	11	7	5	14	10

Topologia malha otimizando a latência.

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
10	10	2163	2	15	4	8	10	14	0	13	7	3	12	9	6	5	1	11
10	20	2119	6	8	9	12	13	15	0	4	11	14	2	3	7	10	5	1
10	30	2053	6	3	14	2	5	12	13	7	9	4	15	10	1	0	11	8
10	40	2034	9	10	14	15	12	13	8	7	5	1	0	3	6	11	4	2
10	50	2032	7	6	14	10	5	2	11	9	4	1	3	15	13	12	8	0
10	60	2075	0	4	1	8	5	3	2	9	6	7	12	14	10	11	15	13
10	70	1967	15	0	5	7	14	3	11	6	13	9	8	10	12	4	1	2
10	80	2026	9	1	15	14	8	13	7	11	10	12	4	5	6	2	3	0
10	90	1970	8	12	13	9	11	5	4	0	14	10	2	1	3	15	6	7
10	100	2063	14	8	13	3	6	2	1	12	5	7	11	15	9	4	0	10

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
20	10	2120	0	9	5	6	3	13	1	7	11	4	8	2	15	14	12	10
20	20	2015	8	12	14	15	9	13	11	2	3	10	7	6	0	1	5	4
20	30	2052	7	15	13	8	11	12	9	5	10	6	1	3	14	2	4	0
20	40	2038	7	1	4	8	2	5	9	12	3	15	10	6	11	14	13	0
20	50	2027	11	8	13	9	7	3	1	4	6	10	12	5	2	14	15	0
20	60	2047	11	15	13	12	3	14	8	1	9	10	5	7	0	4	6	2
20	70	2067	9	4	15	11	14	1	7	10	13	3	2	5	12	6	8	0
20	80	2036	6	5	4	8	11	3	12	13	10	9	1	14	7	0	2	15
20	90	2110	3	7	11	4	2	8	14	1	10	9	15	13	12	0	5	6
20	100	1955	2	6	10	14	3	4	0	15	11	7	8	12	1	5	9	13

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
30	10	2110	7	2	11	8	13	6	10	9	0	1	15	14	12	4	5	3
30	20	2046	15	10	14	1	7	9	8	5	11	4	0	3	13	12	6	2
30	30	1993	7	5	11	15	8	0	4	3	9	13	1	2	14	12	10	6
30	40	**1988**	12	13	2	0	4	8	6	1	9	14	7	5	11	10	15	3
30	50	**1988**	6	2	1	0	14	10	9	4	11	8	13	5	15	12	7	3
30	60	1977	5	9	10	7	4	6	3	2	11	12	13	1	15	14	8	0
30	70	2009	12	10	13	8	9	11	14	15	3	5	4	6	7	1	0	2
30	80	1943	11	15	13	2	3	7	14	6	5	4	8	1	10	9	12	0
30	90	2028	10	3	11	1	9	7	4	2	8	0	5	6	12	13	14	15
30	100	1945	1	2	4	0	5	6	3	12	13	14	7	11	9	8	10	15

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
40	10	2034	10	14	12	0	13	15	8	1	11	6	9	5	2	3	7	4
40	20	2049	3	7	0	4	10	15	1	2	11	14	9	5	6	13	8	12
40	30	2048	13	8	9	14	0	1	15	11	5	4	12	10	7	3	2	6
40	40	1990	0	4	8	6	12	15	11	2	13	14	7	3	9	10	1	5
40	50	1980	5	8	10	11	1	0	4	15	6	2	14	13	7	3	9	12
40	60	1974	2	0	12	8	6	14	13	1	3	15	10	5	7	11	9	4
40	70	1921	13	8	1	0	12	9	5	3	10	4	6	2	15	14	11	7
40	80	2032	10	12	4	1	11	7	5	8	15	2	9	13	3	6	14	0
40	90	1962	2	10	11	15	8	0	14	7	12	9	5	3	13	6	4	1
40	100	2005	12	15	14	9	8	10	13	5	7	4	1	0	11	3	6	2

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
50	10	2051	11	15	3	6	10	13	2	7	14	1	0	4	12	8	9	5
50	20	2016	8	12	5	3	4	1	2	6	0	7	10	14	9	13	11	15
50	30	2031	12	0	2	1	8	4	15	3	6	9	13	11	7	5	14	10
50	40	2001	2	6	1	12	3	0	9	8	4	7	10	13	5	15	14	11
50	50	1972	14	13	8	12	11	10	15	9	7	2	3	1	5	6	0	4
50	60	1942	10	2	6	5	3	1	4	0	7	15	8	12	14	11	9	13
50	70	1907	6	10	15	14	3	7	11	12	2	5	8	4	0	1	9	3
50	80	2026	11	13	3	0	8	12	2	1	14	5	6	9	10	15	7	4
50	90	1932	15	11	7	3	10	9	2	6	14	5	8	1	0	4	13	12
50	100	1934	3	2	4	12	7	6	1	0	11	13	10	9	15	14	8	5

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
60	10	2146	3	10	8	6	2	9	12	4	1	11	14	13	0	7	15	5
60	20	1942	5	1	2	3	4	0	6	7	13	8	9	11	12	14	10	15
60	30	2085	5	3	9	15	1	8	12	14	0	7	3	2	6	11	4	10
60	40	1978	13	4	1	0	12	8	9	11	6	5	14	15	10	2	3	7
60	50	1992	6	8	4	0	9	12	3	1	5	10	7	2	11	15	14	13
60	60	1916	10	6	4	0	14	2	3	1	15	8	7	5	13	12	11	9
60	70	2032	3	1	7	15	4	0	6	2	10	11	9	14	5	12	8	13
60	80	1955	8	12	4	0	9	15	1	3	14	13	6	7	10	5	2	11
60	90	2034	12	5	2	11	8	4	15	9	6	3	13	1	7	0	14	10
60	100	2045	13	9	8	11	0	1	15	14	5	10	12	4	7	6	2	3

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
70	10	2070	6	1	11	15	8	4	7	3	10	12	0	2	9	13	5	14
70	20	2035	4	9	15	10	0	1	11	13	5	6	14	8	7	2	3	12
70	30	2058	3	14	11	2	0	15	9	6	1	5	8	10	13	12	4	7
70	40	1995	0	1	5	6	4	2	13	14	7	3	11	15	10	9	12	8
70	50	2040	9	3	6	10	5	7	2	14	15	11	0	8	13	4	1	12
70	60	1977	15	14	12	0	3	5	4	1	6	7	8	2	10	13	9	11
70	70	2011	2	1	6	12	3	8	9	0	4	7	11	13	14	15	5	10
70	80	1987	12	0	15	1	8	4	2	13	6	9	3	11	14	5	7	10
70	90	2056	8	12	5	7	4	1	6	2	0	13	10	14	9	3	15	11
70	100	1989	3	0	7	4	15	9	10	2	6	11	1	12	5	14	13	8

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
80	10	2081	11	10	13	12	2	6	4	9	1	7	3	14	0	5	8	15
80	20	2018	4	9	6	2	8	1	0	3	13	12	14	11	15	10	5	7
80	30	2052	6	2	5	0	10	15	8	4	14	9	13	12	11	7	1	3
80	40	1989	12	8	4	0	9	7	1	5	13	14	15	10	3	2	6	11
80	50	1920	11	13	9	12	10	15	14	8	3	1	5	0	6	2	7	4
80	60	2021	15	3	11	6	13	10	2	7	1	14	0	9	12	5	4	8
80	70	2087	13	14	9	8	0	1	15	11	5	12	4	10	6	2	3	7
80	80	1993	4	9	15	10	0	11	1	14	5	12	13	2	3	8	7	6
80	90	1965	5	1	15	11	4	0	7	10	13	8	12	3	9	14	6	2
80	100	2034	7	14	6	10	3	2	11	9	5	1	4	0	13	12	8	15

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
90	10	1954	3	6	5	8	10	7	9	4	2	15	13	1	14	11	12	0
90	20	2043	14	7	10	9	13	8	15	2	5	4	1	6	11	12	0	3
90	30	1959	10	14	8	4	12	6	0	1	9	11	7	5	13	15	3	2
90	40	1988	4	0	5	3	1	6	13	2	10	8	12	11	7	14	9	15
90	50	2039	9	4	1	11	14	12	7	8	13	3	5	2	15	6	10	0
90	60	2075	3	9	2	6	2	10	12	11	1	4	14	0	13	7	15	5
90	70	1960	5	8	9	13	1	3	4	14	12	7	15	2	10	11	0	6
90	80	1992	3	2	11	13	0	15	1	6	9	5	7	10	14	12	4	8
90	90	2020	10	12	8	14	9	11	13	15	6	5	0	3	7	4	2	1
90	100	2033	8	14	6	0	13	15	10	1	11	9	12	5	4	3	7	1

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
100	10	2028	9	14	13	0	12	10	15	4	5	6	3	1	8	11	7	2
100	20	1987	0	1	8	12	13	11	9	4	14	10	3	5	15	2	6	7
100	30	2008	5	0	8	15	13	12	11	14	1	2	6	10	7	3	4	9
100	40	1886	1	0	8	12	6	4	5	9	14	7	13	10	2	3	15	11
100	50	1938	6	9	11	10	7	3	8	4	15	12	13	2	1	0	5	14
100	60	1915	0	4	1	5	11	2	3	14	6	7	12	9	10	8	15	13
100	70	2074	6	15	10	14	3	7	12	2	3	5	9	8	0	1	4	11
100	80	2058	8	11	0	12	6	15	4	2	7	14	13	3	9	5	1	10
100	90	1981	14	2	8	9	10	11	15	0	1	13	3	7	6	5	12	4
100	100	1999	3	7	4	15	6	0	1	2	9	11	10	12	5	14	13	8

Topologia toróide otimizando a latência.

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
10	10	2141	0	1	5	8	3	15	12	4	7	9	13	14	11	10	2	6
10	20	2089	9	13	4	5	10	14	15	12	11	6	7	8	0	1	2	3
10	30	2030	3	2	1	9	7	5	4	0	13	8	6	10	12	15	11	14

10	40	2116	12	13	14	11	8	1	0	9	3	7	2	5	4	6	10	15
10	50	2057	7	3	5	1	4	11	9	2	6	12	13	15	0	8	14	10
10	60	2044	1	0	7	3	5	2	11	14	4	6	8	15	9	10	13	12
10	70	2048	1	3	8	9	7	0	4	13	5	15	12	10	2	6	14	11
10	80	2021	5	8	3	1	4	11	7	0	13	9	10	6	12	15	14	2
10	90	2002	12	8	2	6	14	13	0	4	15	10	11	5	3	7	1	9
10	100	1990	2	3	1	0	4	8	5	10	7	9	13	15	6	11	12	14

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**														
20	10	2121	12	13	10	9	11	15	14	7	3	4	6	0	2	5	1	8
20	20	2103	3	2	14	6	0	4	5	15	12	13	1	7	11	10	9	8
20	30	2051	3	7	2	15	5	1	9	14	6	0	8	10	4	13	12	11
20	40	2092	14	11	10	7	15	5	1	3	12	0	4	8	9	13	6	2
20	50	1952	1	2	6	7	3	0	4	11	13	5	9	15	8	12	10	14
20	60	2065	12	2	10	15	9	6	7	13	14	11	3	4	8	5	1	0
20	70	1966	9	11	6	8	12	10	1	0	14	13	5	4	7	15	3	2
20	80	2174	14	1	6	2	10	8	0	7	13	11	4	15	3	9	12	5
20	90	1989	8	4	0	6	11	5	1	15	12	13	3	7	9	14	10	2
20	100	2003	7	14	3	0	1	5	2	4	6	8	9	10	15	12	13	11

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**														
30	10	2122	6	2	8	5	14	15	12	9	11	7	1	0	13	3	10	4
30	20	2056	15	10	5	9	14	11	6	1	4	8	3	2	0	12	13	7
30	30	2044	0	10	12	4	2	3	13	8	5	6	9	11	1	7	15	14
30	40	1951	0	9	10	8	4	5	3	1	13	12	6	2	14	15	11	7
30	50	2023	1	5	3	2	6	4	7	15	14	10	8	0	9	11	13	12
30	60	1991	8	13	12	9	15	14	10	11	4	5	7	3	0	1	2	6
30	70	2060	8	7	0	1	13	2	4	5	10	3	12	14	6	11	9	15
30	80	2043	10	2	5	13	7	3	9	11	15	8	12	14	6	4	1	0
30	90	2076	7	1	3	5	11	4	0	12	10	14	2	6	15	13	9	8
30	100	1971	0	11	15	14	12	9	10	3	8	13	1	2	4	5	6	7

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**														
40	10	1997	3	6	0	1	2	5	4	13	15	9	8	12	10	14	7	11
40	20	2055	11	14	6	5	7	10	15	12	3	4	1	9	2	0	13	8
40	30	2058	1	12	4	3	5	0	2	7	10	6	13	15	14	8	9	11
40	40	2062	7	6	3	4	15	5	2	9	8	12	13	14	0	1	11	10
40	50	2052	12	13	1	3	8	10	5	0	4	14	11	15	2	6	7	9
40	60	2040	5	1	9	12	3	2	0	10	11	15	8	13	7	14	6	4
40	70	1962	7	6	5	13	3	11	2	9	1	0	4	8	10	12	14	15
40	80	2043	8	5	7	2	9	1	4	6	15	13	10	3	0	12	11	14
40	90	1970	3	7	6	15	14	11	8	5	13	10	2	1	12	9	0	4

40	100	1966	1	0	2	7	10	4	5	3	6	8	13	9	15	12	14	11
Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**															
50	10	2044	8	9	10	15	12	5	11	3	13	0	6	2	14	7	1	4
50	20	2020	7	3	4	0	2	5	8	9	1	11	15	14	6	10	12	13
50	30	2072	10	6	7	0	2	15	12	5	3	11	8	1	4	9	13	14
50	40	1985	2	3	8	12	6	5	7	5	14	13	10	0	11	9	4	1
50	50	2042	15	5	1	4	11	12	9	0	10	6	8	14	2	3	7	13
50	60	2002	1	7	2	10	13	5	6	14	12	9	8	11	4	0	3	15
50	70	1935	13	14	10	11	15	12	9	3	2	8	5	1	6	7	4	0
50	80	1992	10	3	7	1	15	14	11	0	6	2	12	9	4	5	13	8
50	90	1968	12	14	13	5	9	15	10	4	8	0	3	7	1	2	6	11
50	100	2018	8	0	1	2	11	9	6	3	7	13	14	5	10	12	15	14

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**															
60	10	2082	0	3	7	4	12	10	2	1	11	14	15	6	13	8	9	5
60	20	2119	11	6	4	14	15	13	9	7	3	1	1	12	10	8	2	5
60	30	2064	3	11	14	1	6	2	7	0	10	4	9	5	13	12	8	15
60	40	2014	14	1	5	4	2	0	8	6	9	13	12	7	10	11	15	3
60	50	1934	4	0	1	5	15	11	3	7	13	9	2	6	12	8	10	14
60	60	1980	3	0	2	6	1	5	7	10	13	12	11	14	4	8	9	15
60	70	1906	0	1	13	4	3	6	5	8	2	7	11	12	9	10	15	14
60	80	2026	15	13	8	0	11	2	3	4	12	10	7	6	9	14	5	1
60	90	2108	8	12	14	15	9	13	11	2	3	10	7	6	0	1	5	4
60	100	2059	10	6	4	0	14	2	3	1	15	8	7	5	13	12	11	9

Partículas	**Iterações**	**Resultados**	**Disposição dos nodos aos roteadores**															
70	10	1952	13	14	15	0	8	10	11	6	12	9	7	3	4	1	5	2
70	20	1999	3	7	11	15	2	10	14	13	1	4	6	5	12	8	0	9
70	30	1945	3	6	11	7	2	5	15	10	1	9	14	13	4	0	12	8
70	40	2070	2	8	9	14	0	12	10	6	4	13	1	7	15	11	3	5
70	50	1981	2	6	10	15	0	7	14	13	4	3	5	11	8	12	9	1
70	60	1925	1	0	2	7	5	6	11	3	4	13	14	10	12	9	8	15
70	70	2061	0	8	14	2	4	12	6	5	10	9	3	7	11	13	15	1
70	80	2134	2	1	6	12	3	8	9	0	4	7	11	13	14	15	5	10
70	90	1988	1	4	8	0	2	13	3	14	6	10	15	7	9	12	5	11
70	100	1969	11	4	15	3	7	13	1	14	10	5	0	8	2	6	12	9

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
80	10	2065	5	6	9	8	2	1	13	11	0	4	12	10	3	7	15	14
80	20	2001	10	14	10	7	9	13	4	2	11	12	8	1	3	15	0	5
80	30	2054	4	6	10	7	12	15	0	3	14	8	1	2	11	13	9	5
80	40	2016	14	9	1	5	11	8	4	6	3	2	15	10	7	0	12	13
80	50	1942	3	6	1	4	2	11	0	5	15	14	8	10	13	12	9	7
80	60	2048	6	5	11	3	2	7	14	13	10	8	15	12	1	9	0	4
80	70	1987	4	8	2	7	13	1	11	3	10	14	5	6	15	9	12	0
80	80	1993	10	13	11	3	7	8	1	15	9	4	12	6	5	0	14	2
80	90	2008	9	4	11	13	1	6	14	12	8	5	2	15	0	10	3	7
80	100	2032	15	3	5	2	7	11	0	12	8	4	13	9	1	10	6	14

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
90	10	2076	11	3	15	9	7	6	2	1	4	0	14	13	5	8	10	12
90	20	1998	2	0	8	14	4	9	12	15	1	5	7	3	13	10	6	11
90	30	1981	7	3	11	9	8	15	10	0	12	14	6	4	5	13	1	2
90	40	1935	1	5	0	12	7	4	8	13	6	2	9	10	3	15	11	14
90	50	1865	0	8	9	13	1	4	12	14	6	7	15	11	2	3	10	5
90	60	1997	4	9	15	10	0	11	1	14	5	12	13	2	3	8	7	6
90	70	1967	3	14	11	2	0	15	9	6	1	5	8	10	13	12	4	7
90	80	2021	7	5	11	15	8	0	4	3	9	13	1	2	14	12	10	6
90	90	2085	7	6	14	10	5	2	11	9	4	1	3	15	13	12	8	0
90	100	1992	12	8	4	0	9	7	1	5	13	14	15	10	3	2	6	11

Partículas	Iterações	Resultados	Disposição dos nodos aos roteadores															
100	10	2076	11	3	15	9	7	6	2	1	4	0	14	13	5	8	10	12
100	20	1998	2	0	8	14	4	9	12	15	1	5	7	3	13	10	6	11
100	30	1981	7	3	11	9	8	15	10	0	12	14	6	4	5	13	1	2
100	40	1935	1	5	0	12	7	4	8	13	6	2	9	10	3	15	11	14
100	50	1865	0	8	9	13	1	4	12	14	6	7	15	11	2	3	10	5
100	60	1997	4	9	15	10	0	11	1	14	5	12	13	2	3	8	7	6
100	70	1967	3	14	11	2	0	15	9	6	1	5	8	10	13	12	4	7
100	80	2021	7	5	11	15	8	0	4	3	9	13	1	2	14	12	10	6
100	90	2085	7	6	14	10	5	2	11	9	4	1	3	15	13	12	8	0
100	100	1992	12	8	4	0	9	7	1	5	13	14	15	10	3	2	6	11