RESOLVENDO O PROBLEMA DE PLANEJAMENTO DA EXPANSÃO DA GERAÇÃO DE ENERGIA COM ENXAMES DE PARTÍCULAS BINÁRIOS

Ramon Diacovo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Priscila Machado Vieira Lima, Ph.D.

Prof. Heloisa Teixeira Firmo, D.Sc.

Prof. Nelson Maculan Filho, D.Habil.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2008

DIACOVO, RAMON

Resolvendo o Problema de Planejamento da Expansão da Geração de Energia com Enxames de Partículas Binários [Rio de Janeiro] 2008

XI, 118 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2008)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Enxames de partículas
2. GEP
3. SATyrus
4. Otimização combinatória

I. COPPE/UFRJ II. Título (série)

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

RESOLVENDO O PROBLEMA DE PLANEJAMENTO DA EXPANSÃO DA GERAÇÃO DE ENERGIA COM ENXAMES DE PARTÍCULAS BINÁRIOS

Ramon Diacovo

Março/2008

Orientadores: Felipe Maia Galvão França
Priscila Machado Vieira Lima

Programa: Engenharia de Sistemas e Computação

A importância da área de planejamento energético, particularmente o planejamento da expansão da geração de energia motiva o desenvolvimento de estudos sobre este problema. As opções para o enfoque são várias: para uma dada instância é possível buscar melhores soluções ou modelá-la de maneira a contemplar grandes quantidades de fatores relevantes. Outra possibilidade é a busca por novas ferramentas para trabalhar com o problema. O presente trabalho apresenta uma modelagem de uma instância do problema do planejamento da expansão da geração de energia utilizando a plataforma SATyrus. Esta plataforma consiste em um sintetizador de funções exatas, baseado nos princípios de minimização de energia e redes e Hopfield de alta ordem. Para a síntese das funções, o SATyrus recebe como entrada a descrição de um problema escrito em uma linguagem declarativa própria. Esta modelagem é, então, utilizada para realizar análises com o método de otimização por enxames de partículas binários (*binary Particle Swarm Optimization*). Este método é baseado no comportamento da movimentação de enxames, e toma proveito da interação entre os componentes dos mesmos para resolver problemas de otimização combinatória cujas dimensões são booleanas.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SOLVING THE ENERGY GENERATION EXPANSION PLANNING PROBLEM WITH BINARY PARTICLE SWARMS

Ramon Diacovo

March/2008

Advisors: Felipe Maia Galvão França
Priscila Machado Vieira Lima

Department: Systems Engineering and Computer Science

The importance of energy planning and the generation expansion planning, in particular, motivates the development of studies regarding this problem. There are plenty of options avaliable for focusing: for a given instance, it is possible to search for better solutions or to model it so that this instance contemplates large amounts of relevant factors. Another possibility is to find new tools to work with the problem. This work presents a modeling of a generation expansion problem instance, through the SATyrus platform. This platform consists of an exact function synthesizer, and it is based on the principles of energy minimization and high-order Hopfield networks. For the function synthesis, SATyrus takes as input the description of a problem, written on SATyrus' own declarative language. This modeling is then used to conduct an analysis of the binary Particle Swarm Optimization method, a paradigm based on bug swarms' behavior. It functions by taking advantage of the interaction between the swarm components in order to solve combinatorial optimization problems which have only boolean dimensions.

Sumário

Lista de Figuras

Lista de Algoritmos

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

O planejamento energético é um setor de grande importância estratégica para o país, especialmente após o aumento da participação da iniciativa privada e da liberalização do setor. Esses fatores, embora permitam que as decisões sejam tomadas sob uma ótica não-centralizadora, aumentam a complexidade da modelagem do setor energético como um todo, afetando inclusive as abordagens matemáticas utilizadas.

Dentro deste campo, o planejamento da expansão da geração da energia (*generation expansion planning* ou GEP) é um dos problemas importantes, onde pesquisadores têm gasto recursos em busca de novos e eficientes métodos de resolução. Boas soluções para ele implicam em um melhor aproveitamento da verba utilizada na construção e operação de usinas produtoras de energia elétrica. Trata-se de um problema de otimização combinatória real — e difícil —, além de ser o primeiro alvo do projeto do Laboratório de Otimização Avançada (LOA) [4], que tem por objetivo explorar novas formulações para modelos de programação matemática orientados ao setor energético.

Por outro lado, outros problemas de difícil resolução motivaram a criação da plataforma SATyrus, que compila funções exatas, desenvolvida para auxiliar na modelagem e resolução de problemas de otimização combinatória. A plataforma emprega redes neurais sem peso para a tarefa da conversão em minimização de energia. O processo tem início a partir da modelagem do problema-alvo, escrita proposici-

onalmente na linguagem declarativa do SATyrus. A rede neural é então utilizada para gerar uma função de energia [13], a ser posteriormente minimizada pelo resolvedor. Observou-se que o SATyrus não contava com um resolvedor adequado, o que enquanto limita sua utilidade prática, motiva estudos em novos métodos de resolução.

1.2 Objetivos e contribuições

Este trabalho tem por objetivo contribuir para a diversificação das opções de ferramentas utilizáveis para a realização de estudos com o GEP. Apresentaremos uma primeira modelagem deste problema para a plataforma SATyrus. A partir dela, será mais simples modelar versões mais complexas do problema, que considerem outros fatores relevantes para o GEP.

Aproveitaremos a modelagem para realizar estudos com alguns resolvedores, mais especificamente com o *Particle Swarm Optimization* (PSO) binário. Verificaremos como este se comporta na tarefa de minimizar a função objetivo de GEP gerada pelo SATyrus, em ambientes de computação paralela. Ao mesmo tempo, realizaremos um extensivo conjunto de testes com nossa modelagem do GEP e o PSO binário, buscando agregar mais informações sobre este paradigma alternativo que, por ser menos difundido do que muitos outros, não conta com dados deste tipo suficientes.

Esperamos que a contribuição da nova modelagem do GEP seja útil para proporcionar alternativas para a resolução do problema. Como outra contribuição, o estudo realizado com os parâmetros do PSO binário proporcionará um maior nível de compreensão sobre seu comportamento.

1.3 Estrutura do texto

No Capítulo 2 apresentamos uma revisão bibliográfica sobre métodos de resolução de problemas de otimização combinatória relevantes para o presente trabalho. No Capítulo 3, o estudo de caso do trabalho, o GEP, é descrito, bem como algumas de suas possíveis formulações. Após a descrição do GEP temos, no Capítulo 4, uma

breve descrição da plataforma SATyrus, incluindo seu funcionamento e sua estrutura de arquivos de entrada.

A contribuição deste trabalho se inicia no Capítulo 5, onde é introduzida a modelagem do GEP para o SATyrus. O código-fonte do arquivo principal desta modelagem pode ser encontrado no Apêndice A. Seguindo, o Capítulo 6 contém a descrição, apresentação e análise dos experimentos realizados. Por questões de organização, os gráficos relativos a alguns dos experimentos não estão presentes no Capítulo 6, mas o leitor pode encontrá-los no Apêndice B. Finalmente, o Capítulo 7 contém as conclusões e comentários finais sobre o trabalho, assim como algumas propostas para trabalhos futuros.

Capítulo 2

Conhecimentos preliminares

Este capítulo apresenta alguns paradigmas e resolvedores para problemas de otimização combinatória. Todos foram estudados como possíveis abordagens para resolver o problema descrito no Capítulo 5, mas atenção especial foi dada ao resolvedor da Seção 2.4, pois há pouco estudo sobre seu emprego na situação apresentada.

2.1 Simulação Metropolis Monte Carlo

Esta seção apresenta conceitos que serão necessários para a compreensão do algoritmo da Seção 2.2, embora não descreva um método de resolução de problemas de otimização combinatória por si própria.

A simulação Metropolis Monte Carlo [23] utiliza movimentos aleatórios (perturbações) dentro do espaço de estados para explorá-lo, em busca do estado mais estável. A simulação é feita a uma temperatura T. A cada iteração, um novo estado candidato é apresentado. Caso este estado seja mais estável que o atual, ele é aceito. Caso contrário, é também aceito com uma probabilidade diretamente proporcional a T.

A probabilidade empregada para a aceitação de um novo estado garante que a média de qualquer propriedade do sistema, como a energia, possa ser calculada facilmente através da média de Boltzmann, para uma amostra suficientemente grande. A prova pode ser encontrada no artigo original [23].

É importante observar que o método utilizado para realizar as perturbações nos estados deve permitir que qualquer estado seja alcançado a partir de qualquer outro

Algoritmo 1 SimulaçãoMMC ($s_{inicial}$, T, $nMaxTentativas$)

$k \Leftarrow$ Constante de Boltzmann
$s_0 \Leftarrow s_{inicial}$
Para $i = 0$ **até** $nMaxTentativas$ **fazer**
  $e_0 \Leftarrow$ Energia calculada em s_0
  $s_1 \Leftarrow s_0 +$ perturbação
  $e_1 \Leftarrow$ Energia calculada em s_1
  Se $e_0 > e_1$ **então**
    $s_0 \Leftarrow s_1$
  Senão
    $r \Leftarrow$ número aleatório no intervalo [0,1]
    Se $e^{\frac{(e_1 - e_0)}{kT}} > r$ **então**
      $s_0 \Leftarrow s_1$

em um número de iterações menor ou igual a $nMaxTentativas$ (e, portanto, finito). Do contrário, alguns estados podem nunca ser alcançados, e a média de Boltzmann não vale.

No algoritmo acima, a média de Boltzmann com respeito a energia pode ser obtida se, ao final de cada iteração, for armazenada em um acumulador a energia do estado atual. Ao final da simulação, a média será o valor deste acumulador dividido pelo número de iterações.

2.2 Têmpera Simulada (*Simulated Annealing*)

Proposto em [20], este método de otimização utiliza uma simulação de Metropolis Monte Carlo para encontrar a configuração mais estável (com menor energia) de um sistema. A inspiração vem do processo de temperamento do vidro.

Inicialmente, o vidro é aquecido a uma temperatura muito elevada, de forma que se torne líquido e os átomos tenham maior mobilidade. A temperatura é reduzida gradualmente e os átomos do vidro buscam a configuração mais estável a cada temperatura. Se o resfriamento for conduzido de maneira lenta o suficiente, ao final do processo os átomos estarão na configuração mais estável.

Descrevemos agora o algoritmo da Têmpera Simulada. T_i e T_f são, respectivamente, as temperaturas inicial (alta) e final (baixa) do processo.

A taxa de resfriamento logarítmica utilizada não é a única possível. A distribuição de Cauchy também pode ser utilizada, mudando a fórmula da atualização da

Algoritmo 2 Têmpera Simulada (T_i, T_f, $nTentativasPasso$)

$nPassos \Leftarrow 0$
$s_0 \Leftarrow$ Estado aleatório
$T \Leftarrow T_i$
Enquanto $T > T_f$ **fazer**
  $s_0 \Leftarrow$ SimulacaoMMC(s_0, T, $nTentativasPasso$)
  $T \Leftarrow T_i/ln(nPassos)$
  $nPassos \Leftarrow nPassos + 1$

temperatura para a Equação 2.1.

$$T \Leftarrow \frac{T_i}{nPassos} \tag{2.1}$$

A determinação do parâmetro *nTentativasPasso* é uma das maiores dificuldades no uso da Têmpera Simulada. Embora a simulação Metropolis Monte Carlo tenha como objetivo reproduzir a distribuição de Boltzmann correta a uma dada temperatura, quando utilizada na Têmpera Simulada, ela só precisa rodar por tempo suficiente para explorar as regiões razoavelmente populadas do espaço de busca. Essa restrição de suficiência permite uma redução no número de iterações da simulação MMC. Entretanto, há uma relação entre este número e o tamanho máximo de cada passo (que depende do modo com que as perturbações nos estados são conduzidas) que pode ser difícil de ser otimizada, por ser muito dependente de características intrínsecas ao problema a ser resolvido.

2.3 Satisfabilidade

2.3.1 Definição do problema SAT

Uma fórmula da lógica proposicional é dita na forma normal conjuntiva (CNF) quando é uma conjunção ("e"s lógicos) de disjunções ("ou"s lógicos) de literais. Um literal é a ou $\neg a$, onde a é uma variável proposicional. A cada uma das disjunções damos o nome de cláusula.

Dada uma fórmula na CNF, resolver o problema SAT associado é encontrar uma valoração de suas variáveis booleanas tal que o valor da fórmula seja *verdadeiro*. Isto é equivalente a dizer que a valoração deve fazer com que cada cláusula da

fórmula possua pelo menos um literal com valor *verdadeiro*. Estas cláusulas são ditas satisfeitas; quando não existe uma valoração que satisfaça a todas as cláusulas, o problema é dito insatisfazível.

2.3.2 Definição do problema pseudo-booleano

Uma inequação pseudo-booleana é da forma $c_0x_0 + c_1x_1 + ... + c_{n-1}x_{n-1} \geq c_n$, onde para todo i, c_i é um coeficiente inteiro. x_i é da forma $p_j \times p_k \times ... \times p_l$, onde p_i é uma variável booleana ou sua negação. A inequação é dita linear se, para todo i, x_i for da forma p_i. Um literal *verdadeiro* é mapeado para o valor 1, ao passo que um literal $falso$ é interpretado como 0. A inequação pseudo-booleana é dita satisfeita sob uma determinada valoração se, e somente se, o primeiro membro for maior ou igual ao segundo. Uma função objetivo é uma expressão na mesma forma do primeiro membro de uma inequação pseudo-booleana. Dado um conjunto de inequações pseudo-booleanas Γ e um função objetivo f, o problema de otimização pseudo-booleana consiste em encontrar uma valoração para as variáveis de f que a minimize, enquanto satisfaz às inequações de Γ. O problema é dito linear se todas as suas inequações e função objetivo são lineares.

2.3.3 Minisat+

Para resolver problemas de otimização pseudo-booleanos, temos um conjunto de técnicas frequentes. Uma delas consiste no uso de um resolvedor SAT como método de decisão para um procedimento de mais alto nível com uma lógica mais rica. Outro baseia-se na extensão de resolvedores SAT para permitir que estes aceitem inequações e funções objetivo pseudo-booleanas.

O Minisat+ (introduzido em [11]) é um resolvedor de problemas pseudo-booleanos lineares que emprega uma técnica menos comum: transformar as inequações em cláusulas lógicas, e então aplicar um resolvedor SAT puro.

Antes da transformação, muitas normalizações são feitas para diminuir a diversidade das inequações e simplificar ao máximo o problema. Eis alguns exemplos:

- Inequações do tipo $\leq$ são transformadas em inequações do tipo $\geq$ através da negação de todas as suas constantes.

- Inequações satisfazíveis trivialmente, como "$p_1 + p_2 \geq 0$" são removidas.
- A presença de inequações trivialmente insatisfazíveis, como "$p_1 + p_2 \geq 9$" interrompe o pré-processamento e gera como resposta "insatisfazível".
- Em $c_0x_0 + ... + c_{n-1}x_{n-1} \geq c_n$, substitui-se todo $c_i \forall i < n$ por c_n, caso $c_i > c_n$.
- Valorações triviais são fixadas e propagadas para outras inequações. Por exemplo, a valoração $p_i = true$ é dita trivial se, ao assumir $p_i = false$, alguma inequação imediatamente se tornar insatisfazível.

Realizado o pré-processamento, as inequações são transformadas em circuitos de uma única saída através de um dos seguintes métodos: *binary decision diagram* (BDD), *network of adders* ou *network of sorters*. A preferência é para os BDDs, depois *sorters*, e por último *adders*, devido a um *tradeoff* entre as propriedades desejáveis do circuito resultante e a complexidade de tempo para gerá-lo.

Finalmente, é aplicada uma variação da transformação de Tseitin [29], que apresenta as cláusulas como saída. É importante notar que alguns dos passos dados até aqui podem introduzir novas variáveis ao problema, aumentando sua dimensão.

Dado um problema de otimização pseudo-booleano definido por inequações Γ e pela função objetivo $f(p)$, pode-se encontrar sua solução com chamadas iterativas ao resolvedor SAT. Primeiro, este é aplicado ao conjunto Γ' de cláusulas pré-processadas, e a valoração encontrada é utilizada para determinar $f(p) = k$. Adiciona-se $f(p) \leq k$ ao conjunto Γ e repete-se o processo até a obtenção de um conjunto insatisfazível. A sequência de passos está sintetizada no algoritmo 3.

Algoritmo 3 Otimizar Minisat+(Γ, $f(p)$)

$\Gamma' \Leftarrow$ preProcessar(Γ)
$v \Leftarrow$ resolverSAT(Γ')
$k \Leftarrow f(v)$
Enquanto $\Gamma \cup (f(p) \leq k)$ for satisfazível **fazer**
  $\Gamma' \Leftarrow$ preProcessar($\Gamma \cup (f(p) \leq k)$)
  $v \Leftarrow$ resolverSAT($\Gamma' \cup (f(p) \leq k)$)
  $k \Leftarrow f(v)$

2.3.4 Resolvedor bsolo

O resolvedor bsolo [22] utiliza uma variação da abordagem *branch and bound* [25] para solucionar problemas de otimização combinatória.

O processo utiliza uma árvore de busca, onde cada nó representa uma valoração para uma variável. Um caminho da raiz até uma folha qualquer determina os valores de tantas variáveis quantos forem os níveis da árvore. Caminhos não promissores são interrompidos (podados).

Os limites superiores para a função objetivo são identificados e os limites inferiores são estimados. Boas políticas para estimar os limites inferiores são de extrema importância para o bom funcionamento do algoritmo; as utilizadas pelo bsolo estão descritas em [10]. A árvore de busca é podada sempre que o valor para o limite superior se iguala ou se aproxima mais do valor ótimo do que a estimativa para o limite inferior. Eis os passos do *branch and bound* utilizado:

Algoritmo 4 Otimizar bsolo(Γ, $f(p)$)

```
lim_sup ⇐ ∞
Enquanto resposta não encontrada fazer
  Se v contém valorações para todas as variáveis então
    lim_sup ⇐ f(v)
    lançar um conflito[1] para garantir a poda
  Senão
    x ⇐ variável não-valorada de v
    v ⇐ v∪ valoração de x
    bifurcar a busca em x
  aplicar propagação booleana
  Se um conflito foi gerado então
    Γ ⇐ Γ∪ cláusulas relevantes
    Se backtracking é necessário então
      realizar backtracking
    Senão
      proceder com a busca
  lim_inf ⇐ nova estimativa
  Se lim_inf ≥ lim_sup então
    lançar um conflito
    realizar backtrack
```

A essa descrição simplificada, são acrescentadas otimizações provenientes de re-

[1] Conflitos ocorrem quando, durante a busca, um ramo cuja valoração correspondente torna alguma das restrições insatisfazível.

solvedores SAT, como *backtracking* não-cronológico e identificação de atribuições necessárias.

2.4 Particle Swarm Optimization (PSO)

O conceito de Particle Swarm Optimization (PSO) foi introduzido em [17], e corresponde à simulação do comportamento de um enxame em busca de comida. Ao invés de realizar a busca em um espaço real, com três dimensões, o espaço de busca do enxame é parametrizado por uma função objetivo, e terá tantas dimensões quantas forem as variáveis da função. Em cada dimensão, os intervalos válidos também são estabelecidos pela função objetivo, e não há nada no paradigma que os obrigue a serem iguais. Neste contexto, o melhor ponto para alimentação é aquele que minimiza o valor desta função, e a tarefa do enxame é encontrar esse ponto.

Os S elementos do enxame são chamados de partículas. Cada uma delas armazena apenas um pequeno conjunto de informações, a saber:

- sua velocidade vetorial $\vec{v}$;
- sua posição atual $\overrightarrow{pos}$;
- uma lista com as partículas que pertencem a sua vizinhança $\overrightarrow{neigh}$;
- o melhor ponto já alcançado por ela própria $\overrightarrow{pBest}$;
- o melhor ponto já alcançado por qualquer partícula na sua vizinhança $\overrightarrow{lBest}$.

No início do algoritmo, as partículas estão espalhadas aleatoriamente. Em todo passo, cada partícula se movimenta, seguindo as três seguintes tendências:

Inércia é a influência da velocidade atual da partícula sobre sua próxima posição. Faz com que ela continue seguindo na mesma direção no espaço de busca.

Individualismo é a influência do conhecimento local da partícula. Tende a trazer a partícula para o melhor ponto visitado por ela própria, por vezes contrariando a experiência conjunta do enxame.

Coletivismo é a influência do conhecimento do coletivo sobre a partícula. É uma tendência que move a partícula em direção ao ponto ótimo (até então) da vizinhança, em detrimento da exploração individualista.

Além da movimentação, as partículas também atualizam suas informações. Após um certo número de passos, espera-se que o enxame tenha convergido para um ponto, o *mínimo global* da função.

Em linhas gerais, este é o comportamento do algoritmo. Desde a sua criação, foram concebidas diversas variações, que têm por objetivo melhorar a performance do PSO em determinados casos. A seguir apresentamos o algoritmo original, e também algumas variantes interessantes, inclusive as utilizadas nos experimentos.

2.4.1 Tipos de vizinhança

Na Seção 2.4, este conceito foi mencionado superficialmente. Alterações na vizinhança podem mudar significativamente o comportamento do PSO. A topologia da vizinhança é facilmente visualizada se pensarmos nas partículas do PSO como nós em um grafo orientado, e nos relacionamentos de vizinhança como arestas nesse grafo.

A figura 2.1 mostra os dois tipos de vizinhança mais difundidos: a global, onde cada partícula é vizinha de todas as outras do enxame; e a em anel, onde cada partícula p_i é vizinha de outras duas p_{i-1} e p_{i+1}, $\forall i \in \mathbb{Z}_S$. Foi utilizada a notação de um grafo não-orientado porque, nestes exemplos, as vizinhanças são simétricas.

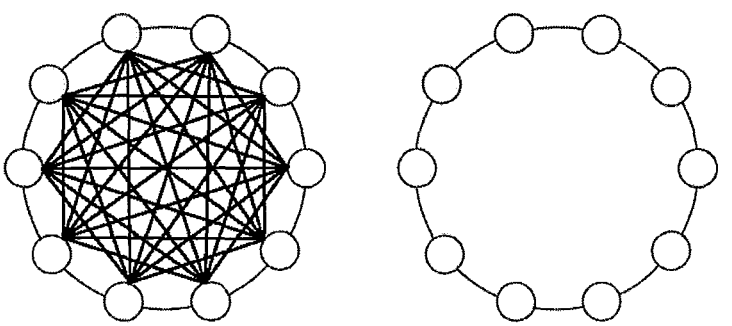

Figura 2.1: Vizinhanças comuns em PSO. À esquerda, global; à direita, anel.

Na vizinhança global, a informação sobre um novo ponto ótimo descoberto leva uma iteração para alcançar todas as partículas do enxame. Já na em anel, o número

de iterações necessárias para tal feito é $S/2$. Essa lentidão na propagação faz com que as partículas demorem mais para que sejam afetadas por um $\overrightarrow{lBest}$ recém-descoberto, permitindo a elas continuar a exploração com a informação atual por algum tempo. Isto tem a consequência teórica de fazer com que a convergência seja alcançada mais tardiamente, em troca de uma exploração mais completa do espaço de busca.

Existem inúmeras outras topologias para a definição das vizinhanças. Na verdade, não há restrição alguma para a formação destas, sendo possível até mesmo que variem durante a execução do algoritmo.

2.4.2 PSO clássico

Em [16], os criadores do PSO descrevem a primeira heurística para a movimentação das partículas do enxame, que serviu de base para todas as outras. Essa heurística consiste basicamente na implementação do comportamento descrito na Seção 2.4. O PSO clássico foi concebido para problemas cujas dimensões são números reais. Portanto, nesta seção, devemos considerar toda a informação numérica armazenada nas partículas ($\vec{v}$, $\overrightarrow{pos}$, $\overrightarrow{pBest}$ e $\overrightarrow{lBest}$) como valores em $\mathbb{R}^D$, onde D é o número de dimensões do problema.

A equação que rege a movimentação das partículas é simples, e dada pela 2.2.

$$\overrightarrow{pos} \Leftarrow \overrightarrow{pos} + \vec{v}. \tag{2.2}$$

Ou seja, a cada passo do algoritmo, as partículas movem-se seguindo suas velocidades atuais. A alma do PSO clássico está na atualização dessas velocidades, que é dada pela Equação 2.3.

$$\vec{v} \Leftarrow \vec{v} + \varphi_1 \times rand(0,1) \times (\overrightarrow{pos} - \overrightarrow{pBest}) + \varphi_2 \times rand(0,1) \times (\overrightarrow{pos} - \overrightarrow{lBest}), \tag{2.3}$$

onde $rand(0,1)$ é um número real aleatório entre 0 e 1; φ_1 e φ_2 são parâmetros do algoritmo. A correspondência entre as tendências da Seção 2.4 e as parcelas da Equação 2.3 é simples. *Inércia*, *individualismo* e *coletivismo* estão denotados pela primeira, segunda e terceira parcelas, respectivamente. Assim, podemos explicar a

atualização da velocidade como sendo a velocidade anterior perturbada pelas tendências do *individualismo* e *coletivismo*. A intensidade dessas perturbações varia aleatoriamente a cada atualização, respeitando um máximo para cada uma (φ_1 e φ_2).

Neste ponto temos todas as informações para explicitar o algoritmo.

Algoritmo 5 OtimizarSwarm()

inicializar cada partícula em posição e velocidade aleatórias
Enquanto critério de convergência não alcançado **fazer**
  Para toda partícula do enxame **fazer**
    avaliar a função objetivo na posição atual da partícula
    armazenar este valor no vetor temporário $\vec{p}$
    Se $\vec{p} < \overrightarrow{pBest}$ **então**
      $\overrightarrow{pBest} \Leftarrow \vec{p}$
      Se $\vec{p} < \overrightarrow{lBest}$ **então**
        Para toda partícula em *neigh* **fazer**
          atualizar *lBest*;
    atualizar $\vec{v}$ segundo a Equação 2.3
    limitar cada componente de $\vec{v}$ ao intervalo $[V_{min}, V_{max}]$
    atualizar $\overrightarrow{pos}$ segundo a Equação 2.2

2.4.3 Os parâmetros do PSO

Uma característica até então não citada é a limitação da velocidade no intervalo $[V_{min}, V_{max}]$. O fato de as atualizações que ocorrem a cada passo nas velocidades serem estocásticas tem por consequência a possibilidade de que estas cresçam indefinidamente. Caso as velocidades não tivessem seu intervalo restrito, os altos valores que tomariam iriam impedir que as partículas convergissem para qualquer ponto. As constantes V_{min} e V_{max} são parâmetros do PSO. Excetuando-se raríssimos casos, $V_{min} = -V_{max}$, motivo pelo qual nos referiremos apenas a V_{max} no restante do texto. A determinação de um bom valor para este parâmetro é algo que depende de certos conhecimentos do problema. Por exemplo: caso seja configurado de forma que para escapar de um mínimo local seja necessário um passo maior do que V_{max}, então toda partícula que atingir este mínimo local ficará presa até que (ou a menos que) alguma partícula na vizinhança encontre um ponto melhor. A partir daí, a tendência do *coletivismo* atrairá a partícula presa para este novo ponto, possivelmente libertando-a do mínimo local.

Entretanto, não basta escolher um valor arbitrariamente alto para V_{max}. Embora isto contorne o problema descrito no parágrafo anterior, também prejudica a exploração de pontos próximos ao ótimo atual, onde há geralmente grande esperança de melhoria no melhor ponto encontrado.

É interessante notar o efeito que as constantes φ_1 e φ_2 têm sobre o comportamento do algoritmo. Quanto maior for a razão φ_1/φ_2, mais autônomas serão as partículas, o que fará com que a convergência ocorra mais tarde. Em contrapartida, quanto menor for esta razão, mais dependentes umas das outras as partículas serão, o que acarretará em uma convergência mais rápida. Essas constantes devem ser ajustadas empiricamente dependendo do tipo e da dificuldade do problema tratado, pois cada um tem sua velocidade de convergência ideal própria. Mesmo assim, há uma certo consenso na comunidade em aceitar $\varphi_1 = \varphi_2 = x$, e $\{0.9, 1, 2\}$ como bons valores para x.

Como visto em [15], uma partícula idealmente orbita um atrator formado por $\overrightarrow{pBest}$ e $\overrightarrow{lBest}$. Diferentes valores para os parâmetros φ_1, φ_2 e V_{max} alteram de forma definida o comportamento da partícula.

Uma análise do comportamento de uma partícula em condições especiais foi feito em [19], proporcionando uma boa visão da influência dos parâmetros na trajetória. Esta análise é apresentada[2] nesta seção, assim como nas seções 2.4.4 e 2.4.5, e considera um enxame de uma única partícula em um problema de uma única dimensão, onde $pBest$ contém o valor ótimo (0). Nestas condições, é possível simplificar a notação, assumindo $\varphi = \varphi_1 + \varphi_2$. A notação vetorial não é necessária, pois só há uma dimensão.

As Figuras 2.2, 2.3 e 2.4 ilustram as conclusões tiradas sobre o parâmetro V_{max}: quanto menor ele for, menor será o raio da órbita da partícula ao redor do atrator. Nas referidas figuras, $\varphi = 3.9$.

Nas Figuras 2.5, 2.6 e 2.7, o parâmetro V_{max} foi fixado em 2 para a análise do comportamento da trajetória em função de φ. Podemos observar que, à medida em que φ aumenta, a "força gravitacional" que atrai a partícula em direção ao ótimo aumenta. Como consequência, sua trajetória tem a amplitude reduzida, e apresenta cada vez menos passos entre uma passagem pelo ponto ótimo e outra. Podemos notar

[2] Os gráficos utilizados neste capítulo foram retirados de [19].

Figura 2.2: O fenômeno da explosão, que ocorre quando não limitamos a velocidade ($V_{max} = \infty$). Em apenas 150 iterações, a partícula foge da região de interesse.

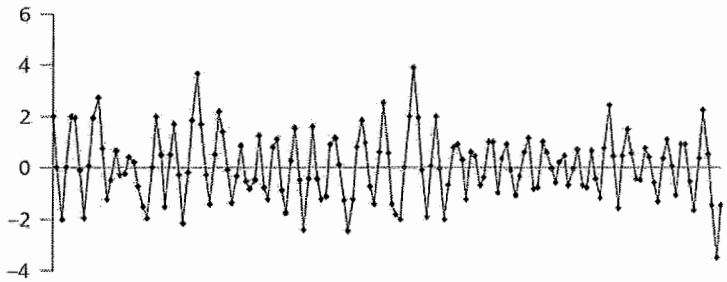

Figura 2.3: Quando $V_{max} = 2$, a partícula orbita o valor de $pBest$ com um raio médio entre 1 e 3.

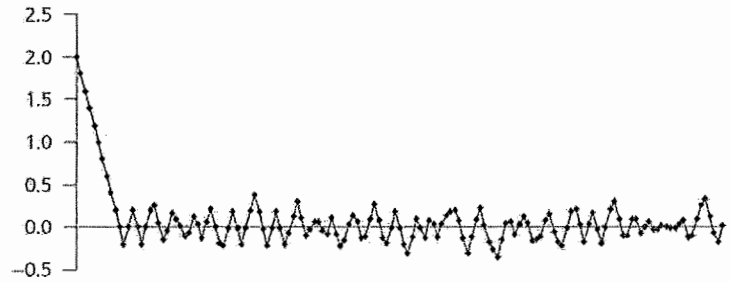

Figura 2.4: Configurando $V_{max} = 0.2$, o movimento da partícula fica restrito a uma região mais estreita ao redor do ótimo.

também, que um valor exageradamente alto para φ satura a trajetória, limitando a exploração a alguns pontos, pois os passos dados são quase sempre em sentidos alternados e de tamanho igual a V_{max}.

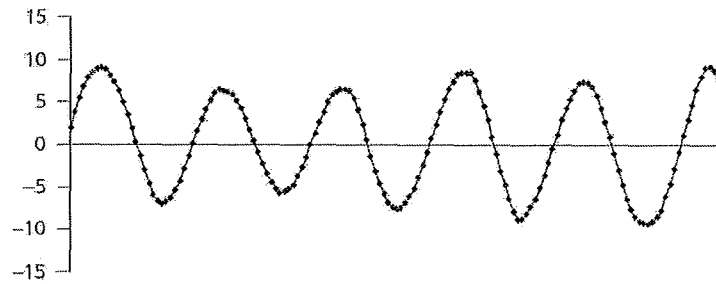

Figura 2.5: Quando $\varphi = 0.1$, a partícula se afasta bastante do ponto ótimo, até que a parcela $(pos - pBest)$ da equação da velocidade se torna grande o suficiente para puxá-la de volta.

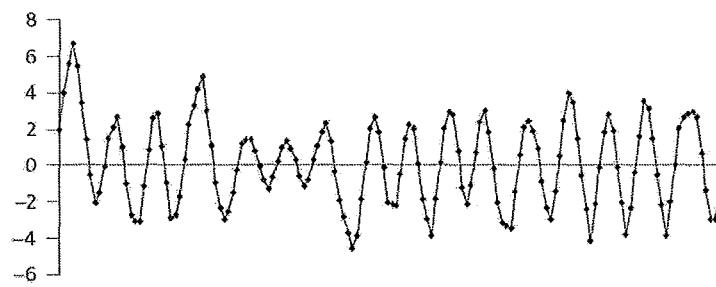

Figura 2.6: Com $\varphi = 1$, a trajetória apresenta o comportamento esperado, uma exploração irregular em torno do ótimo.

As próximas seções descrevem duas das principais variações do PSO clássico, a saber: PSO com inércia (*inertia constrained PSO*) e abordagem por fator de constrição (*constriction factor approach*). Ambas tem por objetivo fazer com que as partículas efetivamente convirjam para um ponto, ao invés de orbitar ao redor dele. A idéia por trás das modificações é aplicar coeficientes de tal forma que, com o passar das iterações, o movimento das partículas seja cada vez mais restrito.

2.4.4 PSO com inércia (*inertia constrained PSO*)

Esta variante, introduzida em [28], é provavelmente a mais difundida do PSO. Um coeficiente φ_0 é aplicado à primeira parcela da equação de atualização da velocidade, que passa a ser como a 2.4. O restante do algoritmo continua exatamente igual ao

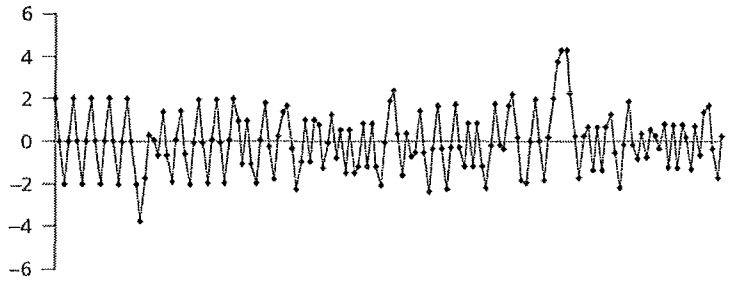

Figura 2.7: Quando $\varphi = 10$, a partícula explora ineficientemente os arredores do ponto ótimo. Dando passos de tamanho V_{max} na maior parte do tempo, ela passa repetidamente pelos mesmos pontos.

descrito na Seção 2.4.2.

$$\vec{v} = \varphi_0 \times \vec{v} + \varphi_1 \times rand(0,1) \times (\overrightarrow{pos} - \overrightarrow{pBest}) + \varphi_2 \times rand(0,1) \times (\overrightarrow{pos} - \overrightarrow{lBest}) \quad (2.4)$$

Assim como os outros parâmetros, a determinação do melhor valor para φ_0 depende do problema a ser resolvido. Geralmente são usados valores fixos menores que, porém próximos de 1, como 0.8, 0.9 ou 0.95. Outra possibilidade é fazer com que φ_0 varie ao longo das iterações. A variação é feita decrementando-se φ_0 linearmente de um valor inicial até um valor final, ambos definidos previamente. Os autores desta variante do PSO sugerem que φ_0 seja decrementado de 0.8 até 0.4.

O que se consegue com o uso do coeficiente de inércia é uma mudança progressiva no comportamento do algoritmo, passando da exploração do espaço de busca como um todo para a exploração de uma pequena região ao redor do(s) valor(es) ótimo(s) encontrado(s) até então. A explicação é simples: à medida em que o PSO progride, as velocidades tendem a diminuir, pois são multiplicadas por valores menores do que 1. Com isso, cada passo dado pelas partículas vai ficando menor, impedindo que as mesmas movam-se para muito longe de seus $\overrightarrow{pBest}$ e $\overrightarrow{lBest}$. Observemos na Figura 2.8 o comportamento desta variante em um experimento nas mesmas condições dos apresentados na Seção 2.4.3.

Podemos traçar um paralelo desta variante com a Têmpera Simulada (Seção 2.2): a desaceleração das partículas é equivalente ao resfriamento do vidro. Embora nem as partículas sejam reaceleradas, nem o vidro seja reaquecido, as chances de sucesso em achar o ponto ótimo aumentam quando diminui-se a velocidade do processo.

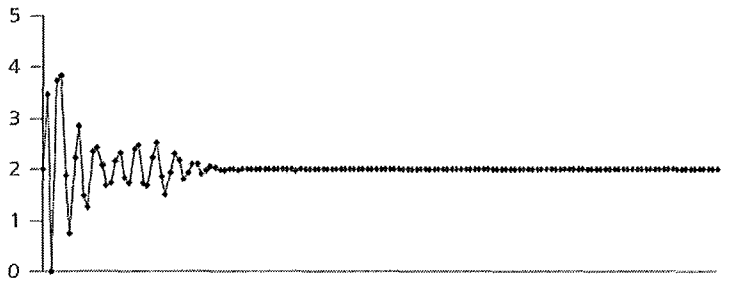

Figura 2.8: O PSO com inércia faz com que a partícula convirja para o atual ponto ótimo com o tempo.

2.4.5 Abordagem por fator de constrição (*constriction factor approach* ou CFA)

Esta variante é fruto de um estudo matemático do comportamento do Sistema de Equações 2.5 realizado em [9], que modela matematicamente o comportamento do PSO clássico.

$$\begin{cases} \vec{v}_{t+1} &= \vec{v}_t + \varphi\vec{y}_t \\ \vec{y}_{t+1} &= -\vec{v}_t + (1-\varphi)\vec{y}_t \end{cases} \tag{2.5}$$

A variável $\vec{y}_t$ é calculada através da Equação 2.6

$$\vec{y}_t = \frac{\varphi_1\overrightarrow{pBest} + \varphi_2\overrightarrow{lBest}}{\varphi_1 + \varphi_2} - \overrightarrow{pos_t} \tag{2.6}$$

A análise mostrou que é possível criar um sistema generalizado, onde a convergência e o fenômeno da explosão das velocidades podem ser controlados, e qua há infinitas maneiras de se fazer isso. Uma delas, denominada *constrição tipo 1"*, consiste na modificação da equação de atualização da velocidade para a Equação 2.7, respeitada a condição de que $\varphi = \varphi_1 + \varphi_2 > 4$.

$$\vec{v} = \chi\Big(\vec{v} + \varphi_1 \times rand(0,1) \times (\overrightarrow{pos} - \overrightarrow{pBest}) + \varphi_2 \times rand(0,1) \times (\overrightarrow{pos} - \overrightarrow{lBest})\Big) \tag{2.7}$$

A variável χ é calculada conforme a Equação 2.8.

$$\chi = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \tag{2.8}$$

onde $\kappa \in [0, 1]$ é um parâmetro que regula a intensidade da constrição. Quanto menor for, mais pronunciado será seu efeito.

O principal diferencial desta variante para o PSO com inércia está na capacidade do CFA de reacelerar as partículas, caso um ponto melhor do que o ótimo atual seja encontrado quando as partículas estão movimentando-se devagar, por estarem próximas da convergência. Este comportamento está exemplificado na Figura 2.9.

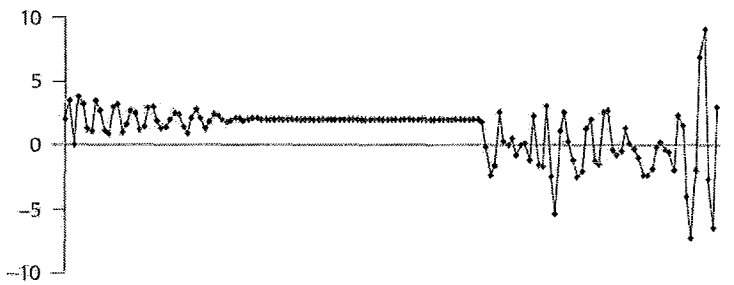

Figura 2.9: Com o CFA, mesmo após a convergência, se um novo ponto ótimo é inserido, a partícula volta a explorar o espaço entre os dois pontos ótimos.

2.4.6 PSO binário

O PSO clássico e suas variantes funcionam bem para problemas cujo espaço de busca é contínuo. Mas será possível aplicar esta técnica a outros grupos de problemas? Particularmente, será possível utilizá-la em situações onde o espaço de busca é discreto, ou até booleano? Em [18] e [8] são propostas alterações no algoritmo para tornar o PSO viável para espaços de busca não-contínuos. A seguir, descrevemos algumas delas. A primeira conserva, dentro dos limites do possível, maior semelhança com o PSO clássico, enquanto a segunda apresenta traços da heurística dos algoritmos genéticos.

2.4.6.1 Heurística 1

Por ser apenas uma discretização da heurística clássica, esta heurística não acrescenta muitos conceitos novos. A estrutura do algoritmo permanece rigorosamente a mesma. O que muda então? Embora a posição da partícula ($\overrightarrow{pos}$) seja agora um vetor de dimensões booleanas — onde *verdadeiro* $= 1$ e *falso* $= 0$ — $\vec{v}$ permanece contínuo. Logo, a única alteração necessária para que a heurística funcione no es-

paço binário deve ser feita na Equação 2.2. Isso é feito de forma que cada posição $\overrightarrow{pos}_d$ em $\overrightarrow{pos}$ seja atualizada conforme a Equação 2.9.

$$pos_d = \begin{cases} 1 & \text{se } \frac{1}{1+e^{-(pos_d+v_d)}} > rand(0,1); \\ 0 & \text{caso contrário.} \end{cases} \tag{2.9}$$

Embora as alterações do PSO clássico para esta versão sejam mínimas, a visão de certos componentes do algoritmo se modificas profundamente, o que requer explicações. O vetor $\vec{v}$ não representa mais a velocidade vetorial da partícula, pois tal conceito não faz muito sentido em um espaço no qual cada dimensão só tem duas posições possíveis. Nesse novo contexto, v_d, para cada dimensão d, representa a pré-disposição da partícula a tomar o valor 0 ou 1 na dimensão d. Quanto menor for v_d, maior a probabilidade de que pos_d, para esta partícula, seja igual a 0, e vice-versa. Valores de módulo extremamente alto para v_d, no PSO clássico, fazem com que a partícula se movimente vigorosamente, possivelmente cobrindo pontos distantes no espaço de busca em poucas iterações. No PSO binário, velocidades deste tipo implicam na indolência da partícula. Por exemplo, caso o módulo de v_d seja muito grande, e seu sinal negativo, a probabilidade de pos_d ser igual a 0 é tão grande que há pouquíssimas iterações do algoritmo nas quais a partícula se movimente de, ou para a posição 1.

O coeficiente de inércia da Seção 2.4.4 também teria um efeito bem diferente do pretendido, caso fosse aplicado a um enxame em um espaço de dimensões booleanas, pois a multiplicação por valores menores do que 1 diminui o módulo de v_d. No PSO clássico, a consequência disso é reduzir a movimentação da partícula até que, com $v_d = 0$, a partícula não se move na dimensão d. No PSO binário, $v_d = 0$ implica em pos_d ser igual a 0 com 50% de probabilidade. Ou seja, este valor para v_d é o que mais incita a movimentação.

2.4.6.2 Heurística 2

Esta heurística realiza modificações mais profundas na estrutura do algoritmo. Uma delas é que aqui não é levada em conta a velocidade da partícula. Para atualizar $\overrightarrow{pos}$, primeiro tomamos cópias de $\overrightarrow{lBest}$ e $\overrightarrow{pBest}$ com alguns bits invertidos; sejam

elas $\overrightarrow{lBest'}$ e $\overrightarrow{pBest'}$, respectivamente. Eis como calcular cada dimensão d da posição da partícula:

$$pos_d = \left\lfloor \frac{pBest'_d + lBest'_d + irand(0,1)}{2} \right\rfloor \tag{2.10}$$

Aqui, $irand(0,1)$ significa escolher aleatoriamente 0 ou 1.

A inversão de bits que produz $\overrightarrow{lBest'}$ e $\overrightarrow{pBest'}$ também é feita de maneira a melhorar a convergência. A cada movimento de uma partícula, um certo número de bits b é invertido nos vetores originais. Quais bits serão invertidos é outra decisão aleatória, mas b descresce regularmente, proporcionalmente ao número de iterações do algoritmo.

Capítulo 3

O Problema do Planejamento da Expansão da Geração

Para aplicar as técnicas estudadas, é necessário escolher um problema-alvo. O interesse em realizar experimentos com problemas reais, de importância significativa em alguma área fez com que escolhêssemos o *Problema do Planejamento da Expansão da Geração* (GEP) como estudo de caso. Este capítulo é baseado em [12].

3.1 Conceituação

O planejamento é entendido como um processo de análise de informações com o objetivo de auxiliar um processo maior de tomada de decisões. No contexto do planejamento da expansão de sistemas de geração de energia elétrica, o conjunto de informações a ser avaliado é extremamente diversificado e vasto. Exemplos de elementos deste conjunto são o custo de combustíveis, o impacto ambiental relacionado à construção de uma usina, ou à projeção do crescimento da demanda de energia.

Devem ser definidos espaços e tempos para a alocação dos recursos necessários para atender a demanda ao longo de um determinado horizonte de planejamento. A solução do problema proporciona um plano que atende aos padrões pré-estabelecidos de qualidade, a um custo mínimo.

Avaliar e dimensionar os recursos energéticos disponíveis para a geração de energia elétrica é uma tarefa demorada que deve ser realizada décadas antes da entrada

em operação das usinas, devido ao longo período de construção das mesmas, especialmente as hidroelétricas. Programas de capacitação tecnológica, estudos de inventário hidroelétrico das bacias hidrográficas, análises de novas tecnologias, projetos de aproveitamento dos recursos e de possíveis usinas térmicas fazem parte desta tarefa anterior à expansão do sistema.

No Brasil, esses estudos são divididos em etapas de análise de horizontes inversamente proporcionais ao detalhamento. Os estudos de longo prazo traçam as linhas gerais de desenvolvimento do sistema, fixando as metas para o programa de expansão de médio prazo, em função das estimativas feitas. Já os estudos de médio prazo estabelecem o programa de expansão propriamente dito, de forma a atender aos requisitos de custo mínimo, qualidade e suprimento. Representam o ajuste do programa frente a variações e imprevistos, como previsões de mercado e restrições financeiras. Têm como principal resultado o Plano Decenal de Expansão de Energia (PDE), formulado pela Empresa de Pesquisa Energética (EPE) [6]. O PDE contém a definição de um cenário de referência para implementação de novas instalações na infra-estrutura de oferta de energia, necessárias para se atender ao crescimento dos requisitos do mercado, e é fruto de extensas pesquisas.

3.2 Situação real

Para proporcionar uma melhor idéia das proporções deste problema, apresentamos aqui alguns dados do PDE 2007/2016 e da Agência Nacional de Energia Elétrica (ANEEL) [5]. A Tabela 3.1 contém a matriz energética brasileira em 14 de abril de 2008, contemplando 1703 usinas. Notemos que, apesar da maior parte da energia ser oriunda de usinas hidroelétricas, há uma grande diversidade nos tipos de usinas da matriz. Cada um desses tipos apresenta particularidades distintas, o que contribui para aumentar ainda mais a complexidade do GEP.

As tabelas 3.2 e 3.3 contêm informações relativas às usinas em construção e as já outorgadas que ainda não iniciaram a construção, com 14/04/2008 como data de referência. Podemos perceber que, para o dado horizonte de planejamento, há centenas de opções de unidades produtoras de energia.

Tabela 3.1: Matriz energética brasileira — 14/04/2008

Tipo		capacidade instalada		%
		Usinas	(kW)	
hidro		674	77.025.406	70,69
Gás	natural	81	10.208.182	9,37
	processo	30	1.146.978	1,05
Petróleo	óleo diesel	578	3.229.434	2,96
	óleo residual	21	1.315.798	1,21
Biomassa	bagaço de cana	248	3.103.283	2,85
	licor negro	13	794.817	0,73
	madeira	27	231.407	0,21
	biogás	3	41.590	0,04
	casca de arroz	3	18.920	0,02
Nuclear		2	2.007.000	1,84
Carvão mineral		7	1.415.000	1,30
Eólica		16	247.050	0,23
Importação			8.170.000	7,89
Total		1.703	108.954.865	100

Tabela 3.2: Empreendimentos em construção

Tipo	Quantidade	Potência outorgada (kW)	%
Central hidroelétrica	1	848	0,01
Central eolielétrica	12	94.000	1,37
Pequena central hidroelétrica	64	1.130.800	16,51
Usina hidroelétrica	21	4.317.500	63,05
Usina termoelétrica	17	1.304.798	19,05
Total	115	6.847.946	100

Tabela 3.3: Empreendimentos outorgados entre 1998 e 2008 (não iniciaram sua construção)

Tipo	Quantidade	Potência outorgada (kW)	%
Central hidroelétrica	70	48.345	0,24
Central eolielétrica	89	4.348.143	21,80
Pequena central hidroelétrica	174	2.693.795	13,51
Usina hidroelétrica	14	2.964.500	14,86
Usina termoelétrica	145	9.891.347	49,59
Total	492	19.946.130	100

3.3 Requisitos e incertezas

Os dois seguintes requisitos fazem parte da modelagem tradicional do GEP:

Requisito econômico está associado principalmente a dois custos: investimento e operação. O custo de investimento reflete despesas com a construção das usinas e linhas de transmissão, ao passo que o custo de operação está relacionado ao custo dos combustíveis para as usinas termoelétricas do sistema.

Confiabilidade serve para garantir o fornecimento adequado mesmo sob condições adversas. Por exemplo: um plano hidroelétrico é considerado adequado se a simulação da operação dado o pior caso (secas e afluências) ocorrido no passado não leva a racionamento. Dentre os aceitáveis, é selecionado o plano de menor custo. Embora de fácil compreensão e implementação, este critério tem sido substituído por outros probabilísticos. O motivo é simples: imaginemos que a seca ocorrida no passado foi extremamente severa. A probabilidade de que outra com as mesmas proporções ocorra em um futuro próximo é pequena. Logo, o critério resultaria em uma subestimação da capacidade de produção e, consequentemente, seria feito um investimento maior do que o necessário. Analogamente, se o cenário no passado tiver sido muito mais favorável do que a média, provavelmente o sistema seria superestimado, e investimentos insuficientes causariam déficits frequentes.

O problema passou então a ser formulado como um cronograma de expansão que minimize o custo atualizado de investimento somado ao custo de operação estimado, sujeito a restrições na probabilidade de falha no atendimento e no risco anual de déficit de energia.

Infelizmente, apenas este exemplo inclui somente uma fatia ínfima das possíveis fontes de incertezas associadas ao sistema. Demanda futura, custos de combustível, juros, restrições financeiras, comportamento da economia e do mercado, tempo de construção das usinas, restrições ambientais e restrições sócio-econômicas acrescentam mais incertezas e, portanto, dificuldades ao planejamento. Incorporar estas informações ao planejamento é um desafio em vários aspectos:

- Por estarem fortemente ligadas a aspectos econômicos, políticos e de organização social, as incertezas acima não podem ser modeladas simplesmente utilizando técnicas de ciências naturais.

- Corre-se o risco de criar um modelo errado pois, como é impossível tratar todas as incertezas envolvidas, reducionismos equivocados podem causar efeitos desastrosos.

- Computacionalmente, o número de aspectos considerados implica em um aumento — muitas vezes maior do que linear — no número de variáveis do problema formulado matematicamente. Isto acarreta a necessidade de recursos maiores, possivelmente inviabilizando a resolução da formulação.

- A formulação matemática necessita de uma medida única de qualidade. É difícil colocar na mesma escala fatores como o impacto ambiental devido à emissão de gases e o custo do carvão em determinada usina.

- O plano visto como um cronograma deixa de ser adequado. Para minimizar o impacto das incertezas passa a ser necessária uma estratégia, que consiga considerar diversos cenários. A vantagem disso é poder levar em conta diversas fontes de incerteza no planejamento. A desvantagem é que esta abordagem pode causar um aumento no tamanho do problema, possivelmente inviabilizando sua resolução em termos computacionais.

3.4 Formulação matemática

Apresentaremos agora a formulação do problema da determinação do plano de expansão ótimo, conforme [1]. Nela, o horizonte de planejamento é dividido em T estágios, e as possibilidades de expansão do sistema são enumeradas dentro das matrizes e vetores.

$$z = \min \sum_{t=1}^{T} \beta_t (cx_t + dy_t) \tag{3.1}$$

sujeito a

$$A_t x_t \geq b_t \tag{3.2}$$

$$\sum_{\sigma=1}^{t} E_\sigma x_\sigma + F_t y_t \geq h_t \tag{3.3}$$

$\forall t \in [1, T]$, onde:

x_t	vetor que contém as opções de construção no estágio t;
c	vetor de custos de construção;
y_t	variáveis de operação no estágio t
	(geração em cada usina, armazenamento dos reservatórios etc);
d	vetor de custos de operação;
β_t	fator de atualização para o estágio t;
b_t, h_t	vetores de recursos;
A_t, E_σ, F_t	matrizes de transformação.

As restrições 3.2 representam limites nas decisões de investimento, proibindo usinas mutuamente exclusivas, regulando datas de obras etc. São chamadas de restrições de unicidade. Já o conjunto 3.3 — restrições operacionais — representa limites de geração, de armazenamento, atendimento à demanda, restrições financeiras e outras.

Nesta formulação, parte-se do princípio de que todos os parâmetros (c, d, A_t, E_σ, F_t, b_t e h_t) são conhecidos a priori. Neste caso a solução x^*, que miniminiza a Equação 3.1, é o plano mais adequado. Entretanto, incertezas que cercam boa parte dos parâmetros levam ao questionamento da otimalidade do plano x^*. Na prática, a realidade futura não será necessariamente constituída pelas previsões. Este questionamento gera várias abordagens para contornar o problema. A seguir descreveremos brevemente algumas delas.

3.4.1 Equivalente determinístico

Esta abordagem visa obter o plano ótimo a partir das melhores previsões disponíveis. No primeiro estágio do planejamento, decisões são tomadas para o próximo estágio baseadas no melhor plano gerado até então. No próximo estágio, por exemplo, cinco

anos após o primeiro, os parâmetros são atualizados com as novas previsões e um novo plano para o horizonte inteiro é criado. Então, as decisões relativas a este estágio são tomadas tendo por base o novo plano. O processo repete-se até o último estágio.

Embora a abordagem reconheça a presença de incertezas e permita a adaptação do plano a cada estágio, ela nem sempre leva ao plano geral mais adequado. Isto se dá porque uma decisão de planejamento para um determinado estágio só é ótima se todas as previsões feitas até então se concretizarem. Caso contrário, a decisão pode não ser ótima (podendo inclusive ser a pior possível!). Por exemplo: a construção de uma dada usina pode fazer parte do plano ótimo caso haja uma previsão de crescimento da demanda futura em uma certa região. Caso esse aumento na demanda não ocorra, a construção não se justifica.

3.4.2 Análise de sensibilidade

A análise de sensibilidade é semelhante ao equivalente determinístico. A melhoria em relação a este é que, após a determinação do plano, uma análise de sensibilidade é feita. O plano é avaliado em duas outras hipóteses de mercado, por exemplo, para demanda acima e abaixo do previsto. A intenção é verificar a robustez do plano, ou seja, se este continua sendo ótimo mesmo com algumas perturbações nos parâmetros. Esta análise tenta refletir o fato de que o plano não é executado exatamente sob as condições previstas. Mesmo assim, há alguns problemas com esta abordagem.

O plano é adequado se não for sensível à variação dos parâmetros. Mas, caso não o seja, nada podemos afirmar. Além disso, a dificuldade da análise cresce de forma diretamente proporcional à quantidade de incertezas consideradas. Outra dificuldade está relacionada ao fato de a abordagem não ser capaz de considerar os ajustes feitos no plano quando são constatadas condições imprevistas: caso seja observado que a demanda está abaixo do previsto, obras possivelmente são postergadas; caso contrário, existe a possibilidade de que elas sejam antecipadas.

3.4.3 Cenários

Aqui, são estabelecidos S cenários $\{c_i,\ d_i,\ A_i,\ E_i,\ F_i,\ b_i,\ h_i\}$, $\forall i \in [1, S]$. Para cada um deles, é computado o plano ótimo, obtendo-se um conjunto de soluções. A patir daí, diversas análises são realizadas, por exemplo: se uma usina é construída em todos os cenários, conclui-se que é uma usina robusta, devendo, portanto, fazer parte do plano ótimo. O problema com esta abordagem é que, como os cenários são distintos, também o serão seus planos de expansão. Devemos lembrar que o objetivo é um plano único, para todos os cenários, meta difícil de ser alcançada a partir de planos específicos para cada um dos cenários.

3.4.4 Otimização estocástica

A abordagem da otimização estocástica visa incorporar ao modelo as incertezas e o processo de decisão associado. O objetivo é determinar o plano de expansão x^* que minimize a soma dos custos de todos os cenários considerados, ponderada pela probabilidade de estes cenários virem a acontecer. Há a necessidade de alterar um pouco a modelagem matemática do problema, que se torna como a seguir:

$$z = \min \sum_{t=1}^{T} (cx_t) + \sum_{s=1}^{S} p_s \sum_{t=1}^{T} (dy_{ts}) \tag{3.4}$$

sujeito a

$$A_t x_t \geq b_t \tag{3.5}$$

$$\sum_{\sigma=1}^{t} E_{\sigma s} x_\sigma + F_{ts} y_{ts} \geq h_{ts} \tag{3.6}$$

$\forall t \in [1, T]$ e $\forall s \in [1, S]$ onde p_s é probabilidade de que o cenário s ocorra.

Por simplicidade, assumimos que $\forall t \in [1, T]$ e $\forall s \in [1, S]$ $\beta_{ts} = 1$, e que os custos de construção e operação são os mesmos independente do cenário e do estágio.

O plano ótimo x^* gerado a partir deste modelo será ótimo em média para os cenários representados. O motivo para o vetor x na Equação 3.4 ser independente dos cenários é que a decisão de construção de usinas é tomada antes de conhecidos os valores de parâmetros (como a demanda). A função objetivo se torna a minimi-

zação da soma do custo de construção com o valor esperado dos custos de operação. Esta abordagem supre todas as deficiências citadas anteriormente. Como não poderia deixar de ser, toda essa melhoria tem seu preço, que é o aumento do número de variáveis no problema. Quanto mais cenários são considerados, mais complexo se torna o problema, e maior o esforço computacional necessário para resolvê-lo. Existem modelos mais elaborados para a resolução do GEP, que consideram ainda mais aspectos da modelagem [12]. Entretanto, as abordagens aqui expostas são suficientes para mostrar sucintamente algumas dificuldades e *tradeoffs* envolvidos na formulação de problemas complexos como este.

Capítulo 4

SATyrus

Neste capítulo introduzimos o SATyrus, uma plataforma para auxiliar na modelagem e resolução de problemas de otimização combinatória. Descrevemos sua linguagem na seção 4.1 e os conceitos que estão por trás de seu funcionamento na seção 4.2.

SATyrus é um sintetizador de funções de energia, que funciona a partir de um problema de otimização combinatória, convenientemente modelado em sua própria linguagem descritiva. Após esta etapa, qualquer resolvedor pseudo-booleano pode ser utilizado buscar a solução ótima do problema, através da minimização do valor da função gerada.

4.1 A linguagem declarativa do SATyrus

Na plataforma SATyrus, é utilizada uma linguagem própria para a formulação dos problemas a serem resolvidos. Nela, a descrição é dividida em arquivos, cuja informação é utilizada para a geração da função de energia. Em [24] e [26] há descrições muito mais detalhadas sobre a entrada do SATyrus. As seções subsequentes têm por objetivo proporcionar apenas conhecimento da linguagem suficiente para a compreensão da modelagem do estudo de caso, no Capítulo 5.

4.1.1 O arquivo principal

É o arquivo que contém as restrições que descrevem o problema a ser solucionado. Único para cada modelagem, é composto por três partes.

4.1.1.1 Definições das estruturas

A primeira parte do arquivo principal contém definições das estruturas que serão utilizadas na formulação das restrições (vide seção 4.1.1.2). O objetivo da estruturação das variáveis é facilitar a leitura e o manuseio do problema modelado. Constantes, vetores e matrizes podem ser definidos aqui. Eis alguns exemplos:

```
cOperacao(3, 2);
nUsinas = 3;
nEstagios = 2;
y(nUsinas, nEstagios, nBits);
cConstrucao read from cConstrucao.txt;
```

4.1.1.2 Restrições

É nesta parte do arquivo onde o problema é modelado de fato. Utilizando as estruturas definidas na primeira parte do arquivo é possível escrever dois tipos de restrição: integridade e otimalidade.

As restrições de integridade são aquelas que devem necessariamente ser satisfeitas para que um solução seja considerada viável. Uma restrição de integridade não-satisfeita normalmente representa um absurdo no domínio do problema. Quando não existe solução que satisfaça a todas as restrições desse tipo, o problema é dito inviável.

Já as restrições de otimalidade são aquelas que queremos minimizar. Custos, prejuízos, distância percorrida por um caixeiro viajante são exemplos do que pode estar denotado por estas restrições de otimalidade. A seguir exemplos de ambos os tipos.

```
integrity group type alfa:  forall{j,s,b}; 1 <= j <= nEstagios;
   s = 7, b = nBits:  not aux2[s][j][b];
optimality group type custo:  forall{i,j}; 1 <= i <= nUsinas,
   1 <= j <= nEstagios:  cOper[i](y[i][j]);
```

Sucintamente, uma restrição é composta por:

- A definição do tipo de restrição - *integrity* para integridade ou *optimality* para otimalidade.

- O grupo ao qual a restrição pertence (*alfa* e *custo* no exemplo dado). Seu significado ficará mais claro na seção 4.1.1.3.

- A definição de variáveis de laço, que são utilizadas como índices para as estruturas que compõem a restrição.

- A definição dos intervalos das variáveis de laço, que limitam os índices para as estruturas.

- A restrição em si, que nada mais é do que uma fórmula em CNF utilizando as posições das estruturas previamente definidas, posições estas que são definidas pelos índices.

Dessa forma, cada restrição neste arquivo define não uma, mas tantas fórmulas lógicas quantas forem as combinações possíveis dos índices utilizados. Podemos ler a restrição de otimalidade do último exemplo — e, de fato, é como o SATyrus interpreta esta linha — como:

cOper[1](y[1][1])∧ cOper[1](y[1][2])∧ cOper[2](y[2][1])∧
cOper[2](y[2][2])∧ cOper[3](y[3][1])∧ cOper[3](y[3][2])

4.1.1.3 Penalidades

Às vezes é necessário priorizar grupos de restrições. Isso acontece principalmente em problemas onde é possível que alguma restrição de integridade seja violada, ou onde as restrições de otimalidade sejam muito complexas para serem enquadradas em um mesmo nível.

A finalidade prática dos níveis de penalidades é que, quando uma restrição de baixo nível for violada, o acréscimo de energia na avaliação da função será menor do que o correspondente a uma violação em outra restrição de alto nível. Mas apenas isto não é sucifiente para garantir que a energia associada a qualquer solução inviável seja maior do que a associada a qualquer solução viável. Por isso, os níveis de penalidades são construídos de maneira que o acréscimo de energia causado pela

violação de uma única restrição do nível i seja estritamente maior do que o acréscimo proveniente da violação de *todas* as restrições dos níveis $i - k$, para quaisquer $i \neq 0$ e $k > 0$. Basta, então, atribuir níveis mais baixos às restrições de otimalidade do que às restrições de integridade para que o comportamento desejado da função seja alcançado. A seguir, um exemplo da parte final do arquivo principal.

```
penalty{alfa is level 1; custo is level 0;}
```

4.1.2 Arquivos de valoração de estruturas

Há dois tipos de estruturas aceitas pelo SATyrus. São eles a *matriz de inicialização neuronal* e a *matriz de valores numéricos*. Através dos nomes fica clara a diferença nos valores armazenados em cada uma: a primeira guarda valores booleanos, enquanto a segunda pode armazenar números inteiros. Entretanto, algumas particularidades não são tão óbvias. Os valores das matrizes neuronais podem ser constantes ou variáveis, dependendo da configuração do atributo *fixo*, presente em cada posição da matriz. Já nas matrizes numéricas esse comportamento não é configurável; todo o seu conteúdo é constante. O conteúdo das matrizes numéricas é, portanto, uma série de coeficientes que são utilizados como multiplicadores de fórmulas lógicas.

É interessante notar que esta simples restrição nas matrizes numéricas obriga que todas as variáveis do problema sejam representadas como valores binários e postas nas matrizes neuronais. É delas que se originam as variáveis da função de energia gerada.

As matrizes neuronais ainda possuem um outro atributo, *prob*, que determina a frequência relativa (aos outros neurônios) de atualização. Este é um atributo que só desempenha algum papel caso utilizemos o resolvedor descrito em [26]. Para o presente trabalho, podemos simplesmente ignorar *prob*.

A sintaxe das matrizes é a seguinte: cada linha no arquivo representa uma entrada na matriz. Os números separados por vírgulas, antes do primeiro hífen são os índices desta entrada. Usam-se tantos números quantas forem as dimensões da matriz. O número após o primeiro hífen é o valor da referida entrada na matriz. Para as matrizes de valores numéricos, a linha termina neste ponto. Na matriz

neuronal há mais dois números com o domínio 0 para *falso* e 1 para *verdadeiro*. O primeiro denota o atributo *fixo*, enquanto o outro denota o atributo *prob*. Não é necessário preencher todas as posições das matrizes neuronais. As negligenciadas serão interpretadas como variáveis do problema, ficando a decisão de seu valor a cargo do resolvedor. A seguir, um exemplo de matriz neuronal seguido por uma matriz numérica.

```
//Matriz de valores numéricos
1-225
2-70
3-77
4-45
5-0

//Matriz de inicialização neuronal
5,1,1-0-0-1
5,1,2-1-0-1
5,1,3-1-0-1
5,1,4-0-0-1
5,1,5-1-0-1
5,1,6-0-0-1
5,1,7-1-0-1
```

4.2 Da formulação para a função de energia

Esta seção dedica-se a explicar o procedimento através do qual a função de energia é gerada a partir dos arquivos de entrada. Os primeiros passos são analisar léxica e sintaticamente os arquivos, afim de verificar que os mesmos foram escritos corretamente. Feito este pré-processamento, resta a conversão em si.

Seja θ_i a representação em lógica proposicional da i-ésima linha no arquivo principal do SAtyrus, desconsiderando o eventual multiplicador c_i. A análise da entrada da plataforma garante que θ_i esteja na CNF. Seja então θ a conjunção de todos os

θ_i, como descrito na Equação 4.1.

$$\theta = \bigwedge_i \theta_i, \qquad \text{onde } \theta_i = \bigvee_j l_{ij} \tag{4.1}$$

e l_{ij} é um literal proposicional $\forall i, j$. Trivialmente, θ também está na forma normal conjuntiva. Estamos interessados em formular matematicamente a equação de energia associada a θ, mas considerando os coeficientes c_i das cláusulas θ_i. Para tal, utilizamos a Equação 4.2 definida em [27], que define energia como a função $H^*(\neg\theta)$.

$$E = H^*(\neg\theta) = \sum_i c_i H(\neg\theta_i) \tag{4.2}$$

Através de simples manipulações lógicas a partir desta e da Equação 4.1, derivamos a Equação 4.3.

$$E = \sum_i c_i H(\neg \bigvee_j l_{ij}) = \sum_i c_i H(\bigwedge_j \neg l_{ij}) \tag{4.3}$$

Combinando este resultado com a energia para a conjunção, da Tabela 4.1, obtemos a Equação 4.4

$$E = \sum_i c_i \prod_j H(\neg l_{ij}), \tag{4.4}$$

que é a utilizada pelo SATyrus para gerar sua função de energia.

Tabela 4.1: Conversão de lógica para energia: p é uma variável proposicional; ι e ψ são fórmulas proposicionais

Lógica	Energia
$H(verdadeiro)$	1
$H(falso)$	0
$H(\neg p)$	$1 - H(p)$
$H(\iota \wedge \psi)$	$H(\iota) \times H(\psi)$
$H(\iota \vee \psi)$	$H(\iota) + H(\psi) - H(\iota \wedge \psi)$

Capítulo 5

A modelagem do GEP no SATyrus

Neste capítulo definimos a instância do GEP que é utilizada como estudo de caso, e também vemos como ela pode ser modelada utilizando a plataforma SATyrus.

5.1 Definição da instância do GEP modelada

Esta seção é dedicada à definição da instância do *Problema do Planejamento da Expansão da Geração* que será modelada na plataforma SATyrus. As tabelas 5.1 e 5.2 contêm as suas características.

Tabela 5.1: Opções de usinas

tipo de usina	capacidade (MW)	construção ($)	operação ($)
1	3285	225000	0
2	3154	70000	20
3	2453	77000	20
4	1501	45000	20
5	∞	-	300

Tabela 5.2: Demanda esperada

estágio	1	2	3
demanda de energia (MW)	4380	6570	8760

Utilizamos a abordagem do equivalente determinístico, descrita na Seção 3.4.1,

para que o número de variáveis mantenha-se tratável. Assim, a formulação do problema é descrita através dos grupos de equações 5.1, 5.2, 5.3 e 5.4.

$$\begin{aligned}
&\text{minimizar } 225(x_{11}+x_{12}+x_{13})+70(x_{21}+x_{22}+x_{23})+ \\
&\quad 77(x_{31}+x_{32}+x_{33})+45(x_{41}+x_{42}+x_{43})+300(y_{51}+y_{52}+y_{53})+ \\
&\quad 20(y_{21}+y_{31}+y_{41}+y_{22}+y_{32}+y_{42}+y_{23}+y_{33}+y_{43})
\end{aligned} \tag{5.1}$$

sujeito a

$$\begin{aligned}
-x_{11}-x_{12}-x_{13}+1 &\geq 0 \\
-x_{21}-x_{22}-x_{23}+1 &\geq 0 \\
-x_{31}-x_{32}-x_{33}+1 &\geq 0 \\
-x_{41}-x_{42}-x_{43}+1 &\geq 0
\end{aligned} \tag{5.2}$$

$$\begin{aligned}
y_{11}-3285x_{11} &\leq 0 \\
y_{12}-3285x_{11}+x_{12} &\leq 0 \\
y_{13}-3285x_{11}+x_{12}+x_{13} &\leq 0 \\
y_{21}-3504x_{21} &\leq 0 \\
y_{22}-3504x_{21}+x_{22} &\leq 0 \\
y_{23}-3504x_{21}+x_{22}+x_{23} &\leq 0 \\
y_{31}-2453x_{31} &\leq 0 \\
y_{32}-2453x_{31}+x_{32} &\leq 0 \\
y_{33}-2453x_{31}+x_{32}+x_{33} &\leq 0 \\
y_{41}-1051x_{41} &\leq 0 \\
y_{42}-1051x_{41}+x_{42} &\leq 0 \\
y_{43}-1051x_{41}+x_{42}+x_{43} &\leq 0
\end{aligned} \tag{5.3}$$

$$
\begin{aligned}
y_{11} + y_{21} + y_{31} + y_{41} + y_{51} - 4380 &\geq 0 \\
y_{12} + y_{22} + y_{32} + y_{42} + y_{52} - 6570 &\geq 0 \\
y_{13} + y_{23} + y_{33} + y_{43} + y_{53} - 8760 &\geq 0
\end{aligned}
\tag{5.4}
$$

para $x_{ij} \in \{0, 1\}$, $y_{ij} \in \mathbb{R}$, $\forall i \in [1, N_u]$, $\forall j \in [1, N_e]$, $N_u = 5$ e $N_e = 3$

A Equação 5.1 é a função objetivo do problema, que é transformada em restrições de otimalidade. Corresponde à soma dos custos de construção com os custos de operação, em cada usina, em cada estágio do planejamento. As Equações 5.2 são restrições de integridade, impedindo que uma mesma usina seja construída mais de uma vez (usinas mutuamente exclusivas). Por sua vez, o Grupo de Equações 5.3 também é convertido em restrições de integridade, estas com dupla funcionalidade: garantir que só haja produção de energia em usinas já construídas e assegurar que a produção máxima em cada usina por estágio não exceda a capacidade da usina. Por fim, o Grupo de Equações 5.4, que também é transformado em restrições de integridade, serve para impedir que um plano deixe de atender à demanda em qualquer estágio, mapeando os requisitos de confiabilidade do GEP.

A primeira usina da instância representa uma opção hidroelétrica, de alto custo de construção e custo de operação nulo, já que não utiliza combustível algum. A segunda, a terceira e a quarta usina são opções termoelétricas de diferentes portes, com algum custo de operação e baixo custo de construção. Já a última usina da instância não representa uma opção real. Constitui apenas um artifício para que, caso a demanda exceda a soma das capacidades das outras usinas, a elaboração de um plano viável seja possível. Em tal situação, esta usina de déficit atende à demanda excedente. A consequência disto é que passam a ser aceitos como viáveis planos que apresentam pequenos déficits no atendimento à demanda.

Cada estágio do planejamento representa um horizonte de cinco anos. Para simplificar a modelagem, não estão sendo consideradas variações nos custos em função do estágio; são todos fixos.

5.2 A representação numérica

A primeira providência a ser tomada é determinar uma maneira de representar os inteiros presentes no problema como variáveis booleanas. Foi escolhida a representação binária de 14 bits para o número inteiro, mais 1 bit pra o sinal. Com esta representação (de vetores de tamanho $N_b = 15$) é possível trabalhar com números até 16383, o que é suficiente para esta instância.

Foi cogitada a possibilidade de representar números como faixas de valores, como em [21]. Esta abordagem reduz bastante o número de variáveis necessárias para a representação de um número inteiro. Entretanto, parte da simplicidade da modelagem é sacrificada, pois correspondências entre as faixas de valores devem ser estabelecidas, e um mapeamento extra para operações de soma e subtração se faz necessário. Por esses motivos, preferimos descartar esta possibilidade.

Cada variável y_{ij}, $\forall i \in [1, N_u]$, $\forall j \in [1, N_e]$ da formulação matemática é mapeada para um vetor de N_b variáveis binárias y_{ijb}, $b \in [1, N_b]$. Isto não é necessário para $x_{ij} \forall i \in [1, N_u]$, $\forall j \in [1, N_e]$ que, por não representarem números inteiros, podem ser diretamente mapeadas em variáveis booleanas.

5.3 O somador binário

Transposto este obstáculo, devemos agora definir como serão realizadas as operações aritméticas de soma e subtração necessárias para a representação do problema. Lembremo-nos de que a linguagem do SATyrus contempla apenas fórmulas lógicas e coeficientes; não somas. A solução encontrada (em [26]) foi escrever um somador binário utilizando fórmulas lógicas. A solução porém, introduz um desagradável *overhead*, tanto no número de variáveis quanto no número de restrições. Para cada soma de duas parcelas de b bits são necessárias duas estruturas adicionais: uma de b bits para armazenar o resultado da soma, e outra de $b + 1$ bits para armazenar o "vai 1" (*carry*) da operação.

Como exemplo de uso do somador binário, escrevemos um programa para somar os números `n1` = 3 com `n2` = 5, armazenando o resultado em `res`. As parcelas têm 4 bits mais o de sinal. Ao observar o código-fonte deste programa (que está na

Seção A.1 do Apêndice A), podemos perceber a quantidade relativamente elevada de restrições necessárias para sua modelagem. Para o SATyrus, cada violação em uma das regras do somador binário implica em um acréscimo na energia. A solução com energia mínima não viola nenhuma das restrições de integridade, já que estamos tratando de um problema viável. Como, além disso, não há restrições de otimalidade, a energia associada à solução ótima é igual a 0. É interessante notar que, para este simples problema de soma, a função de energia gerada tem 21 variáveis booleanas.

5.4 Definição das constantes e estruturas

Nesta seção definimos as constantes e estruturas necessárias para a modelagem do GEP, começando com as variáveis de decisão do problema. Conforme descrito na Seção 5.2, as variáveis de decisão sobre a construção das usinas (x_{ij}) são diretamente mapeadas em uma matriz booleana de dimensões $[N_u \times N_e]$. As variáveis de decisão sobre a operação (y_{ij}) necessitam de N_b variáveis booleanas cada. Sua estrutura consiste, então, em uma matriz booleana de dimensões $[N_u \times N_e \times N_b]$.

Além de x e y, são necessárias outras estruturas auxiliares, cuja utilidade será melhor compreendida na Seção 5.5. Por hora, seguem suas descrições:

capacidade: Estrutura que contém as capacidades de produção energética das usinas. Devido ao seu envolvimento em operações aritméticas, é necessário que esteja representada como um vetores de bits, assim como y. Devido a uma opção de implementação, seu conteúdo está representado em complemento a 2, e suas dimensões são $[N_u \times N_b]$.

demanda: Contém a demanda por estágio. Assim como a **capacidade**, também necessita da representação como vetores de bits. Suas dimensões são $[N_e \times N_b]$.

aux1 e aux2: Estruturas auxiliares para armazenar resultados intermediários de somas e subtrações. As dimensões são $[N_u \times N_e \times N_b]$ e $[(N_u + 2) \times N_e \times N_b]$, respectivamente.

vai1, vai2 e vai3: Estruturas auxiliares para armazenar o "vai 1" (*carry*) das somas e subtrações da modelagem. Também são vetores de bits representando

inteiros, e têm por dimensões $[N_u \times N_e \times (N_b + 1)]$, $[N_u \times N_e \times (N_b + 1)]$ e $[1 \times N_e \times (N_b + 1)]$, respectivamente.

cConstrucao: Aqui estão contidos os custos de construção de cada usina. Como esses dados não são usados em nenhuma operação aritmética, esta estrutura pode ser representada como um vetor numérico, de dimensão $[N_u]$.

cOperacao: Apresenta forma e dimensões idênticas à **cConstrucao**, mas guarda os custos de operação das usinas.

As constantes necessárias são N_u, N_e e N_b, mapeadas, respectivamente em: `nUsinas`, `nEstagios` e `nBits`. Eis abaixo a parte do arquivo principal contendo as declarações das constantes e estruturas para o estudo de caso.

```
nUsinas=5;
nEstagios=3;
nBits=15;
capacidade(nUsinas, nBits);
demanda(nEstagios, nBits);
vai1(nUsinas, nEstagios, nBits+1);
vai2(nUsinas, nEstagios, nBits+1);
vai3(1, nEstagios, nBits+1);
aux1(nUsinas, nEstagios, nBits);
aux2(nUsinas+2, nEstagios, nBits);
xis(nUsinas,nEstagios);
ypsilon(nUsinas, nEstagios, nBits);
cConstrucao read from cConstrucao.txt;
cOperacao read from cOperacao.txt;
```

5.5 Definição das restrições

A próxima tarefa é modelar as restrições 5.1, 5.2, 5.3 e 5.4 na linguagem do SATyrus. Uma parte dessas restrições utiliza os somadores binários da Seção 5.3. Para não complicar a leitura, neste capítulo as restrições relativas ao somador serão substituídas pela forma compacta a seguir.

$\texttt{sum}(p_1[dimens_{p_1}],\ p_2[dimens_{p_2}],\ r[dimens_r],\ c[dimens_c])$

Esta forma *não* pertence à linguagem do SATyrus. Esta linha sintetiza as restrições de soma para a operação $r = p_1 + p_2$, com os *carries* sendo guardados em c. O número de bits das estruturas é sempre $N_b = 15$, e $dimens_X$ é substituído pelo conjunto de índices envolvidos na operação para a estrutura X. Isto permite a cada linha nesta forma representar mais de uma operação. O código fonte completo, com as restrições dos somadores explicitadas pode ser encontrado no Apêndice A, Seção A.2.

5.5.1 Restrições de construção

Impedem que uma mesma usina seja construída mais de uma vez. A modelagem dessas restrições se assemelha bastante à da própria formulação matemática (5.2) pois, desde a definição do problema, as variáveis x são booleanas. O conjunto de destrições a ser modelado, em lógica proposicional é como na Equação 5.5.

$$\neg(x_{ij} \wedge y_{ij}), \tag{5.5}$$

para $i \in [1, N_u]$, $j \in [1, N_e]$, $i \neq j$. Na linguagem do SATyrus, isto significa:

```
integrity group type alfa:  forall{i,j,k}; 1<=i<=nUsinas,
   1<=j<=nEstagios, 1<=k<=nEstagios; j!=k:  (not x[i][j] or
   not x[i][k]);
```

5.5.2 Restrições de operação

Na formulação original, estas restrições servem tanto para impedir que uma usina produza mais do que permite a sua capacidade quanto para assegurar que usinas que não foram construídas também não produzam energia. No SATyrus, há a necessidade de criar um conjunto de restrições para cada uma dessas finalidades.

Para modelar as restrições que protegem as capacidades das usinas, somamos y_{ij} com $capacidade_i$ bit a bit $\forall i \in [1, N_u]$, $\forall j \in [1, N_e]$. Além disso, inserimos uma restrição que obriga o bit de sinal destas somas a ser igual a 1, ou seja, y_{ij} deve assumir valores tais que a soma seja um número negativo.

```
integrity group type alfa:  forall{i,j,b}; 1<=i<=nUsinas,
   1<=j<=nEstagios, 1<=b<=nBits:  sum(capacidade[i][b],
   y[i][j][b], aux1[i][j][b], vai1[i][j][b]);
integrity group type alfa:  forall{i,u,d,t,s}; 1<=i<=nUsinas;
   u=1, d=2, t=3, s=nBits:  aux1[i][u][s] and aux1[i][d][s]
   and aux1[i][t][s];
```

Garantir que apenas usinas construídas produzam energia é simples. Um pouco mais formalmente, o que queremos é o denotado pelas expressões equivalentes 5.6 e 5.7.

$$\neg y_{ij} \vee \quad (\bigvee_{k} x_{kj}) \tag{5.6}$$

$\forall i \in [1, N_u]$, $\forall j \in [1, N_e]$, $\forall k \in [1, i]$. Ou seja:

$$\begin{aligned} &\neg y_{1j} \vee x_{1j} \wedge \\ &\neg y_{2j} \vee x_{1j} \vee x_{2j} \wedge \\ &\neg y_{2j} \vee x_{1j} \vee x_{2j} \vee x_{3j}, \end{aligned} \tag{5.7}$$

$\forall j \in [1, N_e]$. Traduzindo para a linguagem do SATyrus, temos:

```
integrity group type alfa:  forall{i,u,b}; 1<=i<=nUsinas,
   1<=b<=nBits-1; u=1:  (not y[i][u][b] or x[i][u]);
integrity group type alfa:  forall{i,u,d,b}; 1<=i<=nUsinas,
   1<=b<=nBits-1; u=1, d=2:  (not y[i][d][b] or x[i][u] or
   x[i][d]);
integrity group type alfa:  forall{i,u,d,t,b}; 1<=i<=nUsinas,
   1<=b<=nBits-1; u=1, d=2, t=3:  (not y[i][t][b] or x[i][u]
   or x[i][d] or x[i][t]);
```

5.5.3 Restrições de demanda

O último conjunto de restrições de integridade assegura o atendimento à demanda. Primeiro somamos, por estágio, a energia gerada por todas as usinas, armazenando o resultado em uma estrutura auxiliar. Em seguida, a demanda, também por estágio,

é subtraída da soma correspondente. Na verdade a operação realizada é uma soma, pois a demanda, assim como a capacidade, é representada por números negativos, em complemento a 2. Por fim, o valor obtido, armazenado também em uma estrutura auxiliar, deve ser positivo, o que é assegurado por uma restrição que obriga seu bit mais significativo a assumir o valor 0.

```
integrity group type alfa:  forall{i,j,b}; 1<=i<=nUsinas,
   1<=j<=nEstagios, 1<=b<=nBits:  sum(y[i][j][b], aux2[i][j][b],
   aux2[i+1][j][b], vai2[i][j][b]);
integrity group type alfa:  forall{j,b,i,u}; 1<=j<=nEstagios,
   1<=b<=nBits; i=6, u=1:  sum(aux2[i][j][b], demanda[j][b],
   aux2[i+1][j][b], vai3[u][j][b]);
integrity group type alfa:  forall{j,s,b}; 1<=j<=nEstagios;
   s=7, b=nBits:  not aux2[s][j][b];
```

5.5.4 Função objetivo

Finalmente, analisemos a modelagem das restrições de otimalidade, que consistem na soma de dois custos: o de construção e o de operação:

$$\sum_{i=1,j=1}^{N_u,N_e} (cConstrucao_i \times x_{ij})$$
$$\sum_{i=1,j=1}^{N_u,N_e} (cOperacao_i \times y_{ij})$$

Aqui, não é necessário utilizar o somador binário, já que as somas envolvidas podem ser representadas como conjunções de cláusulas lógicas, sendo convertidas pelo próprio SATyrus durante o processo de geração da função de energia. As multiplicações também não implicam na necessidade do somador pois, como `cConstrucao` e `cOperacao` são matrizes numéricas, podem ser utilizadas como coeficientes nas cláusulas. Assim, podemos modelar as restrições de otimalidade desta instância como segue a seguir.

```
optimality group type custo:  forall{i,u,d,t}; 1<=i<=nUsinas;
```

```
    u=1, d=2, t=3:  cConstrucao[i]((x[i][u] or x[i][d] or
    x[i][t]));
optimality group type custo:  forall{i,j,b}; 1<=i<=nUsinas,
    1<=j<=nEstagios, 1<=b<=nBits-1:  cOperacao[i][b](y[i][j][b]);
```

5.5.5 Penalidades

Todas as restrições de integridade são postas no mesmo grupo (`alfa`), ou seja, é atribuída a cada uma delas a mesma importância relativa. Para a modelagem do GEP, não foram realizados estudos muito aprofundados sobre os efeitos de diferentes penalidades para as diferentes restrições de integridade. A existência ou não de outros níveis não torna esta modelagem errada ou certa, pois qualquer solução viável não viola nenhuma dessas restrições. O efeito possível é uma mudança no grau de dificuldade de otimização da função de energia, responsabilidade do resolvedor.

As restrições de otimalidade são colocadas em um nível mais baixo do que as de integridade (`custo`), já que as primeiras são inevitavelmente violadas de alguma forma, e não queremos que o resolvedor decida produzir energia em uma usina não-construída em prol de uma redução nos custos totais, por exemplo. Eis o trecho do arquivo principal relativo às penalidades:

```
penalty{
alfa is level 1;
custo is level 0;}
```

5.5.6 Ressalvas

Para que as restrições tenham o comportamento desejado, é fundamental que as estruturas auxiliares tenham seus valores fixados corretamente. Por exemplo, na Seção 5.5.3, a estrutura `aux2[i+1][j]` $\forall i \in [1, N_u]$, $j \in [1, N_e]$, vista como um número inteiro, armazena o valor de `aux2[i][j]` somado ao valor de `y[i][j]`. Mas quando $i = 1$, o que está sendo somado a `y[1][j]`? Em grande parte dos resolvedores estocásticos, incluindo o PSO, as variáveis são inicializadas aleatoriamente. Se nenhuma providência fosse tomada para garantir que `aux2[1][j]` é igual a 0, esta

parte da estrutura poderia ser inicializada com qualquer valor, introduzindo um erro no resultado do somatório da energia produzida nas usinas.

No SATyrus, é possível contornar este problema de duas maneiras. A primeira é configurar os valores corretos para as estruturas nos próprios arquivos de estruturas, lembrando de configurar também o atributo *fixo* para 1. Assim, as posições nas estruturas configuradas desta forma aparecem como termos constantes na função de energia. Outra opção é criar restrições de integridade que associem violações aos valores diferentes dos pretendidos. A seguir ilustramos as duas possibilidades.

```
integrity group type alfa:  forall{j,b,u}; 1<=j<=nEstagios,
   1<=b<=nBits; u=1:  not aux2[u][j][b];
integrity group type alfa:  forall{i,j,b}; 1<=i<=nUsinas,
   1<=j<=nEstagios; b=1:  not vai1[i][j][b];
integrity group type alfa:  forall{i,j,b}; 1<=i<=nUsinas,
   1<=j<=nEstagios; b=1:  not vai2[i][j][b];
integrity group type alfa:  forall{j,u,b}; 1<=j<=nEstagios;
   u=1, b=1:  not vai3[u][j][b];
integrity group type alfa:  forall{i,j,b}; 1<=i<=nUsinas,
   1<=j<=nEstagios; b=15:  not ypsilon[i][j][b];
```

Arquivo aux2 (seção):

```
1,1,1-0-1-1
1,1,2-0-1-1
1,1,3-0-1-1
1,1,4-0-1-1
1,1,5-0-1-1
1,1,6-0-1-1
1,1,7-0-1-1
1,1,8-0-1-1
1,1,9-0-1-1
1,1,10-0-1-1
1,1,11-0-1-1
```

```
1,1,12-0-1-1
1,1,13-0-1-1
1,1,14-0-1-1
1,1,15-0-1-1
```

Existe uma seção como a acima para cada estágio, para cada usina.

Arquivo vai1 (seção):

```
1,1,1-0-1-1
```

As estruturas `vai1`, `vai2` e `vai3` precisam ter o bit menos significativo (em respeito à dimensão $b \in [1, N_b]$) fixo em 0. Do contrário, existe a possibilidade da introdução de erros nos somadores binários.

5.6 A função de energia

A função de energia gerada a partir da modelagem descrita neste capítulo ainda precisa de alguns ajustes antes de ser fornecida para o resolvedor PSO. A informação sobre variáveis cujos valores são fixos não vem escrita diretamente na função, ficando armazenada em um arquivo separado, que contém as informações de cada variável. Para incorporar esses dados à função de energia, substituímos nesta cada variável fixa por seu valor. Este artifício reduz em aproximadamente 10% o número de variáveis da função de energia, reduzindo também sua complexidade e seu número de termos. Consequentemente, o esforço computacional para realizar cada avaliação também é reduzido. Dados o número de experimentos realizados e a quantidade de avaliações necessárias por experimento, é seguro afirmar que este recurso é de grande valia para acelerar o andamento da experimentação.

A função (otimizada) tem 1308 variáveis e 8259 termos da forma $c_i \times v_j \times \ldots \times v_k$, onde $c_i \forall i \in [1..8259]$ é uma constante e $v_j \forall j \in [1..1308]$ é uma variável booleana do problema. Embora o número de variáveis por termo varie, neste estudo de caso ele nunca excede quatro. Abaixo podemos ver um pequeno trecho (0.25%) da função gerada pelo SATyrus

$$(980 \times x_{472} \times x_{457} \times x_{1300} \times x_{1056}) + (980 \times x_{473} \times x_{458} \times x_{1301} \times x_{1057}) +$$
$$(980 \times x_{474} \times x_{459} \times x_{1302} \times x_{1058}) + (980 \times x_{475} \times x_{460} \times x_{1303} \times x_{1061}) +$$
$$(980 \times x_{476} \times x_{461} \times x_{1304} \times x_{1062}) + (980 \times x_{477} \times x_{462} \times x_{1305} \times x_{1063}) +$$
$$(980 \times x_{478} \times x_{463} \times x_{1306} \times x_{1064}) + (980 \times x_{479} \times x_{464} \times x_{1428} \times x_{1065}) -$$
$$(490 \times x_{240} \times x_{1083} \times x_{255}) - (490 \times x_{241} \times x_{1084} \times x_{256}) -$$
$$(490 \times x_{242} \times x_{1085} \times x_{257}) - (490 \times x_{243} \times x_{1086} \times x_{258}) -$$
$$(490 \times x_{244} \times x_{1087} \times x_{259}) - (490 \times x_{245} \times x_{1088} \times x_{260}) -$$
$$(490 \times x_{246} \times x_{1089} \times x_{261}) - (490 \times x_{247} \times x_{1090} \times x_{262}) -$$
$$(490 \times x_{248} \times x_{1091} \times x_{263}) - (490 \times x_{249} \times x_{1092} \times x_{264}) -$$
$$(490 \times x_{250} \times x_{1093} \times x_{265}) - (490 \times x_{251} \times x_{1094} \times x_{266}) -$$
$$(490 \times x_{252} \times x_{1095} \times x_{267})$$

Capítulo 6

Experimentos, resultados e avaliações

Neste capítulo apresentamos os experimentos realizados com o GEP modelado para a plataforma SATyrus e o resolvedor PSO binário. Na Seção 6.1, descrevemos sucintamente o ambiente de testes utilizado. Na seção seguinte, 6.2, descrevemos a metodologia utilizada para os testes. Na Seção 6.3, são apresentados os resultados dos experimentos. Feito isso, estamos aptos para mostrar a análise dos resultados na Seção 6.4. Finalmente, alguns comentários complementares seguem na Seção 6.5.

6.1 Ambiente de testes

Os experimentos aqui apresentados foram executados em diversos ambientes paralelos, com recursos computacionais variados. As descrições destes ambientes, a seguir, são referentes à data da elaboração deste trabalho:

LOA O cluster do LOA conta com oito máquinas com processadores Intel® Pentium® D de 3.0GHz e 2GB de RAM;

LabIA No laboratório de inteligência artificial do Programa de Engenharia de Sistemas da COPPE, o ambiente não é muito homogêneo, contando com cinco máquinas equipadas com processadores Intel® Pentium® D de 2.8GHz e 1GB de RAM e uma Intel® Core® 2 Duo de 2.13GHz e 4GB de RAM. Além disso, há um cluster composto por dezesseis máquinas, sendo oito delas Intel® Pentium® D de 3GHz com 2GB de RAM e oito Intel® Pentium® HT de 3.2GHz com 1GB de RAM;

EELA O grid do EELA [2] é o menos homogêneo dos ambientes onde os experimentos foram executados, o que faz parte de sua natureza altamente distribuída. Contém centenas de computadores com inúmeras configurações diferentes.

A quantidade de memória RAM não é mencionada nesta seção por ser irrelevante. O PSO binário, assim como o contínuo, tem por característica a necessidade de pouco espaço de armazenamento para o funcionamento eficiente. Na verdade, a quantidade deste recurso utilizada é constante durante toda a execução do resolvedor, apenas o suficiente para que sejam mantidas as estruturas de cada partícula, conforme a Seção 2.4. Nas máquinas do LabIA, por exemplo, a memória utilizada durante o maior experimento não atinge ao menos 10% da disponível.

Cada um dos ambientes possui particularidades que vão além do tipo e número de CPUs disponíveis. Fatores como a disponibilidade e o nível de compartilhamento dos recursos precisam ser levados em conta para o estabelecimento de uma métrica imparcial para a medição do tempo de execução. Devido a todas estas heterogeneidades nos ambientes, utilizamos outra medida para comparar o tempo de execução, explicada na Seção 6.2.

6.2 Metodologia

Esta seção dedica-se à explicação dos parâmetros utilizados nos experimentos e das informações retiradas deles.

No PSO binário, como visto no Capítulo 2, há uma certa quantidade de parâmetros configurados manualmente. Em teoria, os valores atribuídos a eles podem ter um grande impacto no desempenho do resolvedor. Nossos experimentos visam a verificação dessa asserção e a avaliação do comportamento do PSO utilizando diferentes configurações de parâmetros.

Baseado em [3], desenvolvemos uma implementação do PSO em C++. Cada experimento consiste em cinquenta execuções do resolvedor para uma dada configuração dos parâmetros. Mais de uma execução por configuração é necessária devido à natureza estocástica do PSO, tornando as medições mais precisas e confiáveis.

Cada execução é finalizada quando passam-se quinhentas iterações sem que o

melhor ponto encontrado pelo enxame sofra alteração. Outros possíveis critérios de convergência seriam:

número de iterações igual a k, $k \in \mathbb{N}$ É o critério mais simples de ser implementado, mas não é adequado para nossos experimentos. O número de iterações necessárias para a convergência difere em algumas ordens de grandeza, dependendo da configuração dos parâmetros. Assim, teríamos duas opções. A primeira seria que o número máximo de iterações fosse determinado a priori para cada experimento, o que implicaria na repetição dos mesmos até que o mínimo número necessário de iterações fosse encontrado. A segunda seria utilizar um único limite superior suficiente para que todas as configurações convergissem, o que desperdiçaria muitos recursos computacionais, pois as configurações que convergem mais rapidamente continuariam executando por muito tempo, mesmo após a convergência.

distância entre as partículas menor do que ϵ, $\epsilon \in \mathbb{R}$ Este é o critério mais preciso, mas preferimos não utilizá-lo porque aumentaria o tempo de execução dos experimentos, já que cálculos extras passariam a ser feitos periodicamente. Além disso, existe a possibilidade de que as partículas convirjam em blocos, tornando necessário o emprego de heurísticas complexas para a determinação destes, possivelmente introduzindo mais parâmetros e a necessidade de mais experimentos para a validação dos mesmos.

Em [8] é sugerida uma fórmula para o tamanho do enxame (S) ideal, em função do número de dimensões do problema. Para o caso estudado, com 1308 dimensões, $S = 170$. Entretanto, a fórmula é válida apenas para problemas menores, com até 500 dimensões. Por isso, variamos o parâmetro S utilizando valores de 10, 40, 180 e 250 neste trabalho.

No que concerne os parâmetros φ_1 e φ_2, utilizamos o material apresentado na Seção 2.4.3 para obter pontos de partida para seus valores razoáveis. Os experimentos utilizam os seguintes valores para φ_1 e φ_2: 0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1, 5 e 10. Os valores para V_{max} também foram escolhidos baseados nas considerações da Seção 2.4.3, e compreendem: 1, 2, 3, 4, 5, 7, 8, 10, 11, 20, 30, 40 e 60. Para o parâmetro φ_0, utilizamos o valor mais frequente na literatura

(0.95), e também valores que julgamos produzir um efeito similar ao da inércia do PSO contínuo: 1.05 e 2. Para a compreensão dos modelos e tamanhos de vizinhança utilizados, são necessárias definições adicionais, encontradas na Seção 6.2.2.

Dos testes são extraídos dois conjuntos de informações. O primeiro diz respeito ao resultado efetivo de cada experimento, medindo o grau de adequação das soluções encontradas por ele. A métrica utilizada para esta informação é a energia média, calculada como a média das cinquenta soluções do experimento. O segundo conjunto de informações serve para medir o tempo necessário para a execução de cada experimento, e foi medido através do cálculo da média do número de avaliações da função de energia até que o critério de convergência fosse atingido. Vale ressaltar que a cada iteração, o PSO faz tantas avaliações quantas forem as partículas no enxame, de maneira que o número médio de iterações do algoritmo, por experimento, pode ser calculado por M/S, onde M denota a média de avaliações medida nos experimentos.

Apresentamos a seguir, nas seções 6.2.1, 6.2.2, 6.2.3, 6.2.4 e 6.2.5, as configurações de parâmetros para cada um dos experimentos, que estão divididos em grupos afim de facilitar a visualização. Um grupo compreende tantos experimentos quantas forem as combinações dos parâmetros descritas neles, explicadas com mais detalhes nas seções correspondentes.

6.2.1 Grupo *global 1*

Os experimentos deste grupo constituem uma primeira exploração do efeito dos parâmetros no comportamento do PSO binário clássico. Combinando todo o leque de valores para os parâmetros, esperamos determinar em quais faixas o resolvedor se comporta melhor.

- *global 1* (728 experimentos)
  - vizinhança global
  - $\varphi_0 \in \{1\}$
  - $\varphi_1 = \varphi_2 \in \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1, 5, 10\}$
  - $S \in \{10, 40, 180, 250\}$

- $V_{max} \in \{1, 2, 3, 4, 5, 7, 8, 10, 11, 20, 30, 40, 60\}$

6.2.2 Grupos *local 1*, *local 2* e *local 3*

Para complementar os experimentos da seção 6.2.1, *local 1*, *local 2* e *local 3* utilizam os mesmos valores para φ_0, φ_1, φ_2, V_{max} e S. A diferença está no tipo de vizinhança, e o aqui empregado é uma generalização da vizinhança em anel descrita na seção 2.4.1.

Imaginemos as partículas do enxame dispostas em círculo. Seja $N = |\overrightarrow{neigh}|$. Na vizinhança em anel, $N = 3$, o que significa que cada partícula é vizinha de si própria e também das duas partículas adjacentes a ela no círculo. Para viabilizar casos onde N é par, convencionemos que, em caso de empate na proximidade da última partícula na composição de $\overrightarrow{neigh}$, a escolhida será a mais próxima no sentido anti-horário. A Figura 6.1 ilustra este tipo de vizinhança para $N \in \{3, 4\}$. Podemos interpretar a vizinhança global como um caso particular desta vizinhança, com $N = S$.

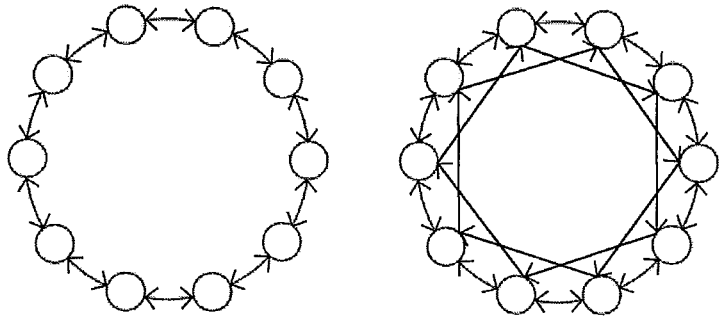

Figura 6.1: Exemplos de vizinhança em anel generalizada. À esquerda, $N = 3$; à direita, $N = 4$. Os laços do grafo não foram representados, mas estão presentes em cada partícula. Os arcos estão orientados no sentido do fluxo de informações.

Os valores para o tamanho da vizinhança em anel generalizada são escolhidos de forma que correspondam a, aproximadamente, 5% e 10% do tamanho do enxame. Além desses, o valor 3, que representa a vizinhança em anel tradicional, também é testado. Quando $S \in \{10, 40\}$, 5% e 10% de S diferem muito pouco da vizinhança em anel tradicional. Assim, os experimentos com $S \in \{10, 40\}$ contemplam apenas o valor 3 para N.

- *local 1* (364 experimentos)

- vizinhança em anel generalizada
- $N \in \{3\}$
- $\varphi_0 \in \{1\}$
- $\varphi_1 = \varphi_2 \in \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1, 5, 10\}$
- $S \in \{10, 40\}$
- $V_{max} \in \{1, 2, 3, 4, 5, 7, 8, 10, 11, 20, 30, 40, 60\}$

- *local 2* (546 experimentos)
  - vizinhança em anel generalizada
  - $N \in \{3, 9, 18\}$
  - $\varphi_0 \in \{1\}$
  - $\varphi_1 = \varphi_2 \in \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1, 5, 10\}$
  - $S \in \{180\}$
  - $V_{max} \in \{1, 2, 3, 4, 5, 7, 8, 10, 11, 20, 30, 40, 60\}$

- *local 3* (546 experimentos)
  - vizinhança em anel generalizada
  - $N \in \{3, 13, 25\}$
  - $\varphi_0 \in \{1\}$
  - $\varphi_1 = \varphi_2 \in \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1, 5, 10\}$
  - $S \in \{250\}$
  - $V_{max} \in \{1, 2, 3, 4, 5, 7, 8, 10, 11, 20, 30, 40, 60\}$

6.2.3 Grupo *global 2*

Neste grupo de experimentos testamos o efeito da aplicação do coeficiente de inércia no PSO binário. É possível verificar, comparando os resultados deste grupo com os de *global 1* se $\varphi_0 \neq 1$ é benéfico ou não para a eficiência do PSO, neste estudo de caso.

Como o número de experimentos executados seria muito grande, reduzimos os domínios os parâmetros φ_1, φ_2 e V_{max}, eliminando os valores que apresentam os piores resultados em *global 1*. Com isto, há uma economia de 888 experimentos.

- *global 2* (1296 experimentos)
  - vizinhança global
  - $\varphi_0 \in \{0.95, 1.05, 2\}$
  - $\varphi_1 = \varphi_2 \in \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1\}$
  - $S \in \{10, 40, 180, 250\}$
  - $V_{max} \in \{5, 7, 8, 10, 11, 20, 30, 40, 60\}$

6.2.4 Grupo *global 3*

Até esta seção, todos os experimentos conduzidos utilizam $\varphi_1 = \varphi_2$. Embora esta seja a configuração mais utilizada na literatura, julgamos importante a experimentação no subespaço de parâmetros onde $\varphi_1 \neq \varphi_2$, em prol da completude dos experimentos.

Seja $\varphi = \varphi_1 + \varphi_2$. Fixando φ em $\{0.1, 0.25, 0.4, 0.55\}$, podemos variar um fator $r = \varphi_1/\varphi_2$. Os valores utilizados para r, que resultam nos valores de φ_1 e φ_2 a seguir são: 1.5, 2.3, 3.1, 3.9, 0.26, 0.32, 0.43 e 0.67.

- *global 3* (256 experimentos)
  - vizinhança global
  - $phi_{0.1} = \{(0.06; 0.04), (0.069; 0.03), (0.075; 0.024), (0.079; 0.02), (0.02; 0.079), (0.024; 0.075), (0.03; 0.069), (0.04; 0.059)\}$
  - $phi_{0.25} = \{(0.15; 0.1), (0.174; 0.075), (0.189; 0.06), (0.198; 0.051), (0.051; 0.198), (0.06; 0.189), (0.075; 0.174), (0.1; 0.149)\}$
  - $phi_{0.4} = \{(0.24; 0.16), (0.278; 0.121), (0.302; 0.097), (0.318; 0.081), (0.082; 0.317), (0.096; 0.303), (0.12; 0.279), (0.16; 0.239)\}$
  - $phi_{0.55} = \{(0.33; 0.22), (0.383; 0.166), (0.415; 0.134), (0.437; 0.112), (0.113; 0.436), (0.133; 0.416), (0.165; 0.384), (0.22; 0.329)\}$

- $\varphi_0 \in \{1\}$
- $(\varphi_1; \varphi_2) \in (phi_{0.1} \cup phi_{0.25} \cup phi_{0.4} \cup phi_{0.55})$
- $S \in \{180, 250\}$
- $V_{max} \in \{7, 8, 10, 11\}$

6.2.5 Grupos *wglobal 1* e *wglobal 2*

Acreditamos que um dos fatores que dificultam a otimização da função de energia gerada pelo SATyrus é o uso dos somadores binários. Na modelagem descrita no Capítulo 5, as restrições relativas a esses somadores estão todas associadas a um mesmo nível de penalidade, o mesmo das demais restrições de integridade. Os grupos *wglobal 1* e *wglobal 2* não introduzem novidade alguma nos domínios dos parâmetros. Ao invés disso, a própria função de energia em cada um desses grupos é que difere da utilizada nos outros.

Para ambos os grupos, dividimos as restrições de integridade associadas aos somadores em cinco partes, cada uma abrangendo três bits adjacentes. A cada parte foi atribuído um grupo de penalidades, de ordem diretamente proporcional a significância dos bits cuja parte é responsável. Ou seja, as restrições relativas aos bits 15, 14 e 13 passam a ter uma penalidade maior do que as relativas aos bits 12, 11 e 10 que, por sua vez, têm uma penalidade maior do que as relativas aos bits 9, 8 e 7 e assim por diante.

Feito isso, resta uma decisão a ser tomada: as restrições associadas aos somadores devem ser mais ou menos importantes — receber uma penalidade maior ou menor — do que as demais? O resolvedor se comporta melhor penalizando mais severamente absurdos matemáticos ($1 + 0 = 0$) ou físicos (energia elétrica sendo gerada onde não existem usinas)? Dada a dificuldade da decisão, as duas possibilidades são testadas. O grupo *wglobal 1* trata os somadores como as restrições mais importantes da modelagem, ao passo que o grupo *wglobal 2* trata o caso análogo.

- *wglobal 1* (252 experimentos)
  - vizinhança global
  - somadores binários com penalidades diferenciadas

- penalidades associadas aos somadores binários maiores do que as associadas às demais restrições de integridade
- $\varphi_0 \in \{1\}$
- $\varphi_1 = \varphi_2 \in \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1, 5, 10\}$
- $S \in \{180, 250\}$
- $V_{max} \in \{4, 5, 7, 8, 10, 11, 20, 30, 40\}$

- *wglobal 2* (252 experimentos)
  - vizinhança global
  - somadores binários com penalidades diferenciadas
  - penalidades associadas aos somadores binários menores do que as associadas às demais restrições de integridade
  - $\varphi_0 \in \{1\}$
  - $\varphi_1 = \varphi_2 \in \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85, 0.99, 1.5, 1.99, 2, 3.9, 4.1, 5, 10\}$
  - $S \in \{180, 250\}$
  - $V_{max} \in \{4, 5, 7, 8, 10, 11, 20, 30, 40\}$

6.3 Apresentação dos resultados

Nesta seção apresentamos os resultados dos experimentos realizados. Excetuando-se quando especificado, para todos os gráficos a seguir, o eixo x (abcissas) sempre representa o valor dos parâmetros φ_1 e φ_2, ao passo que o eixo y (ordenadas) representa a energia média (abreviada como Em).

Pontos mais à esquerda representam experimentos cujo valor dos parâmetros φ_1 e φ_2 é muito pequeno; pontos mais à direita representam experimentos nos quais o valores de φ_1 e φ_2 beiram o limite superior do considerado razoável. Experimentos com valores diversos para V_{max} são colocados em linhas diferentes de forma que, ao observar vários pontos com a mesma coordenada x, seja fácil ver, para a dada configuração, que valor para V_{max} obtém o melhor resultado.

Como o formato das curvas não varia muito em função do parâmetro S, para não prejudicar a clareza do texto, apresentamos aqui somente os gráficos relacionados a um valor de S (250). Os demais podem ser encontrados no apêndice B.

6.3.1 Grupo *global 1*

A Figura 6.2 mostra a energia média alcançada em cada experimento em função das diferentes configurações dos parâmetros $\varphi_1 = \varphi_2$, enquanto a Figura 6.3 mostra, para os mesmos experimentos, o número médio de avaliações da função objetivo necessárias para atingir a convergência.

Para alguns valores de V_{max}, particularmente os maiores que 4, a energia média não varia muito dada a mesma configuração dos demais parâmetros. Com o objetivo de proporcionar uma melhor visualização, esses dados são reapresentados com uma escala diferente na Figura 6.4.

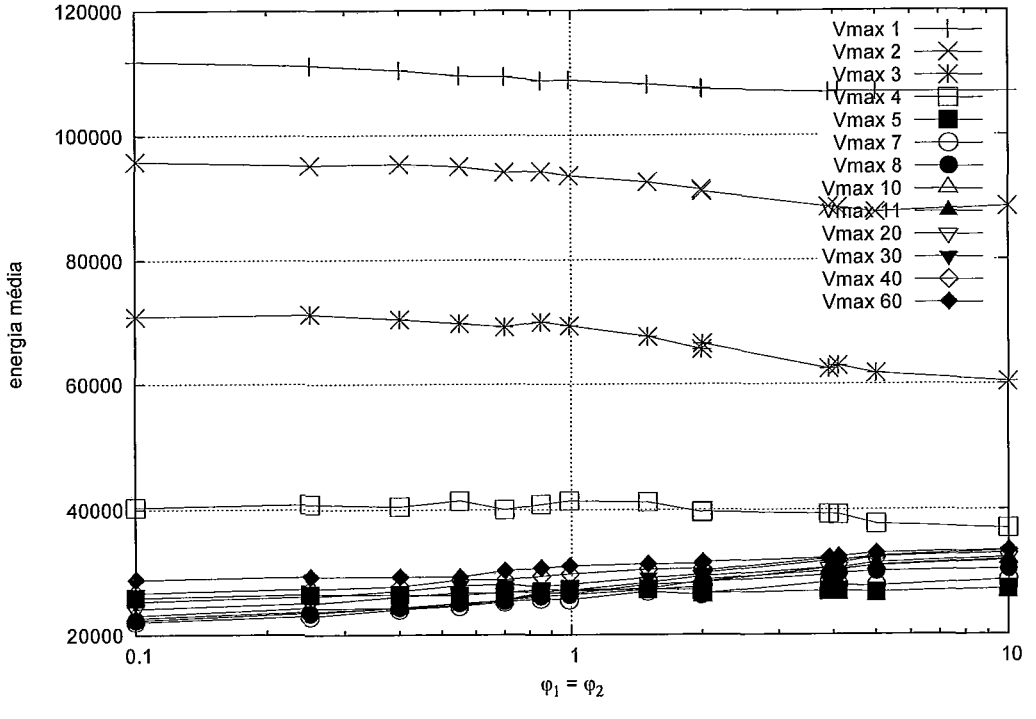

Figura 6.2: Resultados: grupo *global 1*, $S = 250$

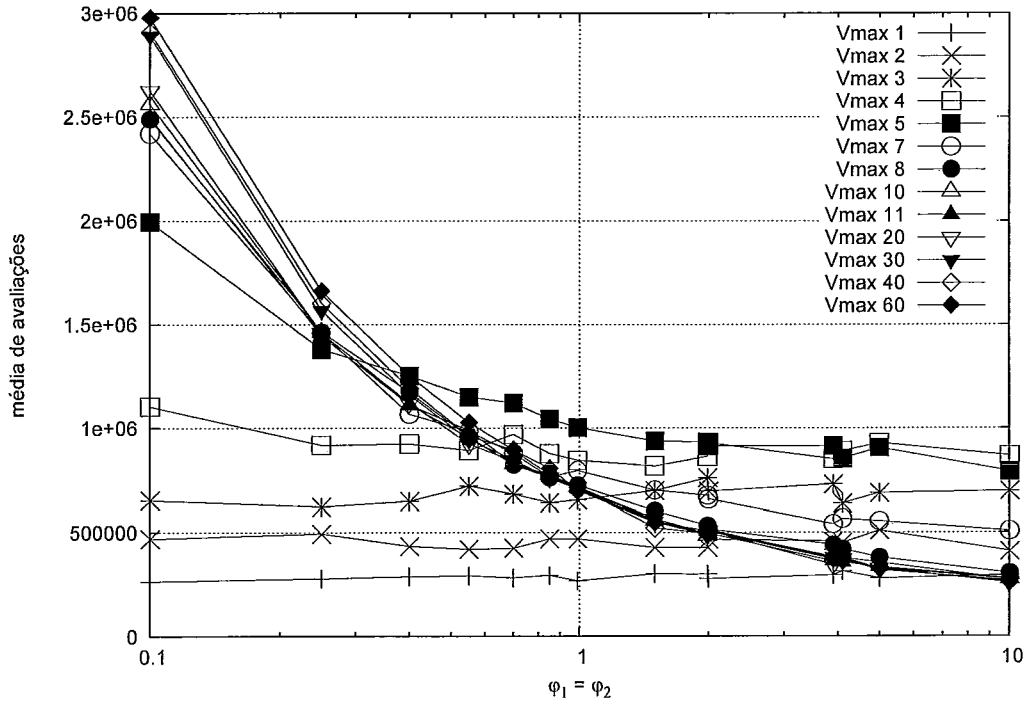

Figura 6.3: Resultados: grupo *global 1*, $S = 250$

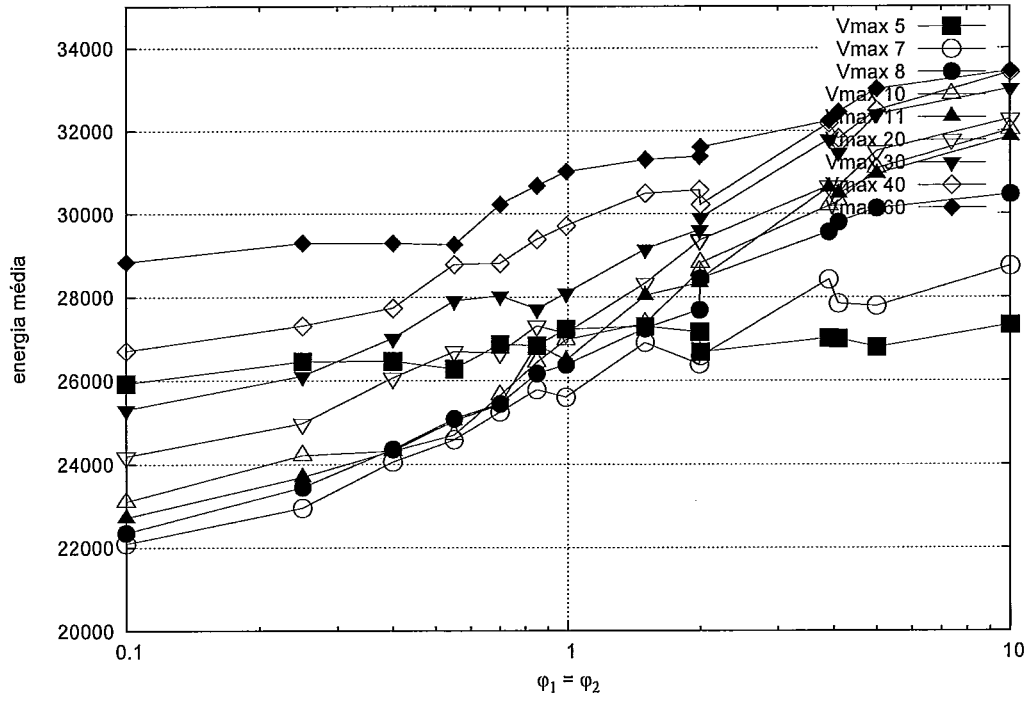

Figura 6.4: Resultados: grupo *global 1*, $S = 250$, $V_{max} \geq 5$

6.3.2 Grupo *local 3*

Apresentamos aqui os resultados dos experimentos realizados utilizando o modelo de vizinhança em anel generalizada. As Figuras 6.5 e 6.6 contêm os resultados com um tamanho de enxame igual a 250, utilizando 3 como tamanho da vizinhança. Resultados com vizinhanças de tamanho 13 (5% de S) são apresentados nas Figuras 6.7 e 6.8; com vizinhanças de tamanho 25 (10% de S), nas Figuras 6.9 e 6.10.

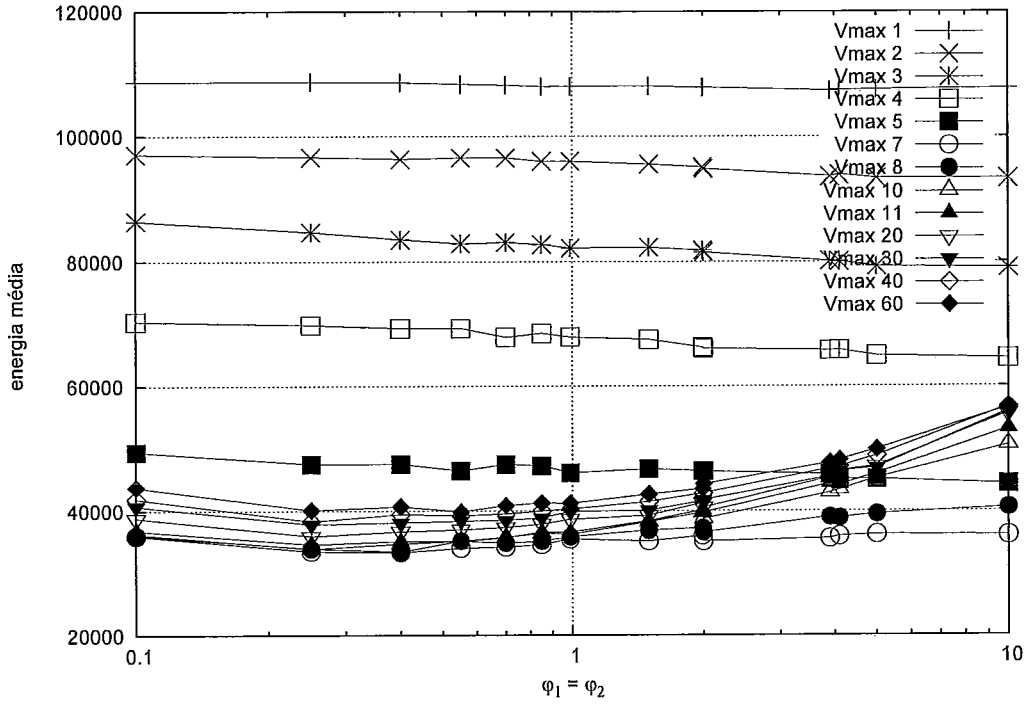

Figura 6.5: Resultados: grupo *local 3*, $S = 250$, $N = 3$

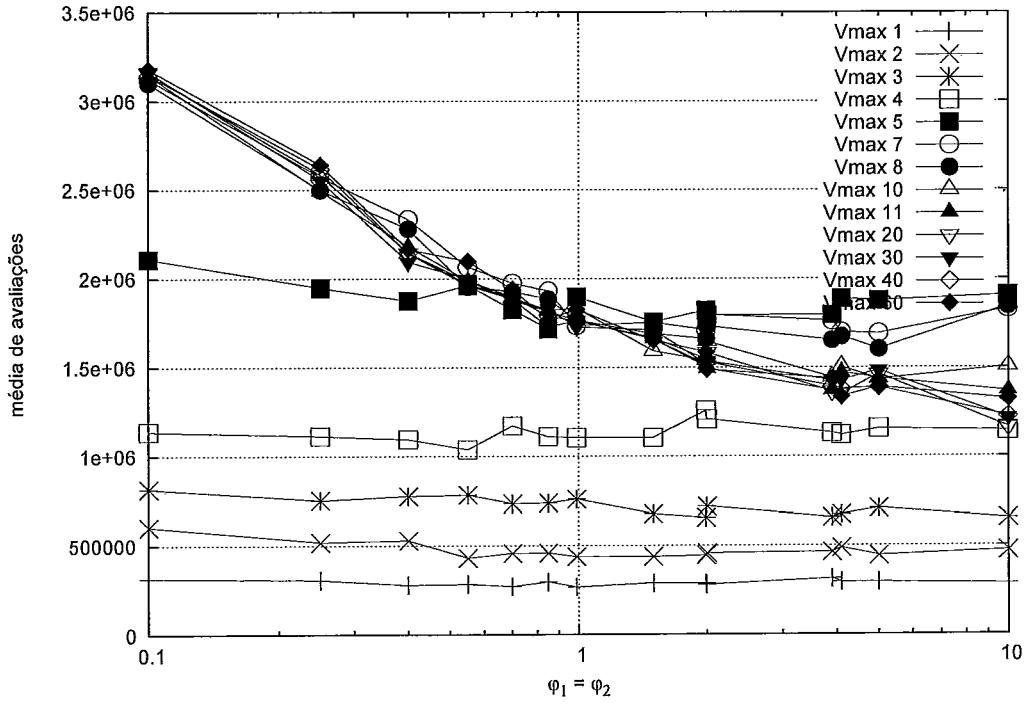

Figura 6.6: Resultados: grupo *local 3*, $S = 250$, $N = 3$

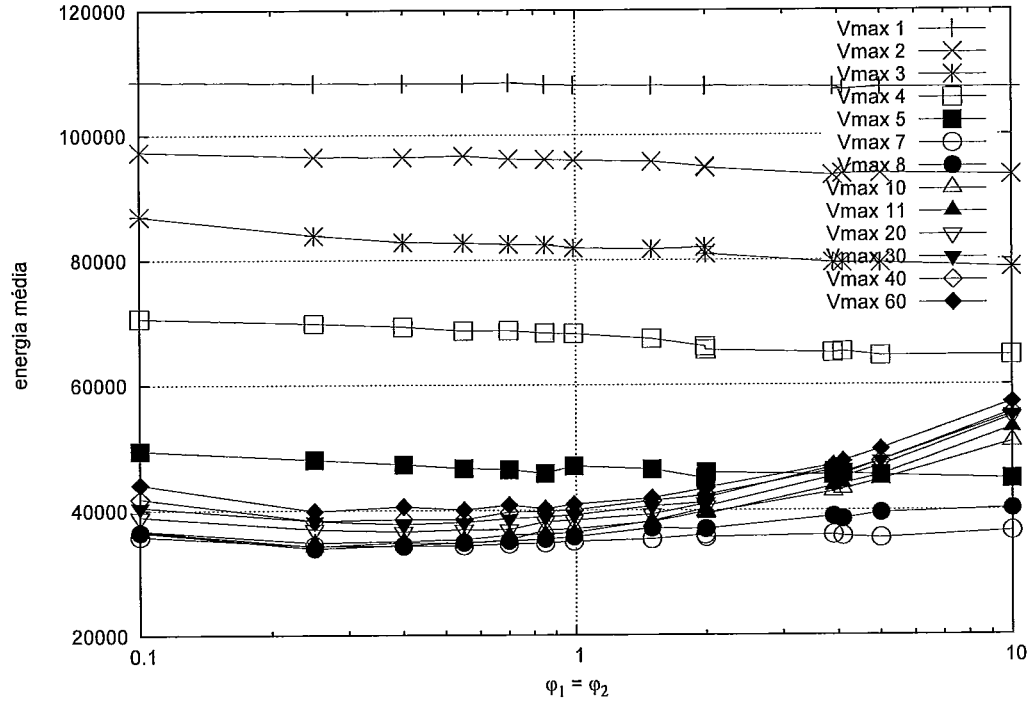

Figura 6.7: Resultados: grupo *local 3*, $S = 250$, $N = 13$

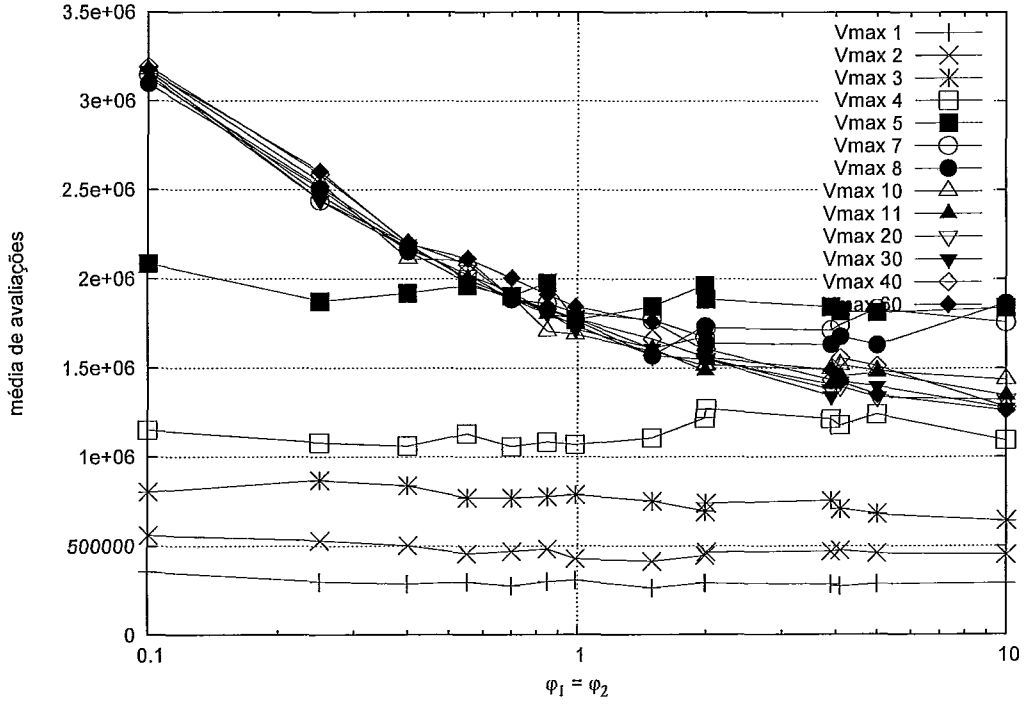

Figura 6.8: Resultados: grupo *local 3*, $S = 250$, $N = 13$

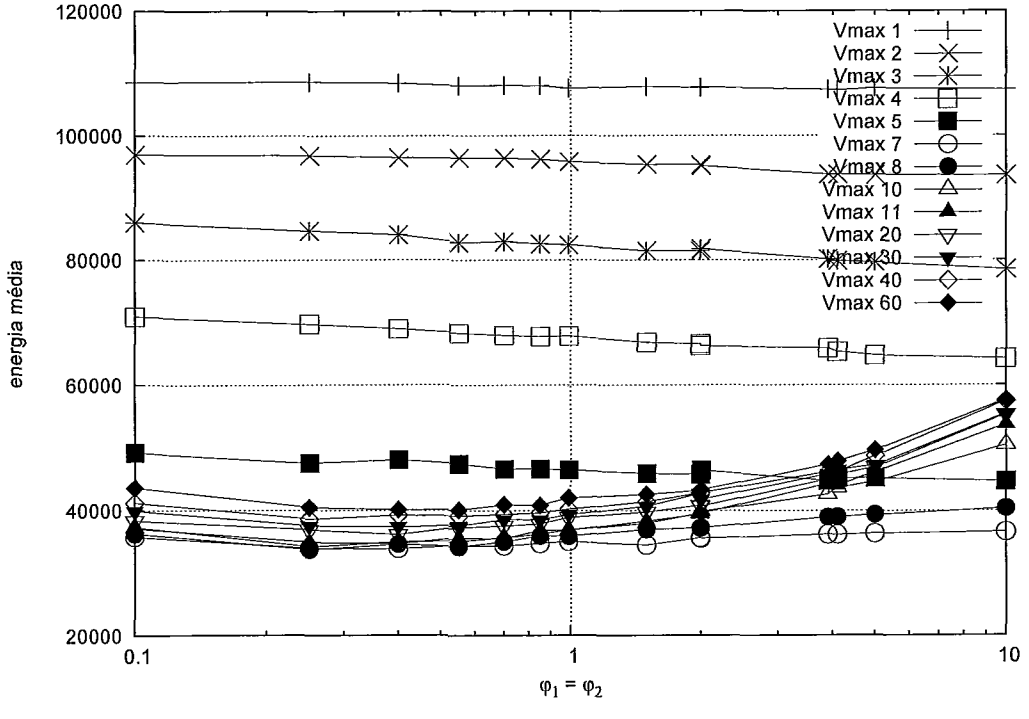

Figura 6.9: Resultados: grupo *local 3*, $S = 250$, $N = 25$

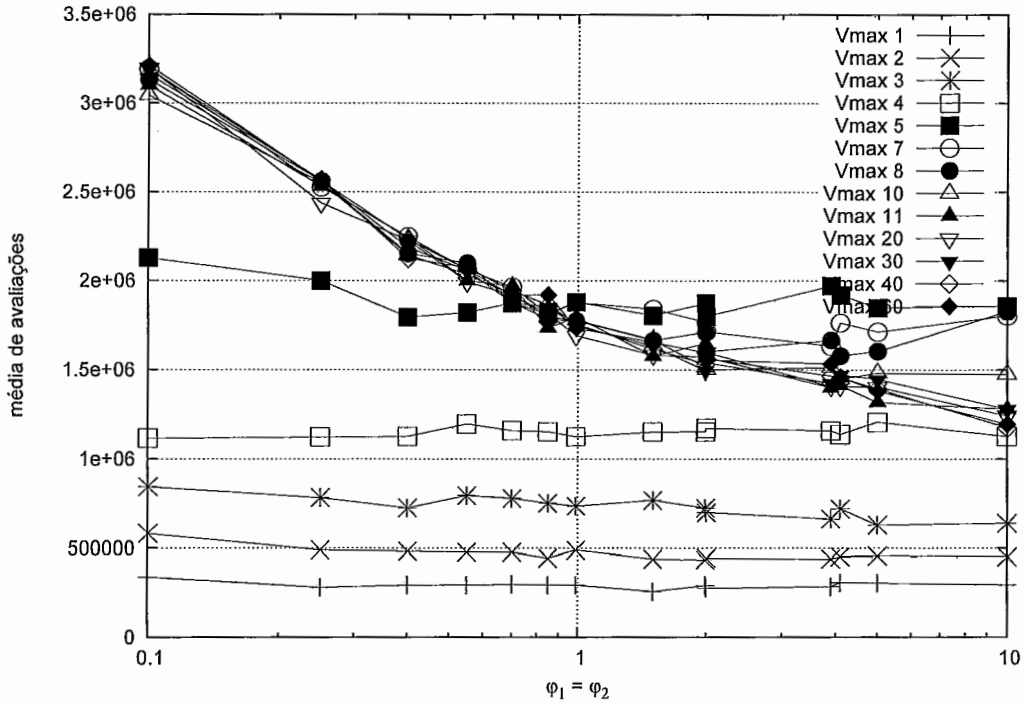

Figura 6.10: Resultados: grupo *local 3*, $S = 250$, $N = 25$

6.3.3 Grupo *global 2*

Nesta seção são apresentados os resultados dos experimentos conduzidos com o coeficiente de inércia φ_0 diferente de 1. As Figuras 6.11, 6.13 e 6.15 apresentam a energia média obtida com valores de φ_0 iguais a 0.95, 1.05 e 2, respectivamente, enquanto as Figuras 6.12, 6.14 e 6.16 apresentam o número médio de avaliações da função objetivo, para os mesmos valores de φ_0.

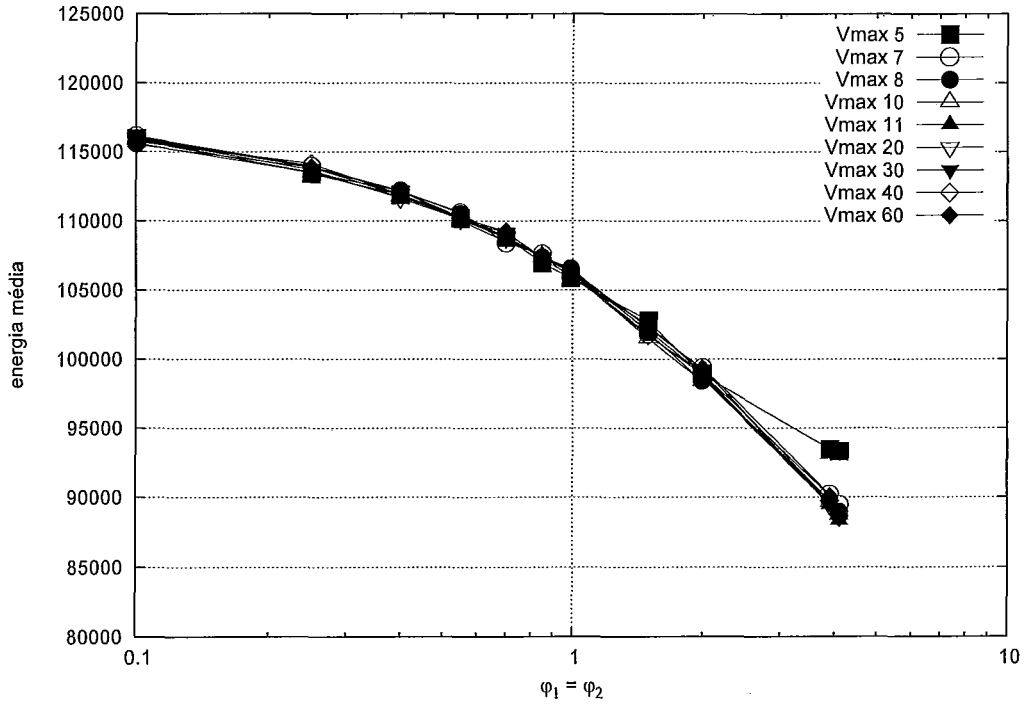

Figura 6.11: Resultados: grupo *global 2*, $S = 250$, $\varphi_0 = 0.95$

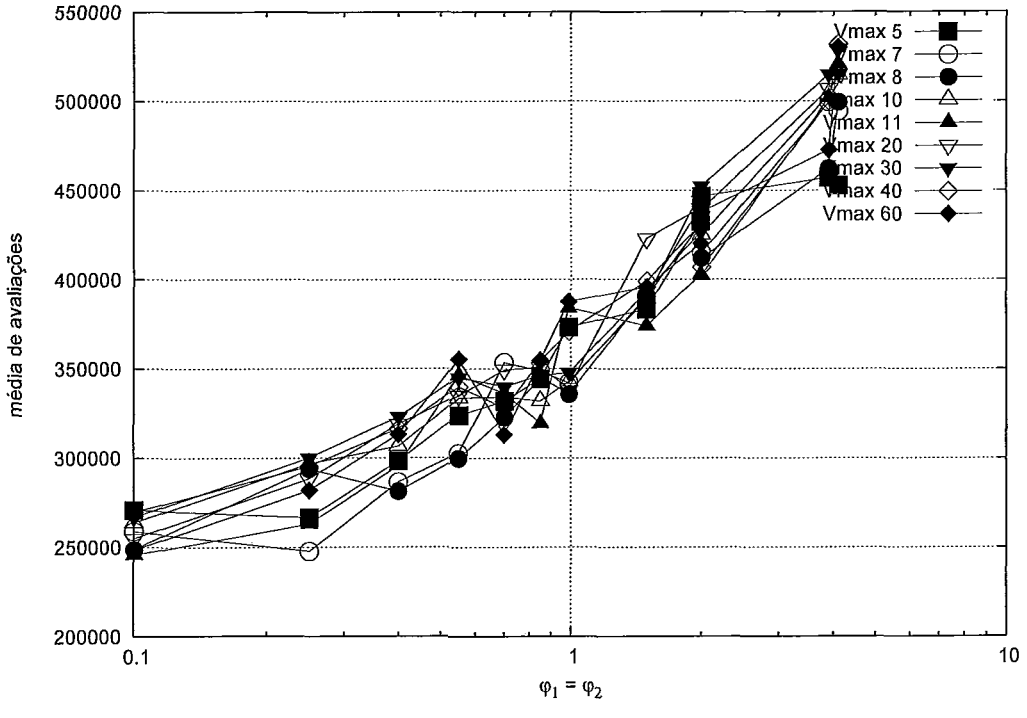

Figura 6.12: Resultados: grupo *global 2*, $S = 250$, $\varphi_0 = 0.95$

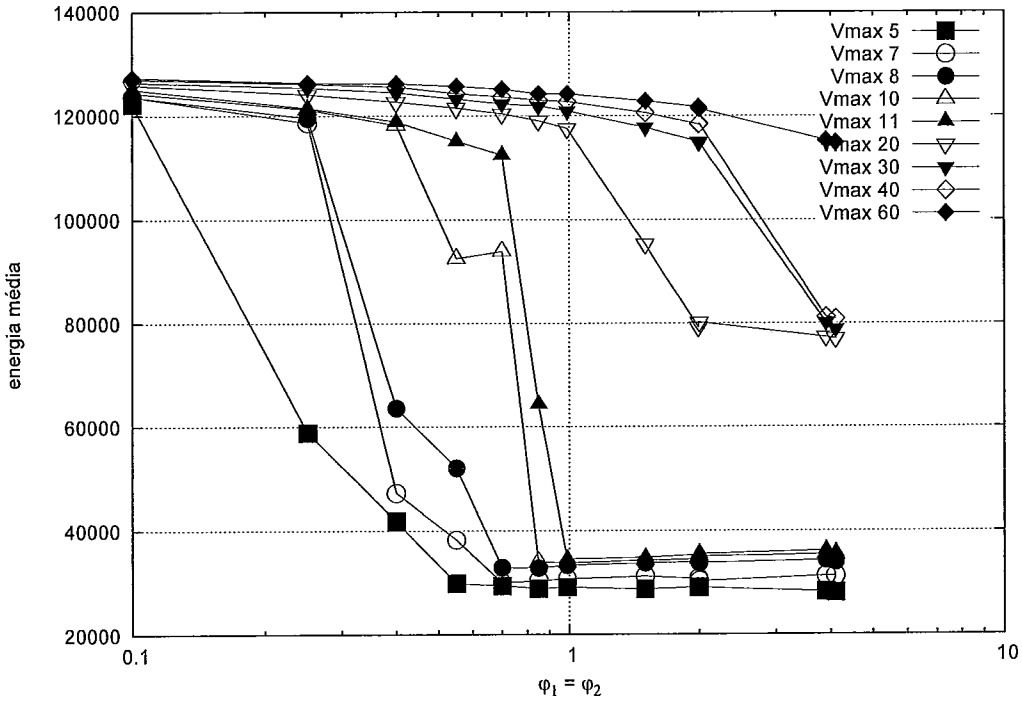

Figura 6.13: Resultados: grupo *global 2*, $S = 250$, $\varphi_0 = 1.05$

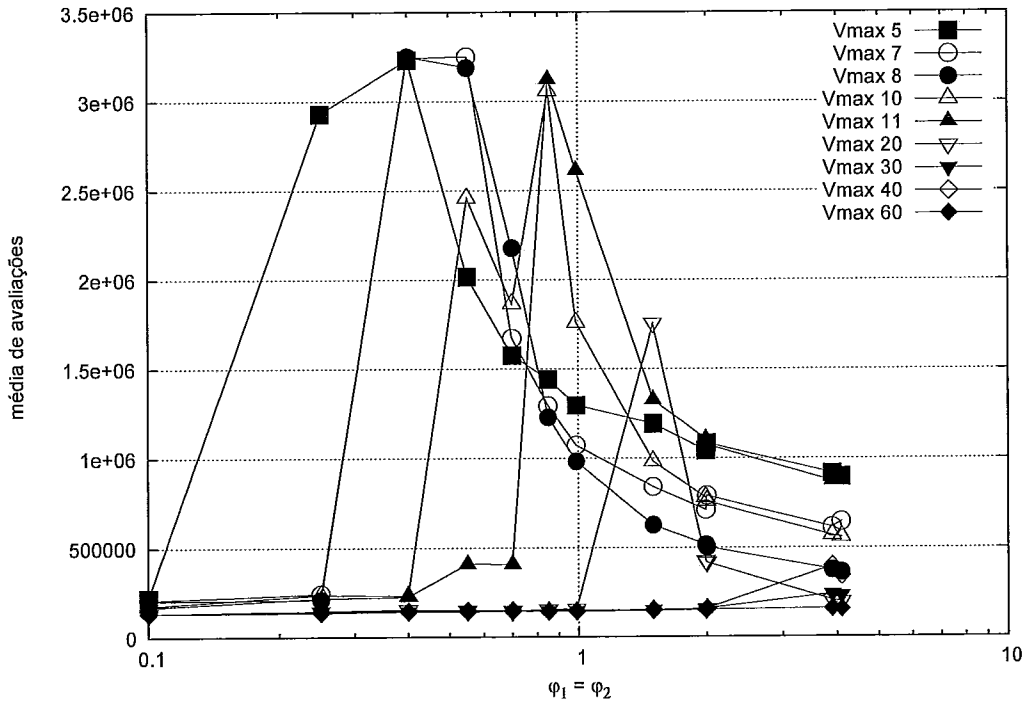

Figura 6.14: Resultados: grupo *global 2*, $S = 250$, $\varphi_0 = 1.05$

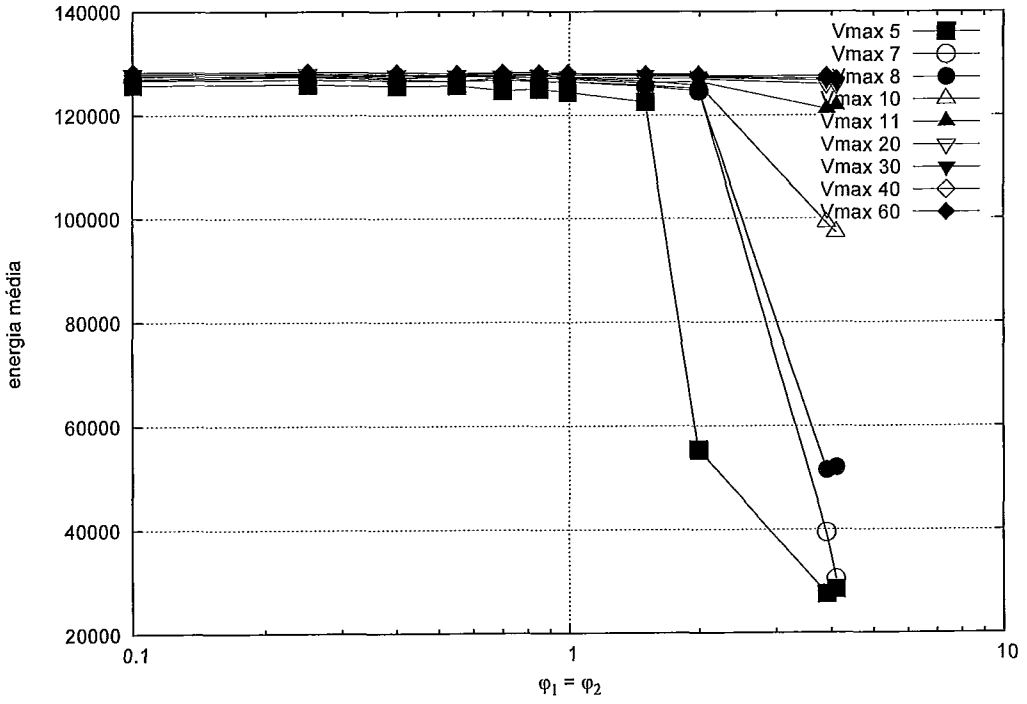

Figura 6.15: Resultados: grupo *global 2*, $S = 250$, $\varphi_0 = 2$

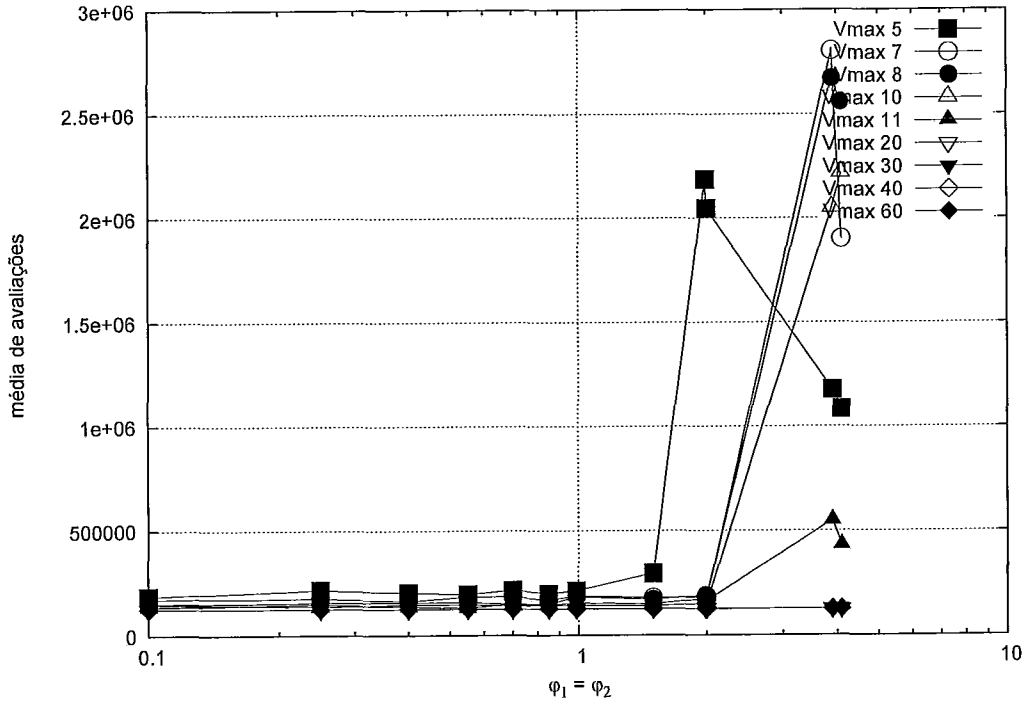

Figura 6.16: Resultados: grupo *global 2*, $S = 250$, $\varphi_0 = 2$

6.3.4 Grupo *global 3*

Os resultados com os parâmetros φ_1 e φ_2 desbalanceados são apresentados nas Figuras 6.17 e 6.18. Nelas, os eixos da base representam os parâmetros φ_1 e φ_2 já que, neste grupo de experimentos, seus valores são diferentes um do outro. O eixo z (cotas) representa a energia média alcançada em cada experimento ou a quantidade média de avaliações da função objetivo. As linhas extras presentes nestes gráficos traçam os caminhos da energia média crescente; há uma sequência delas para cada valor de V_{max}.

Nas Figuras 6.19 e 6.20 os mesmos dados são reapresentados. A diferença é que os parâmetros φ_1 e φ_2 são combinados no eixo x (que passa a representar $\varphi = \varphi_1 + \varphi_2$), proporcionando uma visão sobre o efeito desta soma na energia média e no número médio de avaliações.

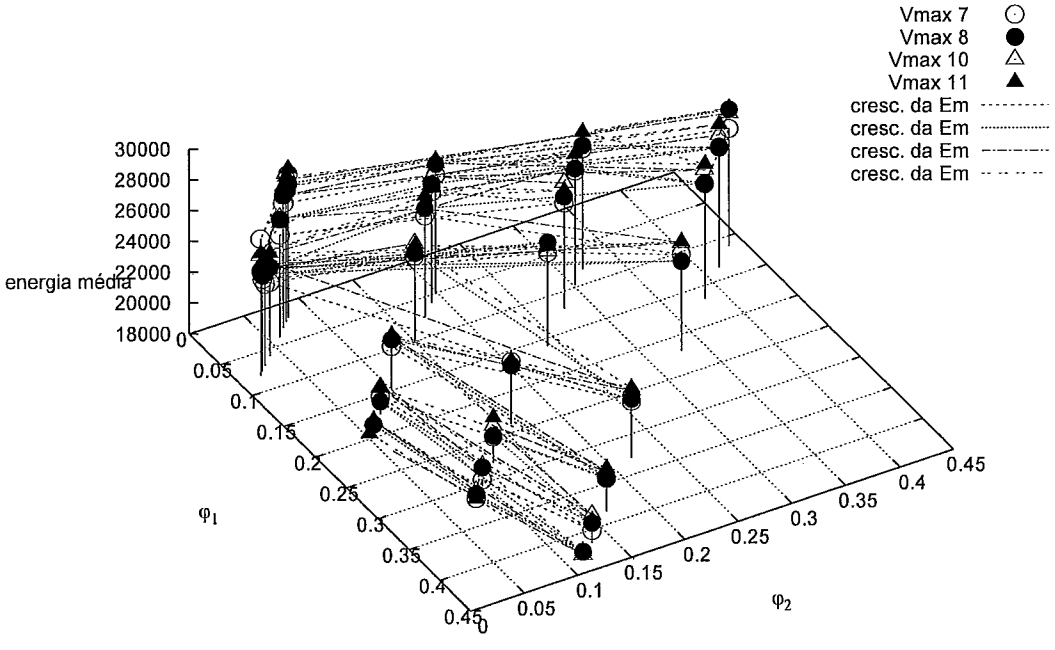

Figura 6.17: Resultados: grupo *global 3*, $S = 250$

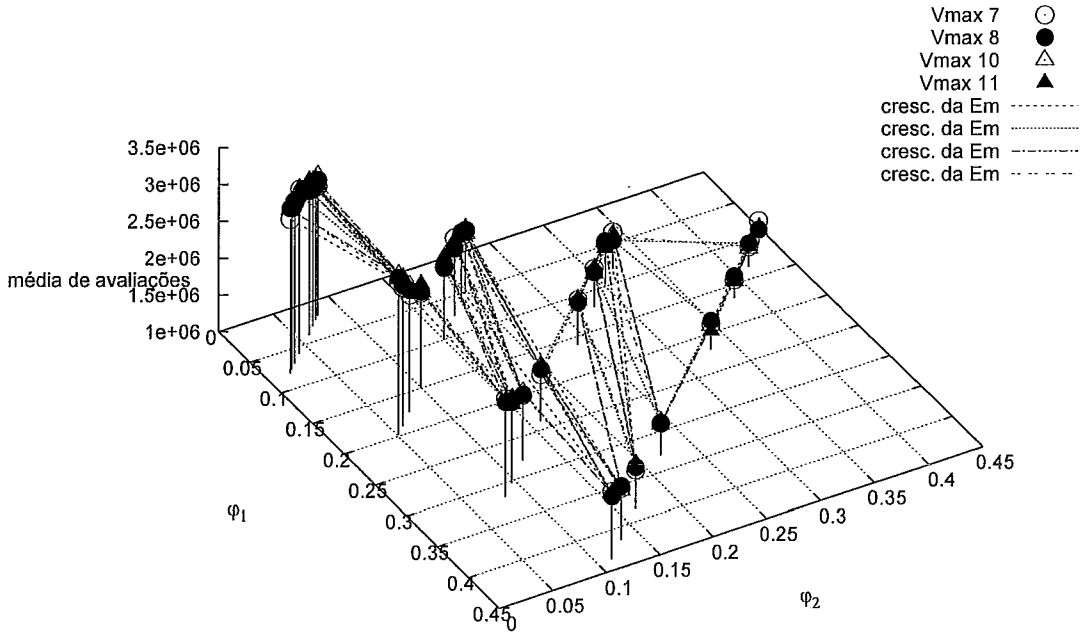

Figura 6.18: Resultados: grupo *global 3*, $S = 250$

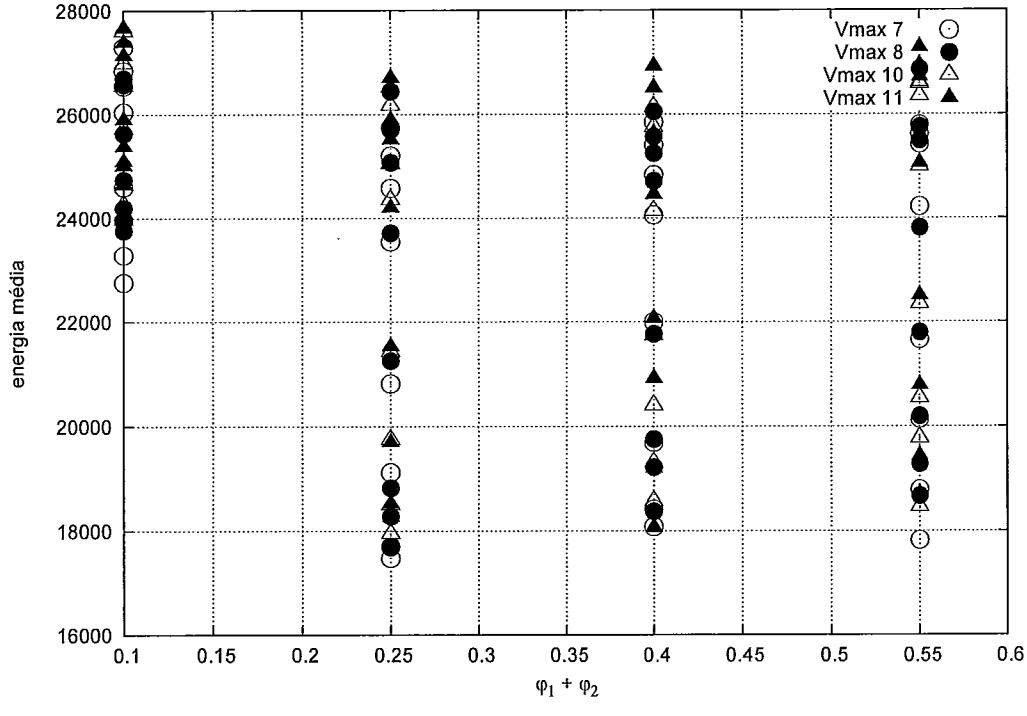

Figura 6.19: Resultados: grupo *global 3*, $S = 250$, agrupados por φ

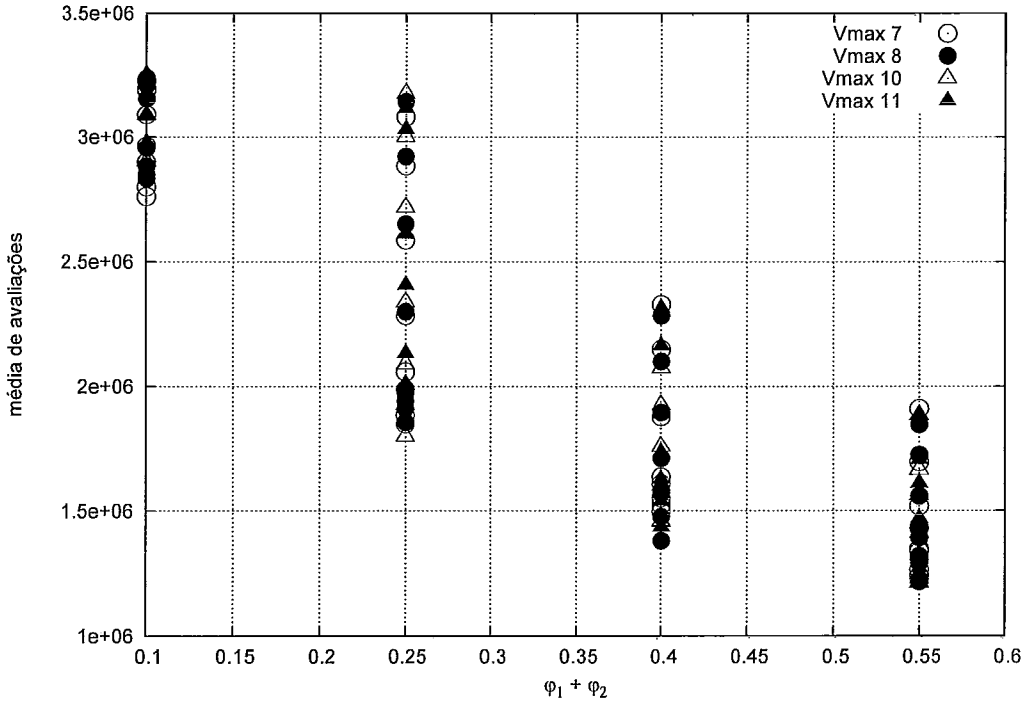

Figura 6.20: Resultados: grupo *global 3*, $S = 250$, agrupados por φ

6.3.5 Grupos *wglobal 1* e *wglobal 2*

Nesta seção estão apresentados os resultados dos experimentos que ponderam as penalidades das restrições de integridade associadas aos somadores binários pela significância dos bits. As Figuras 6.21 e 6.22 tratam do caso em que o somador é considerado mais importante do que as demais restrições; e as Figuras 6.23 e 6.24, do caso análogo.

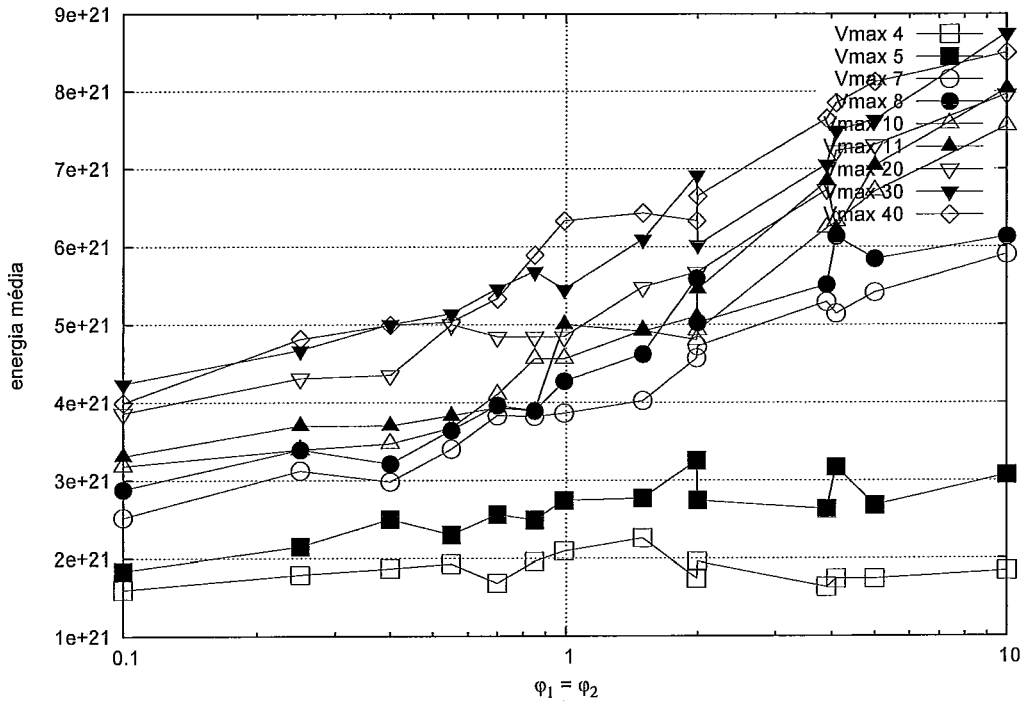

Figura 6.21: Resultados: grupo *wglobal 1*, $S = 250$

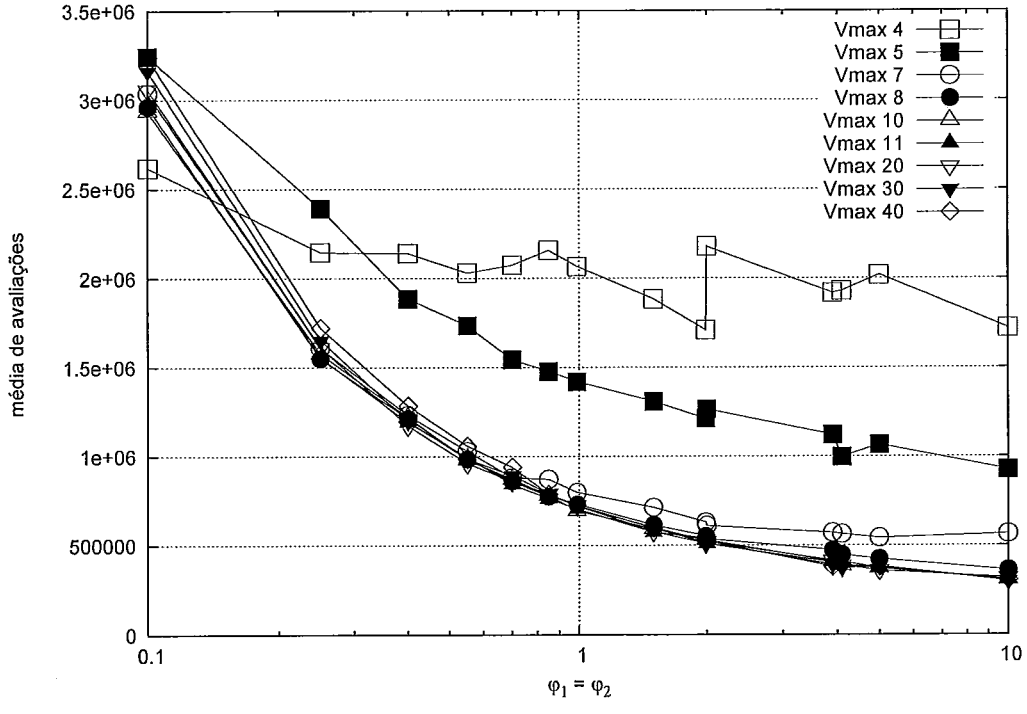

Figura 6.22: Resultados: grupo *wglobal 1*, $S = 250$

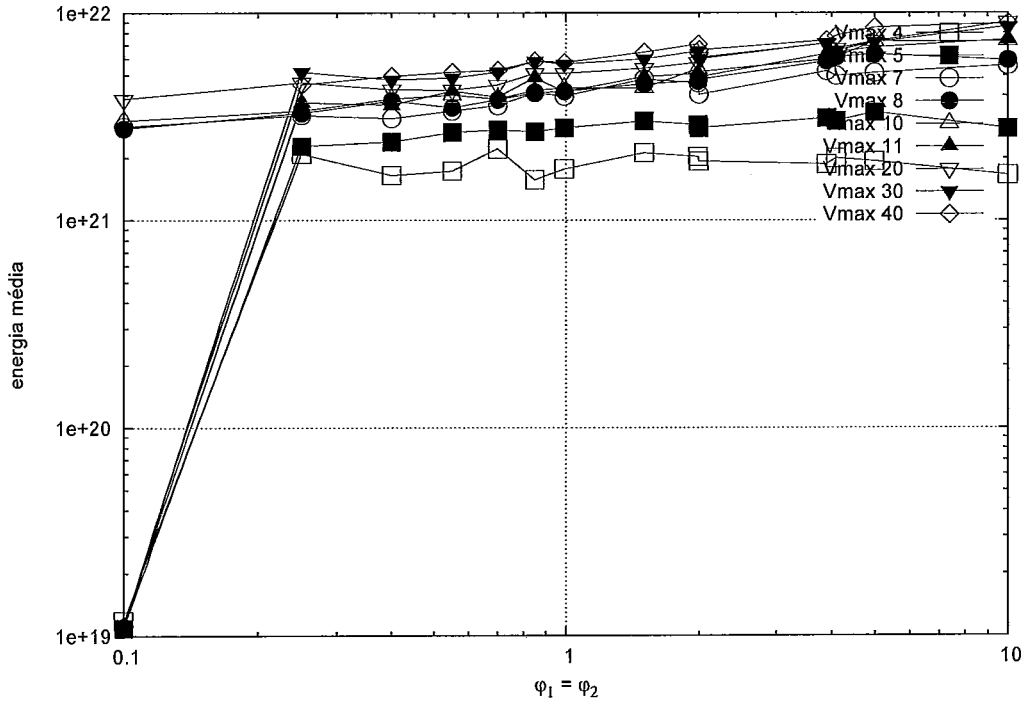

Figura 6.23: Resultados: grupo *wglobal 2*, $S = 250$

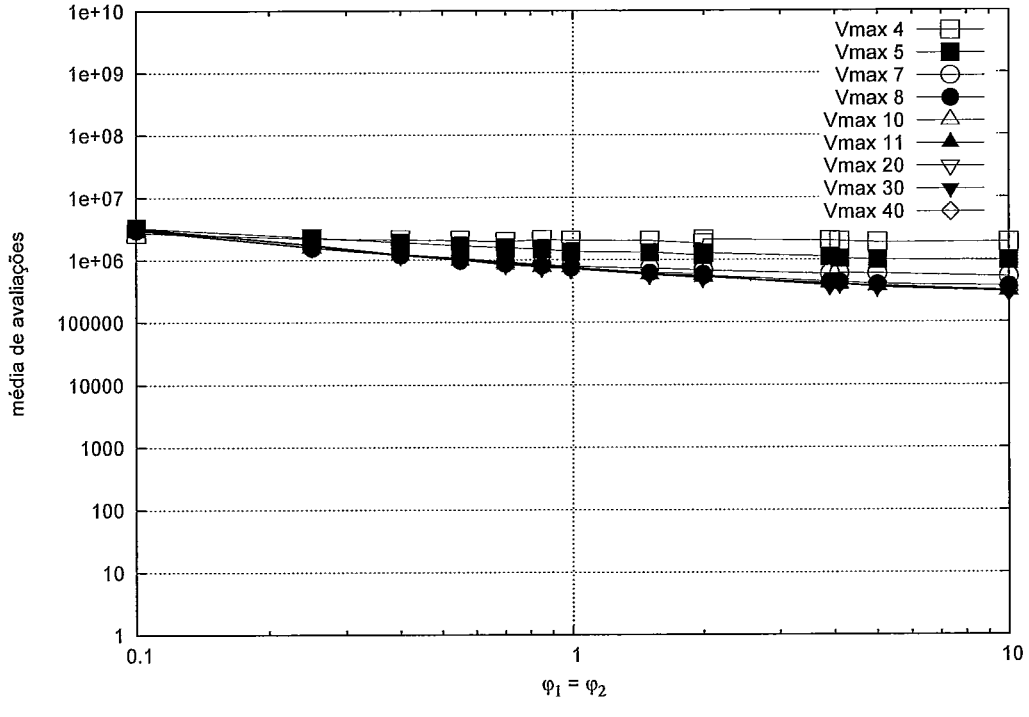

Figura 6.24: Resultados: grupo *wglobal 2*, $S = 250$

6.4 Apreciação dos resultados

Feita a apresentação os resultados, agora é possível analisá-los com o intuito de tirar conclusões sobre a eficácia de determinadas combinações de parâmetros, quando contrapostas a outras. Antes disso, entretanto, é necessário que se defina o sucesso dos experimentos em termos gerais, ou seja, quão perto o resolvedor chega do ótimo global da função a ser minimizada.

6.4.1 Avaliação do sucesso

De todos os experimentos conduzidos, a execução que produziu a solução com energia mais baixa o fez após realizar 3250000 avaliações da função objetivo, alcançando o valor para a energia de 2542. Este valor está bastante longe do ótimo global, que está associado a uma energia de valor 53. Embora a diferença na energia seja significativa, devemos ressaltar que, na modelagem utilizada, exceto para os grupos de experimentos *wglobal 1* e *wglobal 2*, cada restrição de integridade violada implica em um acréscimo de 245 na energia da solução. Na maioria dos casos, um único bit valorado erroneamente é suficiente para que haja uma violação. A energia de 2542 está associada à violação de, no máximo, 10 restrições de integridade. Se lembrarmos que a modelagem contempla 8259 restrições, sendo a grande maioria delas de integridade, não podemos ver este resultado como desencorajador, apesar de reconhecermos não ser o ideal.

Na Tabela 6.1 estão lado a lado esses dois pontos. À esquerda o ponto ótimo (calculado em [21] utilizando o pacote comercial de otimização combinatória mista CPLEX [7]); à direita o melhor ponto encontrado nos experimentos. Os números em itálico representam valores que violam as capacidades das respectivas usinas. Em nenhum dos pontos houve violação nas restrições de demanda ou construção.

6.4.2 Avaliação das configurações dos parâmetros

Prosseguimos então, com a avaliação dss configurações dos parâmetros nos experimentos realizados. Na seção 6.4.2.1 o efeito da variação do parâmetro S é comentada. Nas seções 6.4.2.2 e 6.4.2.3 são mostradas as análises a respeito dos parâmetros φ_0

Tabela 6.1: Melhores pontos encontrados

		Estágio					
		1	2	3	1	2	3
Usina	1	3285	3285	3285	1043	*8665*	1803
	2	521	1858	3267	2442	*14649*	*3373*
	3	574	1427	2208	1026	2145	*8354*
	4	0	0	0	*11608*	*7493*	*14656*
	5	0	0	0	15185	9561	13443
		ponto ótimo (energia=53)			melhor experimento (energia=2542)		

e V_{max}, respectivamente. Comentários sobre a variação de φ_1 e φ_2 podem ser encontradas na seção 6.4.2.4, enquanto a seção 6.4.2.5 é reservada para variações do parâmetro N.

Nas avaliações dos resultados, referenciamos os parâmetros φ_1 e φ_2 por sua soma φ, exceto quando estes assumem valores diferentes um do outro.

6.4.2.1 O parâmetro S

Como pode ser visto nos gráficos das subseções da Seção 6.3 e no apêndice B, através da comparação de gráficos do mesmo grupo que empreguem valores diferentes para S, o tamanho do enxame influencia o comportamento do PSO. Há duas consequências notáveis da variação desde parâmetro.

A primeira é que a energia encontrada cai à medida em que o tamanho do enxame aumenta (mantendo todos os outros parâmetros constantes). Este é o comportamento esperado pois, aumentando o número de partículas, aumenta também o número de pontos no espaço de busca explorados. Cada partícula, ao mover-se de sua posição em um dado estado até o ótimo atual, explora uma série de outros pontos, que podem revelar valores de energia melhores do que o ótimo atual. Quanto mais partículas há no enxame, maior é a chance de que esses pontos sejam descobertos.

A segunda consequência é uma variação muito pronunciada no número de avaliações da função até que seja alcançada a convergência. Estas constituem o cálculo mais demorado realizado pelo resolvedor. A cada iteração de cada partícula, ocorre

uma avaliação. Dada a elevada quantidade destas operações que ocorrem durante cada experimento, é fácil perceber que o número de avaliações da função objetivo é o fator que mais influencia o tempo de execução do algoritmo.

Observemos que enquanto a variação na energia média alcançada em função de S não é tão grande, dificultando sua percepção através da observação dos gráficos, a variação no número médio de avaliações é bem maior, abrangendo algumas ordens de grandeza. A Tabela 6.2 contém um exemplo deste comportamento observado nos experimentos com a vizinhança em anel (grupos *local 1*, *local 2* e *local 3*).

Tabela 6.2: Exemplo do efeito da variação de S

S	V_{max}	N	φ	Energia média	Nº de avaliações médio
10	7	3	0.5	37710.44	103999.2
40	7	3	0.5	35883.46	403452.0
180	7	3	0.5	33919.16	1799899.2
250	7	3	0.5	33458.92	2571500.0

Em alguns grupos as variações nas médias de energia do número de avaliações são menores do que na Tabela 6.2; em outros, são maiores. Apesar disso, o comportamento do PSO em função de S mostrou-se consistente, o que sugere uma linha de ação recomendável para futuras explorações no espaço de parâmetros: realizar uma primeira bateria de experimentos utilizando apenas enxames pequenos. Provavelmente, as configurações de parâmetros que obtiverem os melhores resultados nesses experimentos também serão as que destacar-se-ão quando experimentos com enxames maiores forem conduzidos. Dependendo da complexidade da função objetivo e da quantidade de combinações de parâmetros a ser testada, esta técnica pode reduzir muito o tempo necessário para a condução dos experimentos.

6.4.2.2 O parâmetro φ_0

É sabido que a aplicação do coeficiente de inércia em PSOs contínuos acelera a convergência. Conforme exposto na seção 2.4.6.1, o efeito de φ_0 no PSO binário pode ser diferente.

Os experimentos com $\varphi_0 = 0.95$ (seção 6.3.3) alcançam resultados muito inferiores aos em que $\varphi_0 = 1$. O motivo para este fato parece ser que, com cada componente

do vetor de velocidades tendendo para 0, a evolução do algoritmo fica mais próxima de uma busca randômica pelo espaço de estados. Como este possui um número muito mais elevado de pontos associados a altos valores de energia do que a baixos, o resolvedor rapidamente cessa de melhorar o ponto ótimo atual, satisfazendo ao critério de convergência sem realizar uma busca minuciosa.

Quanto aos experimentos onde $\varphi_0 > 1$ (também na seção 6.3.3), que embutem no resolvedor uma tendência para a aceleração da convergência, com algumas configurações dos outros parâmetros é possível alcançar um desempenho próximo, porém marginalmente inferior aos experimentos com $\varphi_0 = 1$. Aparentemente, a tendência de convergência imposta pelo parâmetro $\varphi_0 > 1$ é compensada por algumas configurações de φ e V_{max}. Para reforçar esta teoria vemos que, com o aumento de φ_0 de 1.05 para 2, menos combinações de φ e V_{max} resultam em resultados comparáveis aos de $\varphi_0 = 1$, sugerindo que fica mais difícil para a influência de φ e V_{max} contrabalançar a tendência de convergência associada a φ_0.

Na Tabela 6.3 vemos um exemplo do comportamento do resolvedor em função da variação de φ_0.

Tabela 6.3: Exemplo do efeito da variação de φ_0

S	V_{max}	φ_0	φ	Energia média	Nº de avaliações médio
40	5	0.95	1.98	108700.80	61156.0
40	5	1	1.98	30491.56	195321.6
40	5	1.05	1.98	32339.22	249964.8
40	5	2	1.98	127117.22	35212.0

6.4.2.3 O parâmetro V_{max}

O efeito da velocidade máxima das partículas não é tão decisivo quanto o do tamanho do enxame. Nossos experimentos mostram que, dependendo de outros parâmetros (especialmente de φ e do modelo de vizinhança utilizado), o melhor valor para V_{max} pode variar.

Incrementando a partir de 1 o valor de V_{max}, observamos que o desempenho em termos de energia média passa por uma fase transiente, melhorando progressivamente com o aumento do parâmetro, antes de estabilizar. Após esse ponto —

$V_{max} = 5$ em grande parte dos experimentos — quando continuamos a incrementar o valor de V_{max}, a energia média volta a subir, embora muito mais lentamente do que a queda observada durante a fase transiente.

Parece haver um limite inferior para V_{max}, abaixo do qual a exploração do espaço adquire um caráter mais e mais aleatório, reduzindo sua eficiência. A justificativa para o aumento na energia média a partir deste ponto estável é que, quanto mais aumentamos V_{max}, mais imóveis permitimos que as partículas se tornem, já que as probabilidades de permanência em um estado (para uma dada dimensão d) são diretamente proporcionais a $|v_d|$, como descrito na seção 2.4.6.1.

Para o presente estudo de caso, concluímos que a faixa ótima para o parâmetro V_{max} está entre 5 e 8. A Tabela 6.4 ilustra o efeito da variação de V_{max} em alguns dos experimentos.

Tabela 6.4: Exemplo do efeito da variação de V_{max}

S	V_{max}	φ	Energia média	Nº de avaliações médio
250	1	4	107393.64	278930
250	2	4	90997.96	467070
250	3	4	66502.58	695895
250	4	4	39659.98	928300
250	5	4	26677.36	914525
250	7	4	26580.14	659165
250	8	4	28447.64	516430
250	10	4	28814.80	506785
250	11	4	28442.04	495925
250	20	4	29374.78	508025
250	30	4	29907.42	509240
250	40	4	30222.14	504125
250	60	4	31609.52	512720

6.4.2.4 Os parâmetros φ_1 e φ_2

Como o comportamento do resolvedor em função das variações de φ_1 e φ_2 depende bastante das configurações dos demais parâmetros, comecemos observando os grupos *global 1* e *local 3*, onde $\varphi_1 = \varphi_2$ e $\varphi_0 = 1$. Quando V_{max} está fora da fase transiente, valores menores para φ_1 e φ_2 resultam em um melhor desempenho em termos de energia média. Em compensação, o número médio de avaliações cresce rapidamente

à medida em que φ_1 e φ_2 de aproximam de 0. Para valores menores de V_{max}, φ_1 e φ_2 parecem não exercer muita influência sobre o resultado dos experimentos, seja no número médio de avaliações ou na energia média.

Passemos agora para o grupo *global 2*. Como constatado na seção 6.4.2.2, quanto maior a diferença $1-\varphi_0$, mais aleatória se torna a busca, pois cresce a tendência das velocidades de assumirem valores próximos de 0. Valores altos para φ combatem a diminuição da velocidade, amortizando o efeito devido a φ_0. Assim, nos experimentos onde $\varphi_0 < 1$, o desempenho melhora à medida em que φ é incrementado.

Já quando $\varphi_0 > 1$, o resolvedor busca rapidamente a convergência. Aparentemente, para cada φ_0, existem valores mínimos para os parâmetros φ e V_{max} que, combinados, melhoram significativamente o desempenho do PSO. A partir desta combinação, incrementos em φ fazem com que o resolvedor responda da mesma forma que quando $\varphi_0 = 1$: com uma ascensão suave na energia média. O salto no desempenho que ocorre quando φ e V_{max} atingem os valores mínimos é um fenômeno inesperado, e seria precipitado atribuir este acontecimento a qualquer motivo sem a realização de um estudo mais aprofundado no assunto.

Finalmente, analisemos o grupo *global 3*, que atribui valores para φ_1 e φ_2. O gráfico 6.17 da seção 6.3.4 nos fornece algumas informações. Primeiro, que valores de φ_1 e φ_2 cuja soma seja igual a (e provavelmente menor que) 0.1, dados os outros parâmetros conforme o grupo *global 3*, não alcançam bons resultados, se comparados aos experimentos com outros valores para φ_1 e φ_2. Entretanto, a informação mais importante é que o desempenho melhora na medida em que cresce a razão φ_1/φ_2. Isto significa que, para este estudo de caso, o PSO funciona melhor priorizando a tendência da exploração de cada partícula isolada (individualismo) em detrimento da tendência de exploração coletivista. Os resultados com $\varphi_1 > \varphi_2$ superam os obtidos no grupo *global 1*, onde $\varphi_1 = \varphi_2$. Isto não é esperado, dada a pouca atenção que a relação entre φ_1 e φ_2 recebe na literatura.

Ainda em *global 3*, podemos observar no gráfico 6.19 da seção 6.3.4 que o fator determinante para o desempenho neste grupo não é φ, mas a relação entre φ_1 e φ_2. No referido gráfico vemos que, para $0.25 \leq \varphi \leq 0.55$, todos os valores testados para φ apresentam resultados de qualidade variada. As linhas presentes no gráfico 6.17 conectam os experimentos, para cada valor de V_{max} em ordem crescente de

energia média. Como a orientação destes segmentos de reta é semelhante à das retas $k = \varphi_1/\varphi_2$, e não a $k = -\varphi_1/\varphi_2$, para diferentes valores de k, podemos inferir que a razão entre os parâmetros, na faixa do grupo *global 3*, é mais importante para o desempenho do que seus valores absolutos.

A Tabela 6.5 contém um exemplo do comportamento típico de φ_1 e φ_2 no grupo *global 3*.

Tabela 6.5: Exemplo do efeito da variação de φ_1 e φ_2

S	V_{max}	φ_1	φ_2	φ_1/φ_2	φ	Energia média	Média de avaliações
180	11	0.051	0.198	0.258	0.25	19088.36	2223457.2
180	11	0.060	0.189	0.317	0.25	19523.36	2096146.8
180	11	0.075	0.174	0.431	0.25	20843.56	1851739.2
180	11	0.100	0.149	0.671	0.25	22752.12	1630015.2
180	11	0.150	0.100	1.500	0.25	25543.30	1440892.8
180	11	0.174	0.075	2.320	0.25	26466.06	1334610.0
180	11	0.189	0.060	3.150	0.25	27934.20	1268658.0
180	11	0.198	0.051	3.882	0.25	27226.92	1316091.6

6.4.2.5 Os modelos de vizinhança e o parâmetro N

Todos os grupos de experimentos utilizam o modelo de vizinhança global, exceto pelos grupos *local 1*, *local 2* e *local 3*, que realizam testes com a vizinhança em anel generalizada. Comparando-os com o grupo *global 1*, podemos avaliar a eficiência relativa de cada um dos modelos. A Tabela 6.6 ilustra o comportamento típico observado: o modelo de vizinhança global consegue, com menos avaliações da função objetivo, encontrar soluções associadas a valores de energia mais baixos do que o mesmo experimento utilizando o modelo de vizinhança em anel generalizada.

Excluindo das considerações desta seção o grupo *global 1*, é possível perceber que o parâmetro N pouco influi no desempenho, seja este medido com base na energia média alcançada ou no número médio de avaliações, sugerindo que, para este estudo de caso, é mais eficiente utilizar o modelo de vizinhança global.

Tabela 6.6: Exemplo do efeito da variação de S

S	V_{max}	N	φ	Energia média	Nº de avaliações médio
180	30	3	0.1	41204.76	2273353.2
180	30	9	0.1	41177.44	2228223.6
180	30	18	0.1	40686.64	2295799.2
180	30	180	0.1	27215.20	1918447.2

6.5 Comentários

6.5.1 Grupos *wglobal 1* e *wglobal 2*

Apesar de não terem colaborado para a análise do comportamento do PSO, o fato de o resolvedor ter respondido às alterações nos parâmetros da mesma forma em grande parte dos experimentos nos leva a crer que, apesar de as configurações ótimas do PSO serem altamente dependentes do problema-alvo, a avaliação apresentada na seção 6.4 pode servir como guia para experimentações futuras, principalmente para outras modelagens utilizando o SATyrus.

Relativo ao desempenho deste grupo em termos absolutos, é necessário colocá-lo na mesma escala dos outros experimentos, pois há uma enorme disparidade no valor das penalidades associadas às restrições de integridade, impossibilitando a comparação direta via energia média. Isto pode ser aproximado medindo-se o número de restrições violadas em cada um dos experimentos, ao invés da energia média. Comparando *as melhores soluções* encontradas por cada uma das três modelagens (*wglobal 1*, *wglobal 2* e demais experimentos) em termos do número de restrições violadas, observamos que o desempenho de *wglobal 1* e *wglobal 2* foi muito inferior. Um trecho desta comparação pode ser visto na Tabela 6.7, onde níveis de significância aumentam de forma diretamente proporcional à proximidade do bit de sinal.

6.5.2 Outros resolvedores

Os resolvedores bsolo e Minisat+ (apresentados nas seções 2.3.3 e 2.3.4), assim como a Têmpera Simulada (seção 2.2) e uma outra heurística para o PSO (seção 2.4.6.2) também foram testados.

O bsolo teve sua execução abortada duas vezes, após ultrapassar o limite estabe-

Tabela 6.7: Violações em restrições de somadores, por nível de significância do bit

	wglobal 1	*wglobal 2*	*global 3*
5	0	0	7
4	3	3	6
3	29	9	5
2	63	52	4
1	73	67	5
total	168	131	27

lecido de 168 horas sem retornar. O Minisat+ também não conseguiu completar a execução, mas por um motivo diferente: estourou o limite de memória RAM disponível na máquina onde executava e foi interrompido pelo sistema operacional. Um grande volume de experimentos com esses resolvedores não é necessário, já que seu comportamento não é estocástico.

A Têmpera Simulada apresentou resultados comparáveis aos do PSO binário na bateria de testes realizada. Entretanto, o número de avaliações da função objetivo necessário para a obtenção de valores de energia foi bastante superior, o dobro do necessário com o PSO, na maior parte dos testes.

Também foi testada a segunda heurística para o PSO binário, como descrita na seção 2.4.6.2. Entretanto, os resultados obtidos ficaram tão aquém dos apresentados na seção 6.3 que julgamos ser suficiente mencioná-los brevemente aqui. A melhor energia média alcançada foi da ordem de 10^5, uma ordem de grandeza acima do melhor resultado nos experimentos apresentados.

Capítulo 7

Conclusão

7.1 Comentários finais

O presente trabalho apresentou uma nova alternativa para estudos com o GEP, um importante problema da área de planejamento energético. Uma primeira modelagem do GEP para a plataforma sintetizadora de funções de energia SATyrus foi introduzida. A instância modelada não é do mesmo porte das instâncias reais, mas foi escolhida por possibilitar a condução dos testes desejados com os resursos computacionais disponíveis. Felizmente, para o caso deste trabalho, o esforço necessário para a modelagem não é dependente do tamanho da instância, mas sim da complexidade do problema.

Os testes realizados com o PSO binário infelizmente não alcançaram o valor ótimo da função objetivo. Apesar da energia da solução teórica (53) diferir bastante da melhor encontrada nos experimentos (2542), vale lembrar que esta solução encontrada representa uma violação de menos de 0.13% das 8259 restrições da modelagem, um resultado que não deve ser desprezado.

Também foi realizada uma grande quantidade de experimentos abrangendo um amplo espectro do espaço de parâmetros do PSO binário. Tais experimentos proporcionaram um melhor entendimento sobre o comportamento deste resolvedor e possivelmente farão parte da composição de uma base para estudos futuros, pois este tipo de análise sobre o resolvedor adotado não abunda, se comparado a outros mais difundidos.

O trabalho também teve por consequência a inclusão de problemas de planejamento energético no contexto do projeto EELA, mostrando a comunidade da área que grids podem ser úteis no auxílio à resolução de problemas como o GEP.

7.2 Trabalhos futuros

A modelagem do GEP para o SATyrus, assim como os experimentos realizados inspiram uma grande variedade de trabalhos futuros. Na área do planejamento energético, é possível modelar e tentar solucionar instâncias maiores do GEP, tanto no número de usinas e estágios considerados como na quantidade de fatores: impacto ambiental, cenários de demanda, restrições políticas etc.

Como trabalho futuro no SATyrus, propõe-se o estudo de uma forma de gerar funções de energia mistas, que contemple dimensões booleanas e contínuas. Com isso a modelagem de problemas que contém variáveis contínuas, como o GEP, não teria seu número de dimensões elevado devido à necessidade da conversão das variáveis contínuas em vetores de booleanas. Uma consequência certa de tal trabalho seria a melhoria no desempenho do PSO, pois a função de energia se aproximaria da formulação matemática, onde o PSO (contínuo) já foi utilizado com sucesso [14].

Sabemos que o PSO não é o único resolvedor existente, e experimentos com a modelagem apresentada podem ser conduzidos utilizando outros métodos de resolução. Esperamos que tais experimentos atinjam bons resultados.

Trabalhos futuros focados no PSO binário incluem a experimentação com outras variantes, como as descritas em [8], assim como o estudo de diferentes topologias de vizinhança. Pode ser benéfica para o desempenho a criação de um parâmetro V_{min} no contexo binário, que teria o efeito de amenizar a movimentação das partículas, apresentando talvez um comportamento diferente do observado através do uso de $\varphi_0 > 1$. Além disso, visamos estudar alternativas para a paralelização do resolvedor que funcionem bem, principalmente, em ambientes massivamente distribuídos como os grids. Tal paralelização certamente será necessária para a resolução da modelagem de instências maiores do GEP, que necessitará de um volume de recursos computacionais bem maior do que o preciso para trabalhar com uma instância como a apresentada aqui.

Referências Bibliográficas

[1] Electric generation expansion analysis system. Relatório técnico, 1982.

[2] E-infrastructure shared between europe and latin america. http://www.eu-eela.org/. 2007.

[3] JSwarm-PSO. http://jswarm-pso.sourceforge.net/. 2007.

[4] Portal do laboratório de otimização avançada. http://loa.ppe.ufrj.br/. 2007.

[5] Agência nacional de energia elétrica. http://www.aneel.gov.br/. 2008.

[6] Empresa de pesquisa energética - EPE. http://www.epe.gov.br/. 2008.

[7] ILOG CPLEX: High-performance software for mathematical programming and optimization. http://www.cplex.com/. 2008.

[8] CLERC, M. Binary particle swarm optimisers: toolbox, derivations, and mathematical insights, *http://clerc.maurice.free.fr/pso/binary_pso*, 2005.

[9] CLERC, M., KENNEDY, J. "The particle swarm - explosion, stability, and convergence in a multidimensional complex space", *IEEE Transactions on Evolutionary Computation* v. 6, n. 1, pp. 58–73, 2002.

[10] COUDERT, O. "On solving covering problems", In: *Proceedings of the 33rd Design Automation Conference*, pp. 197–202, Las Vegas, Jun. 1996.

[11] EÉN, N., SORENSSON, N. "Translating pseudo-boolean constraints into SAT", *Journal on Satisfability, Boolean Modeling and Computation* v. 2, n. 1, pp. 1–25, 2006.

[12] FIRMO, H. *O Planejamento da Expansão da Geração do Setor Elétrico Brasileiro Utilizando os Algoritmos Genéticos.* Tese de Doutorado, Programa de Planejamento Energético, COPPE/UFRJ, Rio de Janeiro, 2001.

[13] HOPFIELD, J. J. "Neural networks and physical systems with emergent collective computational abilities", In: *Proceedings of the National Academy of Sciences*, pp. 2554–2558, 1982.

[14] KANNAN, S., SLOCHANAL, S. M. R., SUBBARAJ, P., PADHY, N. P. "Application of particle swarm optimization technique and its variants to generation expansion planning problem", *Electric Power Systems Research* v. 70, n. 3, pp. 203–210, 2004.

[15] KENNEDY, J. "The behavior of particles", In: *Proceedings of the 7th annual conference on evolutionary programming*, pp. 581–589, San Diego, 1998.

[16] KENNEDY, J., EBERHART, R. C. "A new optimizer using particle swarm theory", In: *Proceedings of the Sixth International Symposium on Micro Machine an Human Science*, pp. 39–43, Nagoya, Out. 1995.

[17] KENNEDY, J., EBERHART, R. C. "Particle swarm optimization", In: *Proceedings of the IEEE International Conference on Neural Networks* v. 4, pp. 1942–1948, Perth, Nov. 1995.

[18] KENNEDY, J., EBERHART, R. C. "A discrete binary version of the particle swarm algorithm", In: *Proceedings of the Conference on Systems, Man and Cybernetics* v. 5, pp. 4104–4109, Orlando, 1997.

[19] KENNEDY, J., EBERHART, R. C., SHI, Y. *Swarm Intelligence.* 1 ed. San Francisco, Morgan Kaufmann Publishers, 2001.

[20] KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P. "Optimization by simulated annealing", *Science* v. 220, n. 4598, pp. 671–680, 1983.

[21] LEGEY, L. F. L., FIRMO, H. "Generation expansion planning: A genetic algorithm approach", In: *Proceedings of the International Conference on Intelligent System Applications to Power Systems* v. 1, pp. 201–207, Rio de Janeiro, 1999.

[22] MANQUINHO, V. M., FLORES, P. F., SILVA, J. P. M., OLIVEIRA, A. L. "Prime implicant computation using satisfiability algorithms", In: *Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence*, pp. 232–239, Newport Beach, 1997.

[23] METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., TELLER, E. "Equation of state calculations by fast computing machines", *Journal of Chemical Physics* v. 21, n. 6, pp. 1087–1092, 1953.

[24] MORVELI-ESPINOZA, M. M. M. *Compilando Resolução de Problemas para Minimização de Energia.* Tese de Mestrado, Programa de Engenharia de Sistemas, COPPE/UFRJ, Rio de Janeiro, 2006.

[25] NEMHAUSER, G. L., WOLSEY, L. A. *Integer and Combinatorial Optimization.* New York, John Wiley & Sons, 1998.

[26] PEREIRA, G. C. *Mapeamento e Combinação de Problemas NP-difíceis através de Restrições Pseudo-Booleanas para Redes Neuronais Artificiais.* Tese de Mestrado, Programa de Engenharia de Sistemas, COPPE/UFRJ, Rio de Janeiro, 2006.

[27] PINKAS, G. *Resolution-Based Inference on Artificial Neural Networks.* Tese de Doutorado, Sever Institute of Technology, Washington University, Saint Louis, 1992.

[28] SHI, Y., EBERHART, R. C. "A modified particle swarm optimizer", In: *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 69–73, Anchorage, 1998.

[29] TSEITIN, G. "On the complexity of derivation in propositional calculus", *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pp. 115–125, 1970.

Apêndice A

Códigos-fonte

A.1 Somador binário

```
 //Arquivo soma.txt
bits=5;
n1(bits);
n2(bits);
res(bits);
vai1(bits+1);
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or
   n1[b] or not n2[b] or res[b]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or not
   n1[b] or n2[b] or aux1[b]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   n1[b] or n2[b] or aux1[b]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   not n1[b] or not n2[b] or aux1[b]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or
   n1[b] or n2[b] or not aux1[b]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   not n1[b] or n2[b] or not aux1[b]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   n1[b] or not n2[b] or not aux1[b]);
```

```
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or not
   n1[b] or not n2[b] or not aux1[b]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   not n1[b] or not n2[b] or vai1[b+1]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   not n1[b] or n2[b] or vai1[b+1]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or not
   n1[b] or not n2[b] or vai1[b+1]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   n1[b] or not n2[b] or vai1[b+1]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or
   n1[b] or n2[b] or not vai1[b+1]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or
   n1[b] or not n2[b] or not vai1[b+1]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (vai1[b] or not
   n1[b] or n2[b] or not vai1[b+1]);
integrity group type alfa:  forall{b}; 1<=b<=5:  (not vai1[b] or
   n1[b] or n2[b] or not vai1[b+1]);
penalty{alfa is level 0;}

 //Arquivo "n1"
1-1-1-1
2-1-1-1
3-0-1-1
4-0-1-1
5-0-1-1

 //Arquivo "n2"
1-1-1-1
2-0-1-1
3-1-1-1
4-0-1-1
5-0-1-1
```

A.2 GEP

A.2.1 Arquivo principal

```
 //Arquivo gep.txt
nU=5; //# de usinas
nE=3; //# de estagios
nB=15; //# de bits
cap(nU, nB); //capacidade
dem(nE, nB); //demanda
C1(nU, nE, nB+1); //carry 1
C2(nU, nE, nB+1); //carry 2
C3(1, nE, nB+1); //carry 3
A1(nU, nE, nB); //acumulador 1
A2(nU+2, nE, nB); //acumulador 2
x(nU,nE);
y(nU, nE, nB);

cCon read from cCon.txt; //custos de construcao
cOp read from cOp.txt; //custos de operacao

//(a) Plant Operation Constraints - Nao construir a mesma usina
   mais de uma vez.
integrity group type a:  forall{i,j,k};1<=i<=nU,1<=j<=nE,
   1<=k<=nE;j!=k:(not x[i][j] or not x[i][k]);

//(b) Supply constraints - Nao exceder as caps das usinas.
// As caps estao representadas no arquivo como numeros negativos.
// Soma y com a cap
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or y[i][j][b] or not cap[i][b] or A1[i][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or not y[i][j][b] or cap[i][b] or A1[i][j][b]);
```

```
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or y[i][j][b] or cap[i][b] or A1[i][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or not y[i][j][b] or not cap[i][b] or A1[i][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or y[i][j][b] or cap[i][b] or not A1[i][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or not y[i][j][b] or cap[i][b] or not A1[i][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or y[i][j][b] or not cap[i][b] or not A1[i][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or not y[i][j][b] or not cap[i][b] or not A1[i][j][b]);

integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or not y[i][j][b] or not cap[i][b] or C1[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or not y[i][j][b] or cap[i][b] or C1[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or not y[i][j][b] or not cap[i][b] or C1[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or y[i][j][b] or not cap[i][b] or C1[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or y[i][j][b] or cap[i][b] or not C1[i][j][b +
   1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or y[i][j][b] or not cap[i][b] or not C1[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C1[i][j][b] or not y[i][j][b] or cap[i][b] or not C1[i][j][b
   + 1]);
```

```
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C1[i][j][b] or y[i][j][b] or cap[i][b] or not C1[i][j][b
   + 1]);

// Garante que o resultado da soma acima eh negativo.
integrity group type a:  forall{i,u,d,t,s}; 1<=i<=nU;u=1,d=2,t=3,s=nB:
   A1[i][u][s] and A1[i][d][s] and A1[i][t][s];

// Garante que so usinas construidas possam produzir.
integrity group type a:  forall{i,u,b}; 1<=i<=nU,1<=b<=nB-1;u=1:
   (not y[i][u][b] or x[i][u]);
integrity group type a:  forall{i,u,d,b}; 1<=i<=nU,1<=b<=nB-1;u=1,d=2:
   (not y[i][d][b] or x[i][u] or x[i][d]);
integrity group type a:  forall{i,u,d,t,b}; 1<=i<=nU,1<=b<=nB-1;
   u=1,d=2,t=3:  (not y[i][t][b] or x[i][u] or x[i][d] or x[i][t]);

//(c) Demand Constraints - Atender a dem por estagio.

// Soma y[i] com A2[i] e guarda em A2[i+1]
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or y[i][j][b] or not A2[i][j][b] or A2[i+1][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or not y[i][j][b] or A2[i][j][b] or A2[i+1][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or y[i][j][b] or A2[i][j][b] or A2[i+1][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or not y[i][j][b] or not A2[i][j][b]
   or A2[i+1][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or y[i][j][b] or A2[i][j][b] or not A2[i+1][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or not y[i][j][b] or A2[i][j][b]
   or not A2[i+1][j][b]);
```

```
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or y[i][j][b] or not A2[i][j][b]
  or not A2[i+1][j][b]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or not y[i][j][b] or not A2[i][j][b]
  or not A2[i+1][j][b]);

integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or not y[i][j][b] or not A2[i][j][b] or
   C2[i][j][b + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or not y[i][j][b] or A2[i][j][b] or C2[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or not y[i][j][b] or not A2[i][j][b] or C2[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or y[i][j][b] or not A2[i][j][b] or C2[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or y[i][j][b] or A2[i][j][b] or not C2[i][j][b +
   1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or y[i][j][b] or not A2[i][j][b] or not C2[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (C2[i][j][b] or not y[i][j][b] or A2[i][j][b] or not C2[i][j][b
   + 1]);
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE,1<=b<=nB:
   (not C2[i][j][b] or y[i][j][b] or A2[i][j][b] or not C2[i][j][b
   + 1]);

// Subtrai a dem da soma dos ys.  (A2[i][j] + dem[j] = A2[i+1][j])
// i = nU + 1;
```

```
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (C3[u][j][b] or A2[i][j][b] or not dem[j][b] or A2[i+1][j][b]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (C3[u][j][b] or not A2[i][j][b] or dem[j][b] or A2[i+1][j][b]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or A2[i][j][b] or dem[j][b] or A2[i+1][j][b]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or not A2[i][j][b] or not dem[j][b]
  or A2[i+1][j][b]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (C3[u][j][b] or A2[i][j][b] or dem[j][b] or not A2[i+1][j][b]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or not A2[i][j][b] or dem[j][b]
  or not A2[i+1][j][b]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or A2[i][j][b] or not dem[j][b]
  or not A2[i+1][j][b]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (C3[u][j][b] or not A2[i][j][b] or not dem[j][b]
  or not A2[i+1][j][b]);

integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or not A2[i][j][b] or not dem[j][b] or
   C3[u][j][b + 1]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or not A2[i][j][b] or dem[j][b] or C3[u][j][b
   + 1]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (C3[u][j][b] or not A2[i][j][b] or not dem[j][b] or C3[u][j][b
   + 1]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or A2[i][j][b] or not dem[j][b] or C3[u][j][b
   + 1]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
```

```
   (C3[u][j][b] or A2[i][j][b] or dem[j][b] or not C3[u][j][b +
   1]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (C3[u][j][b] or A2[i][j][b] or not dem[j][b] or not C3[u][j][b
   + 1]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (C3[u][j][b] or not A2[i][j][b] or dem[j][b] or not C3[u][j][b
   + 1]);
integrity group type a:  forall{j,b,i,u}; 1<=j<=nE,1<=b<=nB;i=6,u=1:
   (not C3[u][j][b] or A2[i][j][b] or dem[j][b] or not C3[u][j][b
   + 1]);

// s = nU + 2;
integrity group type a:  forall{j,s,b}; 1<=j<=nE;s=7,b=nB: not
   A2[s][j][b];

//(z) Fixar variaveis
integrity group type a:  forall{j,b,u}; 1<=j<=nE,1<=b<=nB;u=1:
   not A2[u][j][b];

integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE;b=1:
   not C1[i][j][b];
integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE;b=1:
   not C2[i][j][b];
integrity group type a:  forall{j,u,b}; 1<=j<=nE;u=1,b=1:  not
   C3[u][j][b];

integrity group type a:  forall{i,j,b}; 1<=i<=nU,1<=j<=nE;b=15:
   not y[i][j][b];

//(d) Funcao Objetivo - minimizar custos...
// ...de construcao...
optimality group type c:  forall{i,u,d,t};1<=i<=nU;u=1,d=2,t=3:
   cCon[i]((x[i][u] or x[i][d] or x[i][t]));
```

```
// ...e de operacao.
optimality group type c:  forall{i,j,b};1<=i<=nU,1<=j<=nE,1<=b<=nB-1:
   cOp[i][b](y[i][j][b]);

penalty{
a is level 1; //integridade
c is level 0;} //otimalidade
```

Apêndice B

Resultados para enxames de tamanho 10, 40 e 180

B.1 Enxames de tamanho 10

B.1.1 Grupo *global 1*

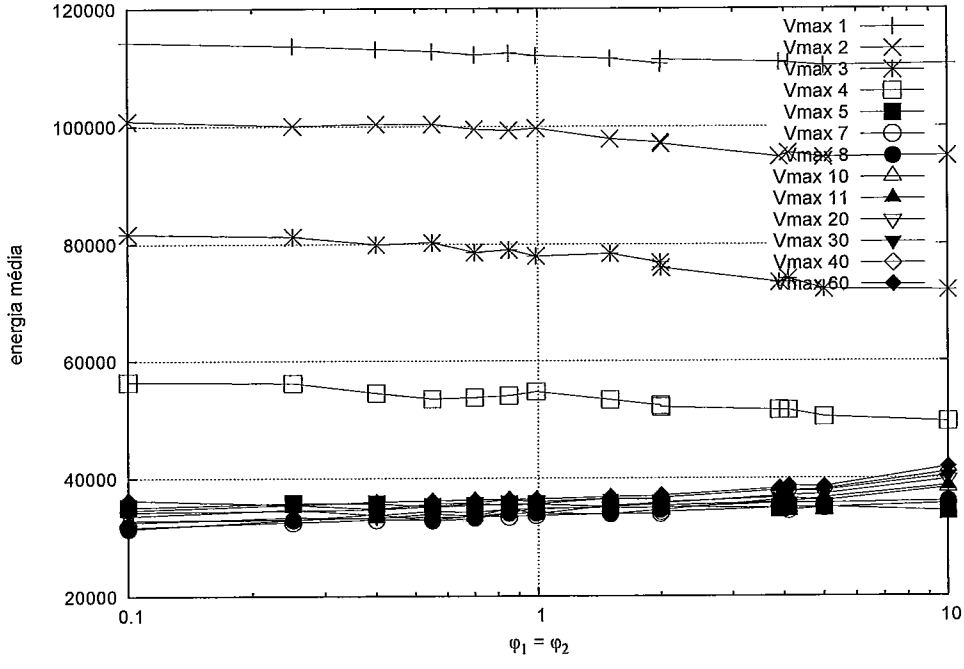

Figura B.1: Resultados: grupo *global 1*, $S = 10$

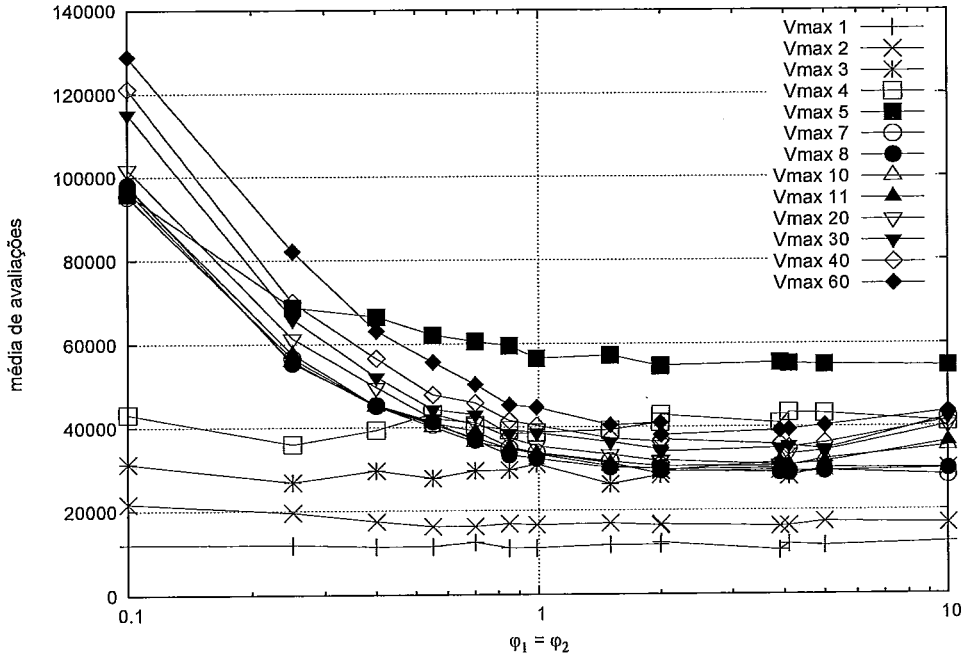

Figura B.2: Resultados: grupo *global 1*, $S = 10$

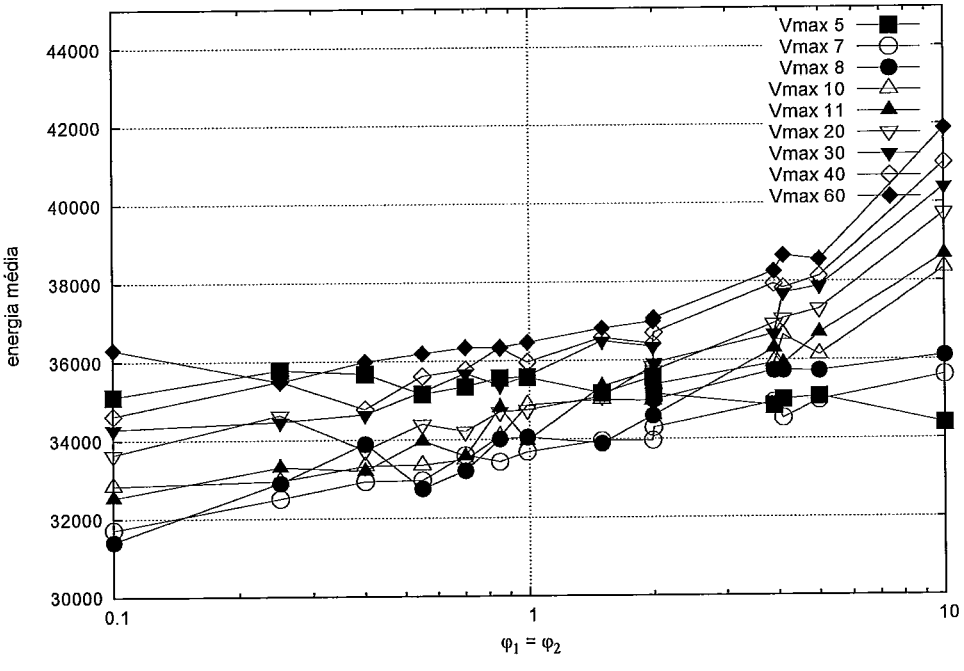

Figura B.3: Resultados: grupo *global 1*, $S = 10$, $V_{max} \geq 5$

B.1.2 Grupo *local 1*

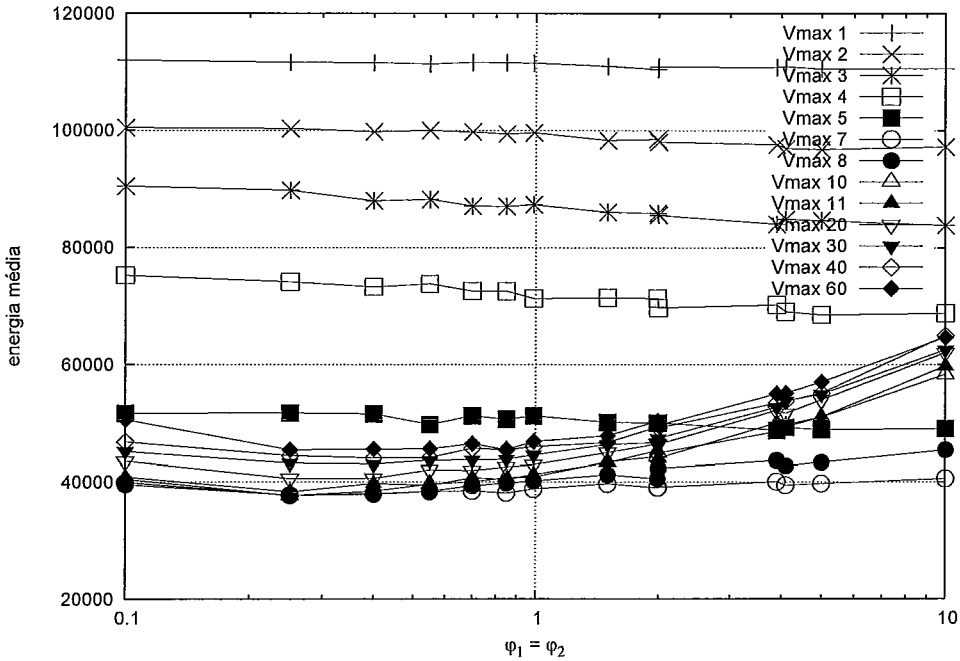

Figura B.4: Resultados: grupo *local 1*, $S = 10$, $N = 3$

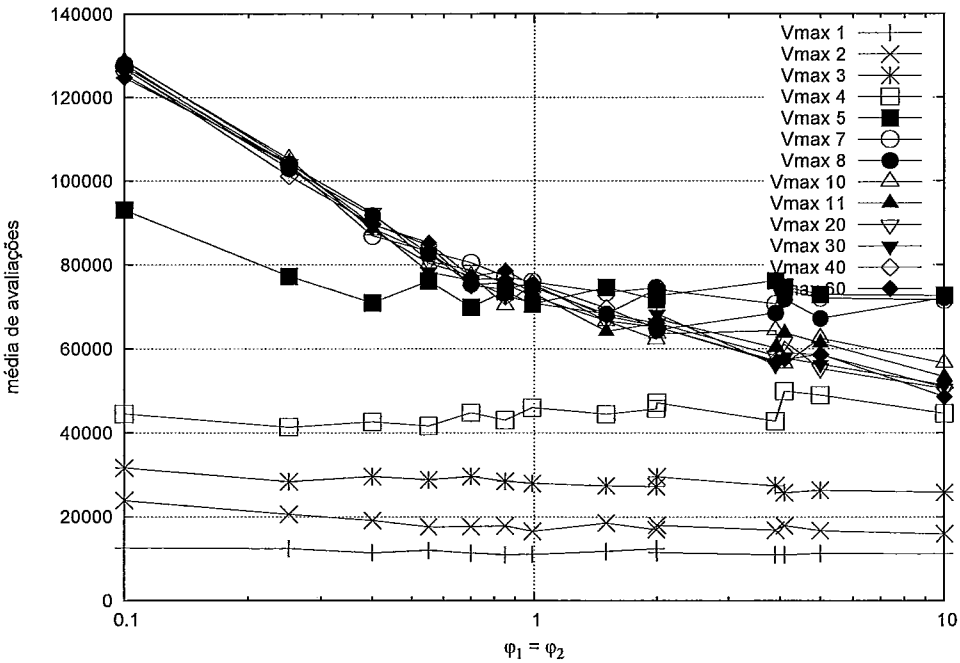

Figura B.5: Resultados: grupo *local 1*, $S = 10$, $N = 3$

B.1.3 Grupo *global 2*

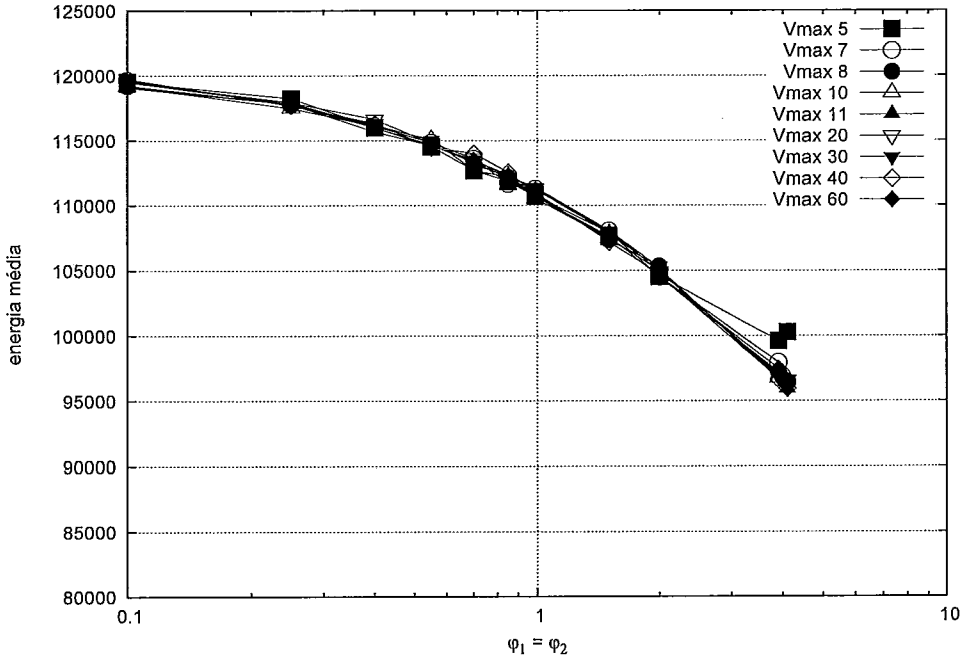

Figura B.6: Resultados: grupo *global 2*, $S = 10$, $\varphi_0 = 0.95$

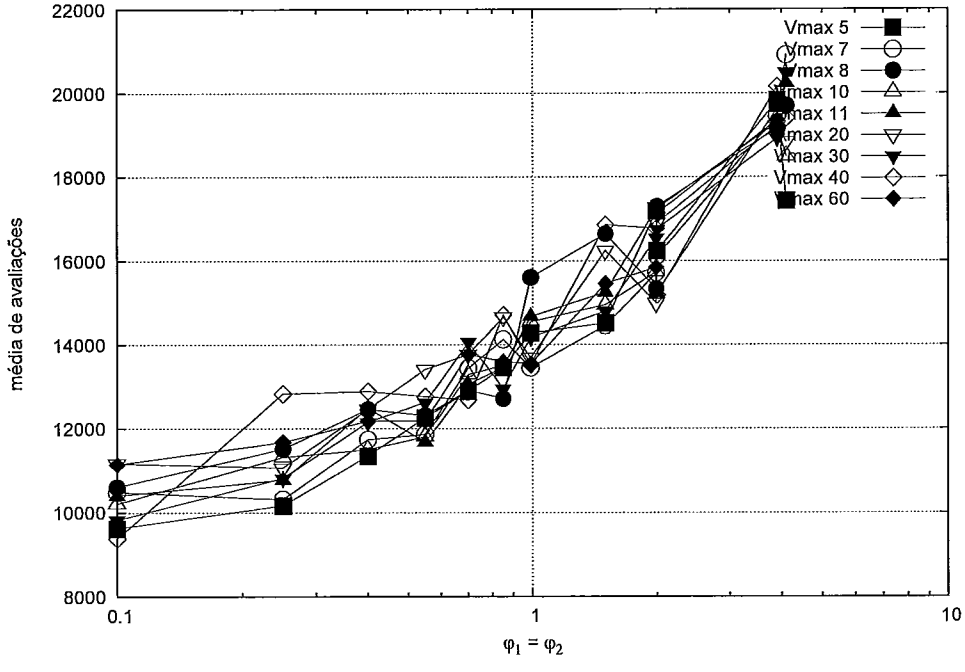

Figura B.7: Resultados: grupo *global 2*, $S = 10$, $\varphi_0 = 0.95$

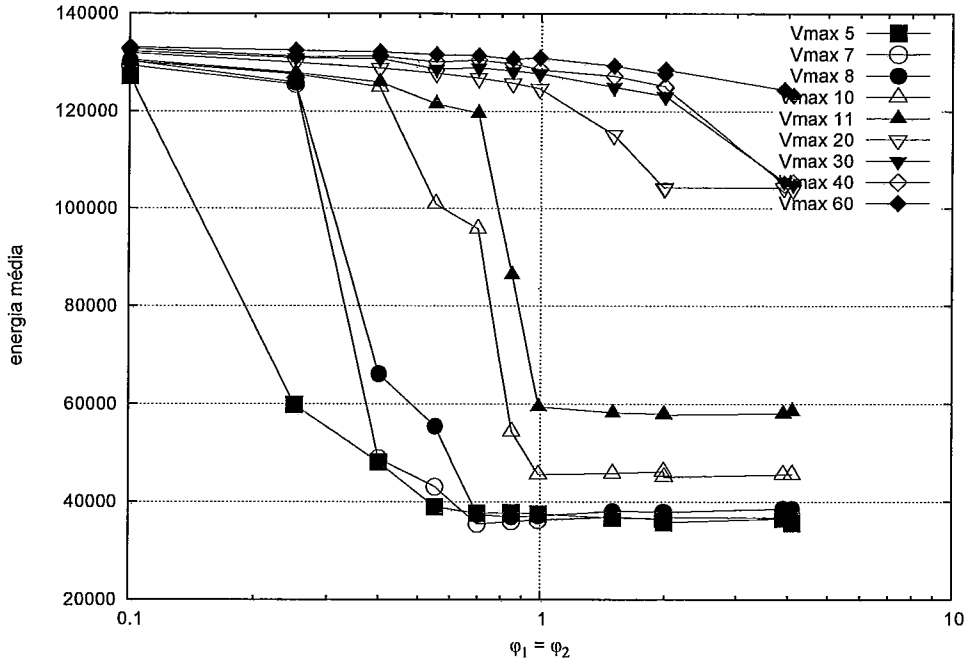

Figura B.8: Resultados: grupo *global 2*, $S = 10$, $\varphi_0 = 1.05$

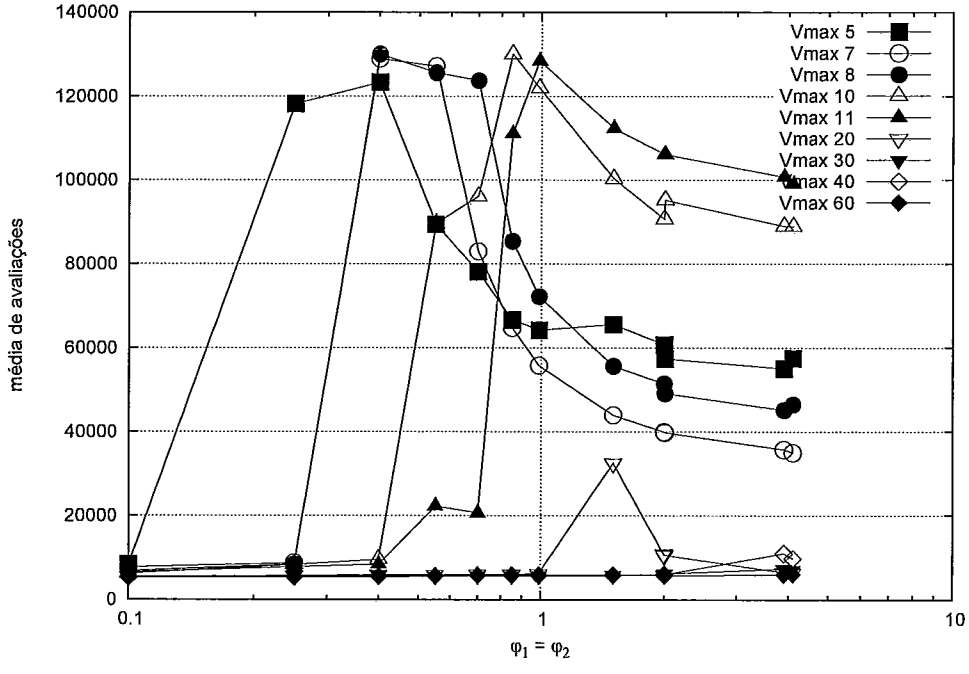

Figura B.9: Resultados: grupo *global 2*, $S = 10$, $\varphi_0 = 1.05$

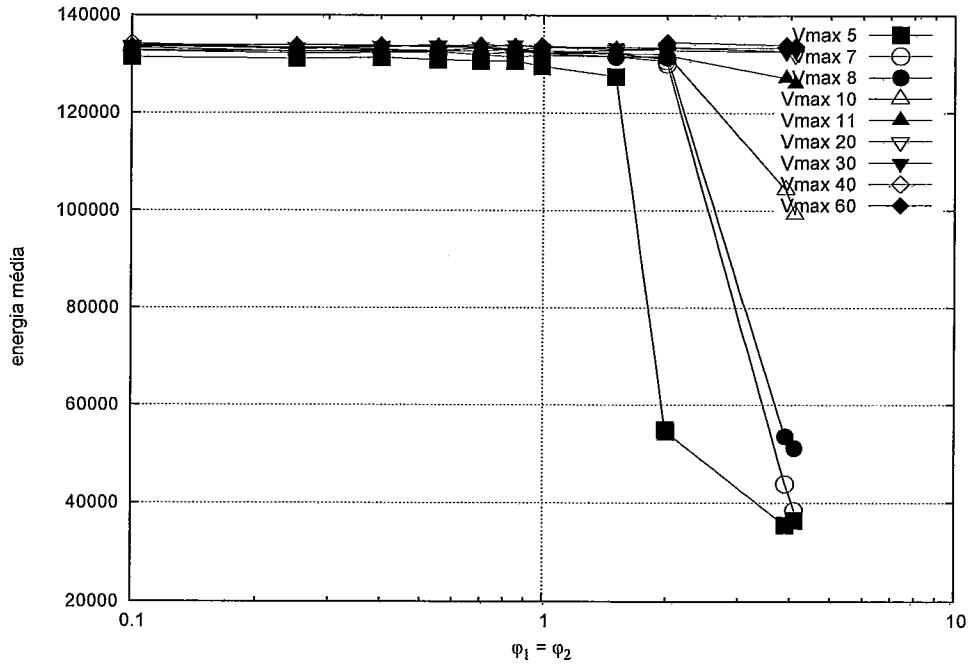

Figura B.10: Resultados: grupo *global 2*, $S = 10$, $\varphi_0 = 2$

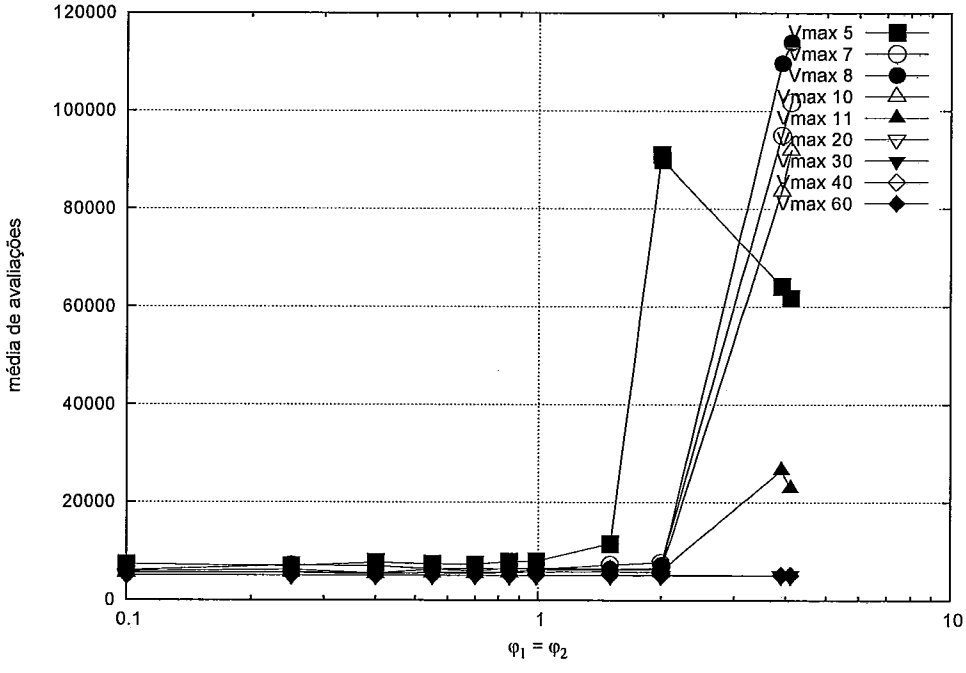

Figura B.11: Resultados: grupo *global 2*, $S = 10$, $\varphi_0 = 2$

B.2 Enxames de tamanho 40

B.2.1 Grupo *global 1*

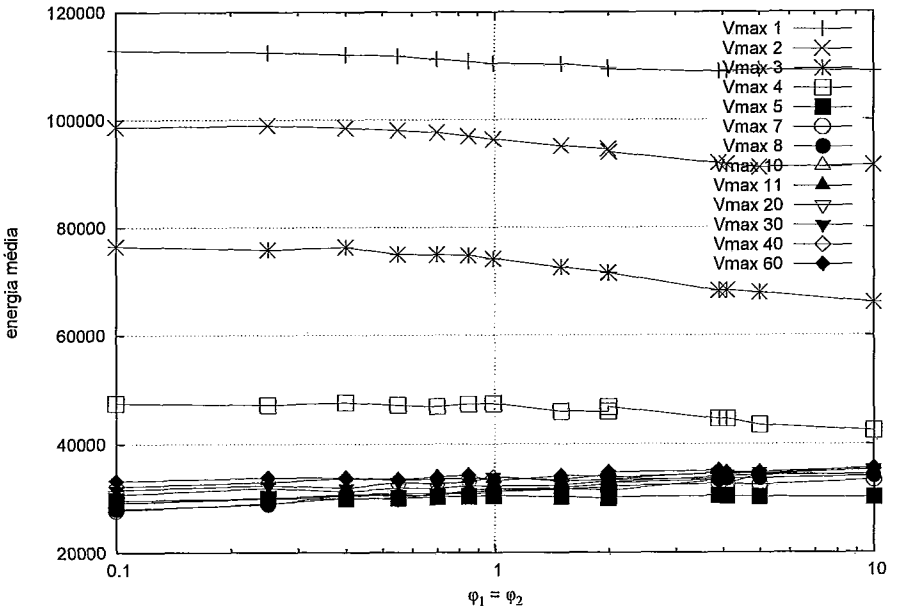

Figura B.12: Resultados: grupo *global 1*, $S = 40$

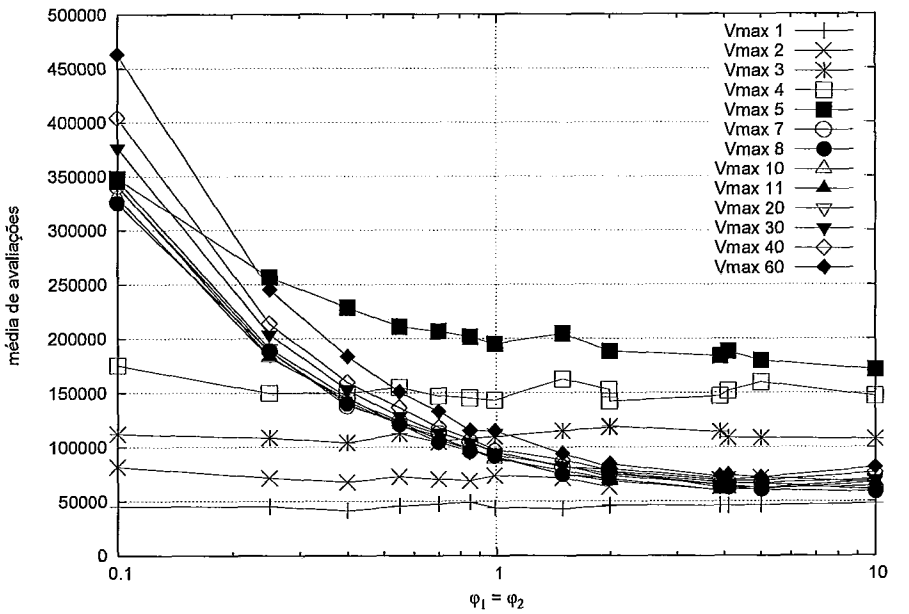

Figura B.13: Resultados: grupo *global 1*, $S = 40$

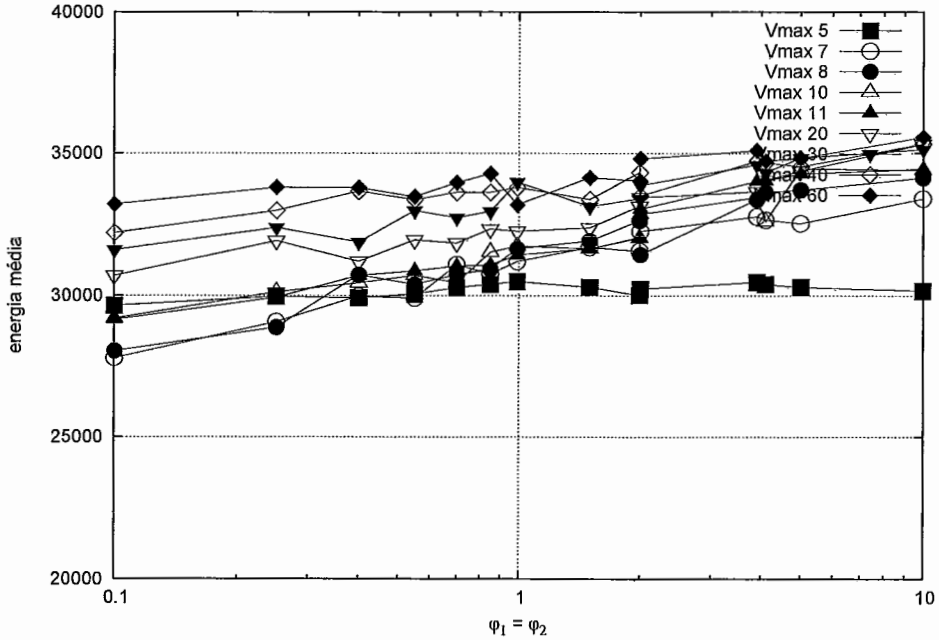

Figura B.14: Resultados: grupo *global 1*, $S = 40$, $V_{max} \geq 5$

B.2.2 Grupo *local 1*

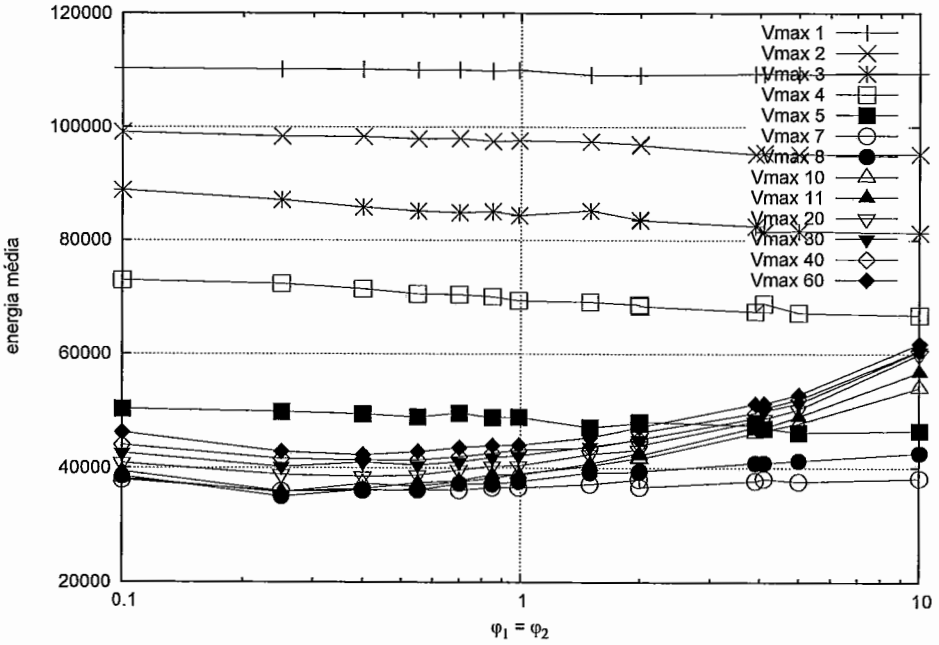

Figura B.15: Resultados: grupo *local 1*, $S = 40$, $N = 3$

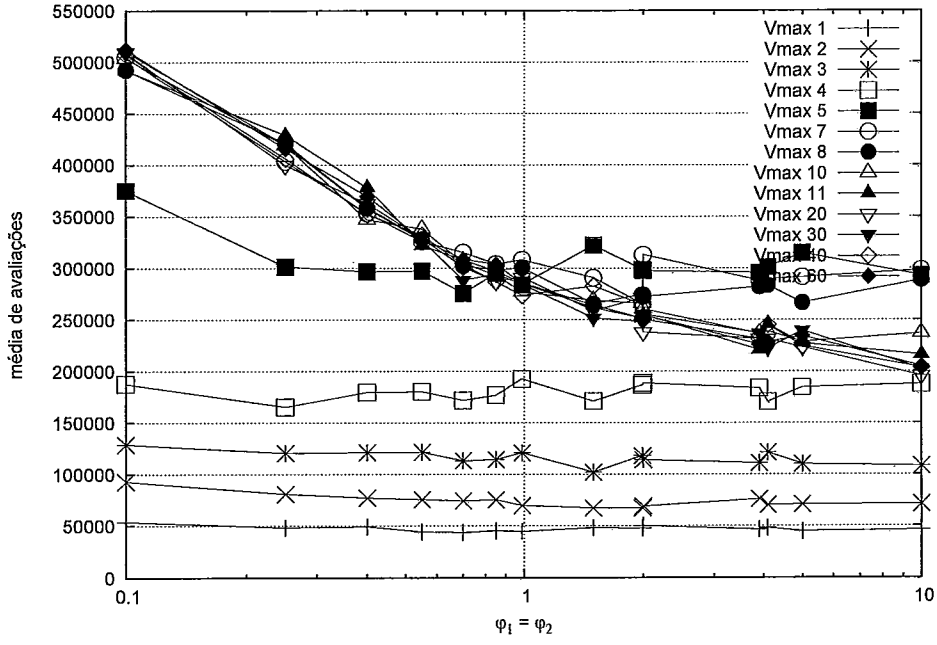

Figura B.16: Resultados: grupo *local 1*, $S = 40$, $N = 3$

B.2.3 Grupo *global 2*

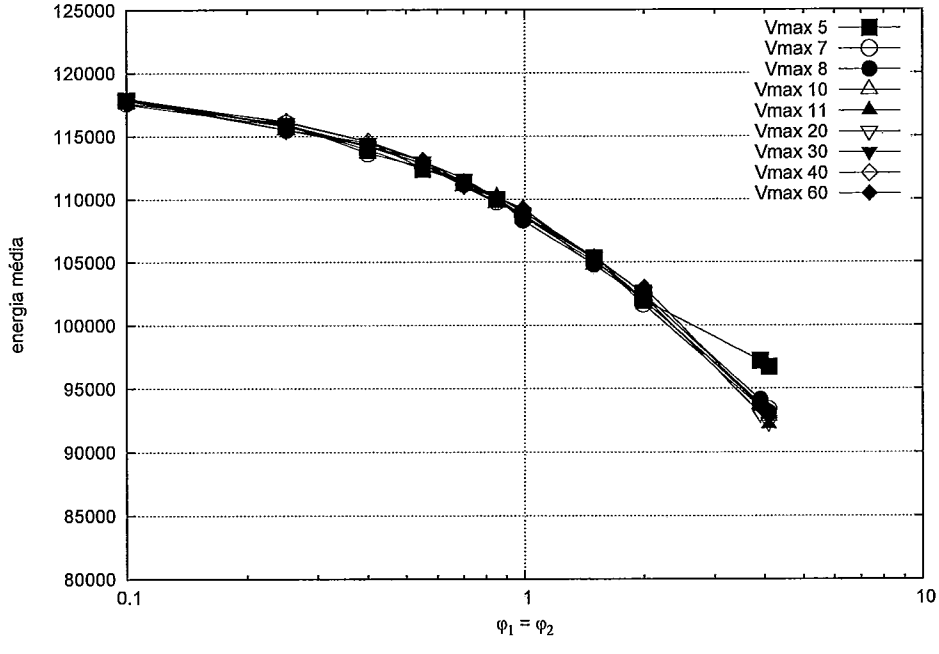

Figura B.17: Resultados: grupo *global 2*, $S = 40$, $\varphi_0 = 0.95$

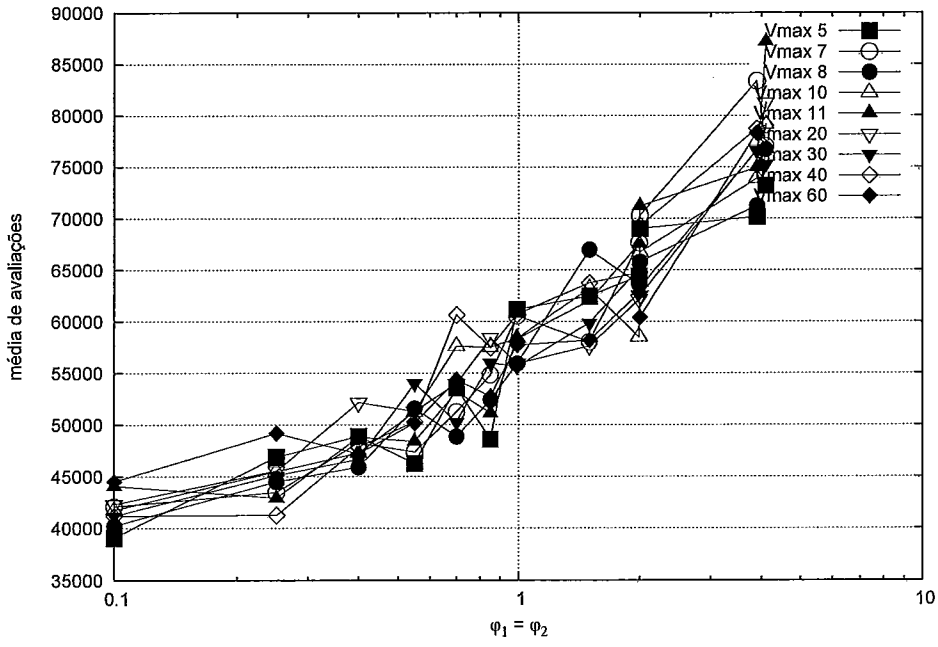

Figura B.18: Resultados: grupo *global 2*, $S = 40$, $\varphi_0 = 0.95$

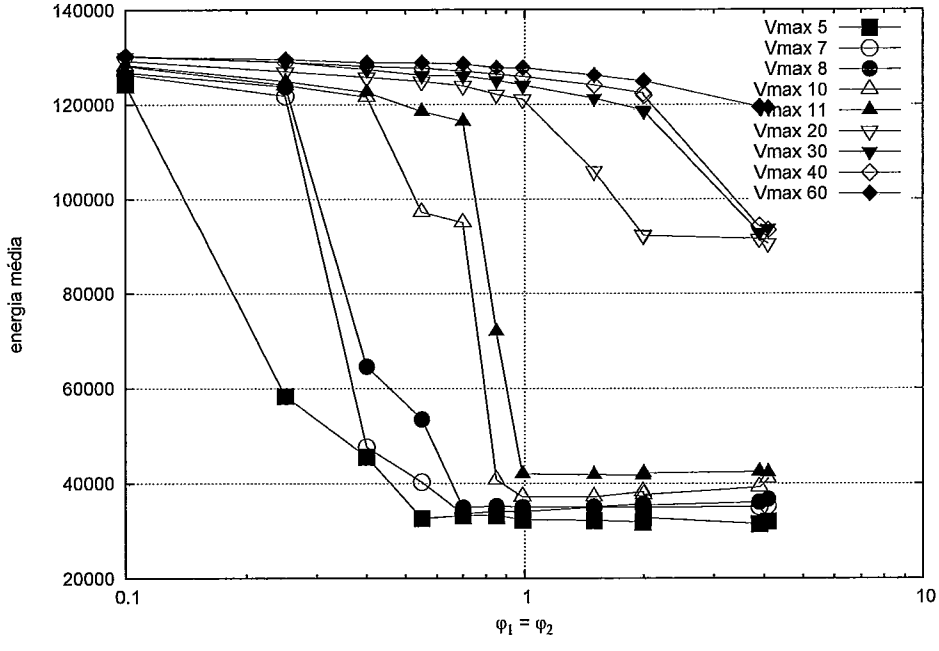

Figura B.19: Resultados: grupo *global 2*, $S = 40$, $\varphi_0 = 1.05$

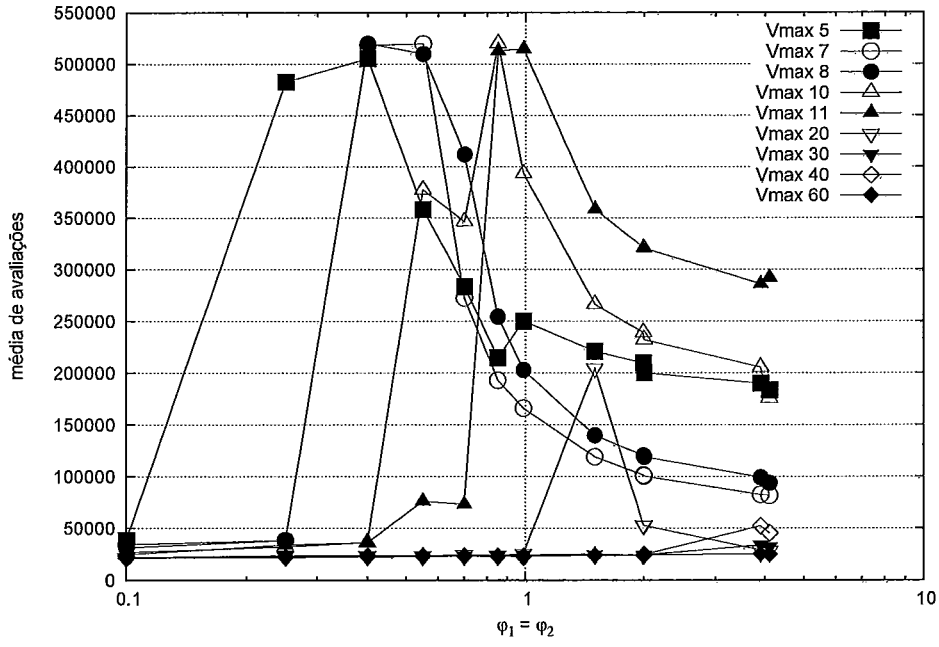

Figura B.20: Resultados: grupo *global 2*, $S = 40$, $\varphi_0 = 1.05$

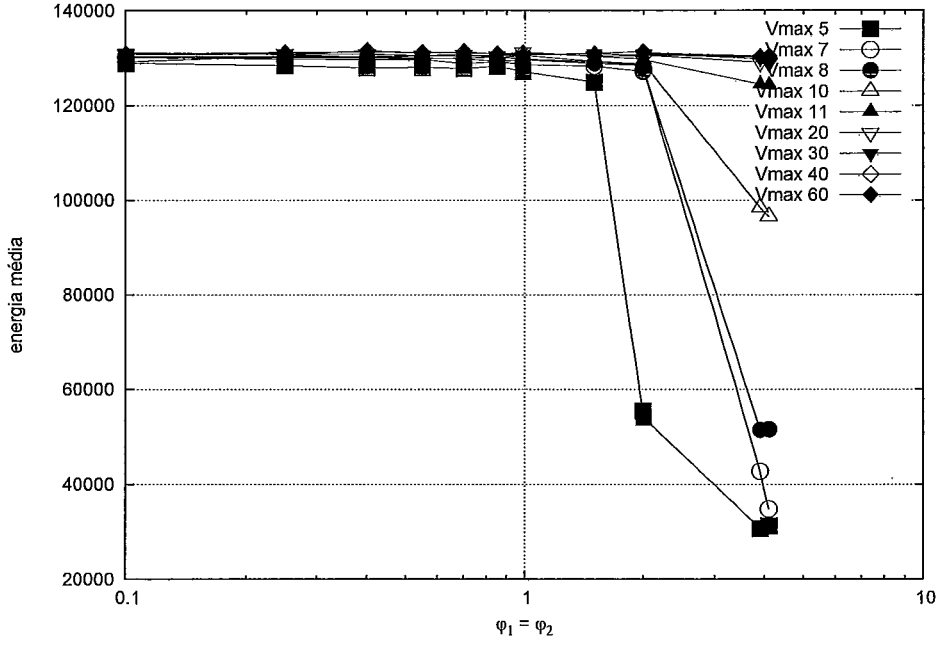

Figura B.21: Resultados: grupo *global 2*, $S = 40$, $\varphi_0 = 2$

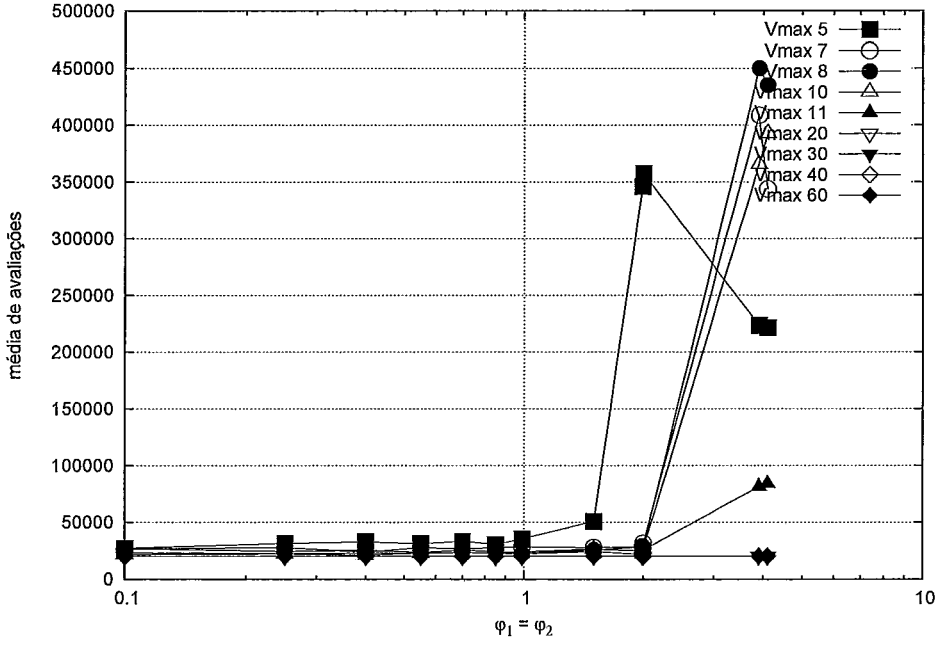

Figura B.22: Resultados: grupo *global 2*, $S = 40$, $\varphi_0 = 2$

B.3 Enxames de tamanho 180

B.3.1 Grupo *global 1*

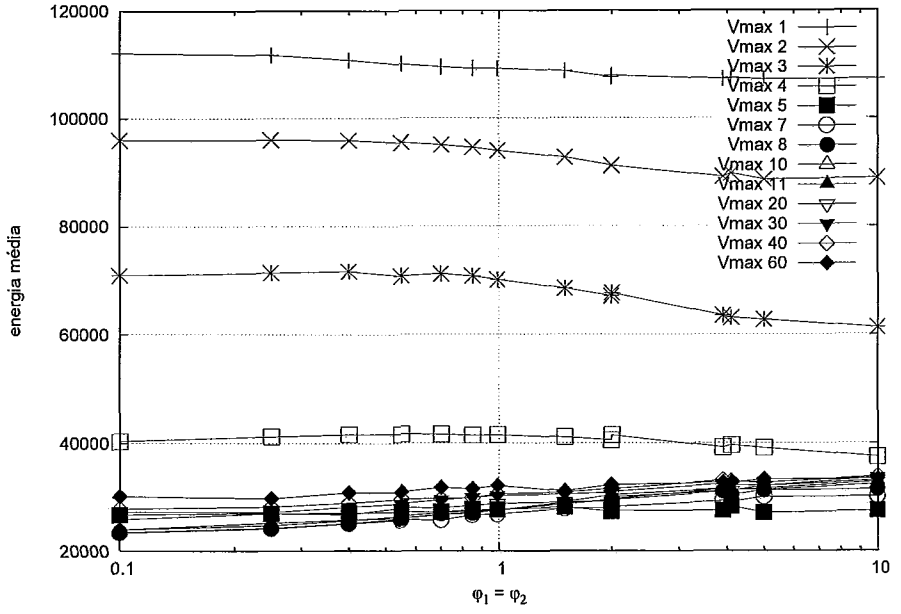

Figura B.23: Resultados: grupo *global 1*, $S = 180$

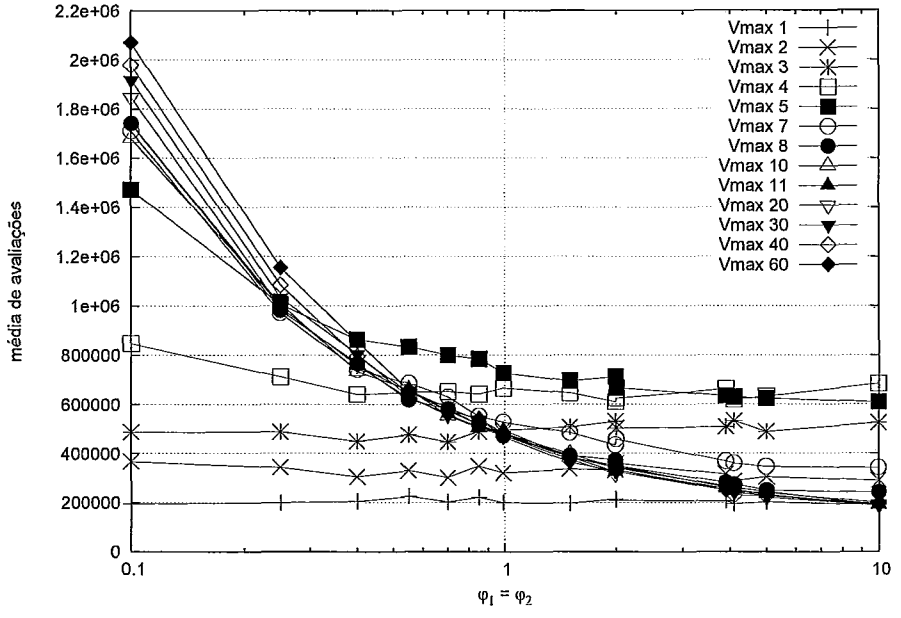

Figura B.24: Resultados: grupo *global 1*, $S = 180$

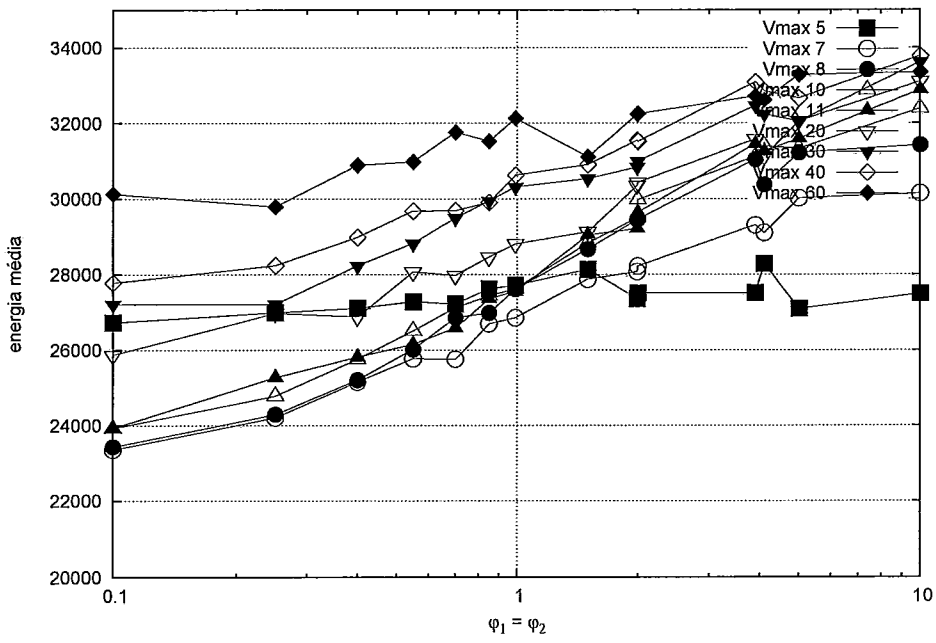

Figura B.25: Resultados: grupo *global 1*, $S = 180$, $V_{max} \geq 5$

B.3.2 Grupo *local 2*

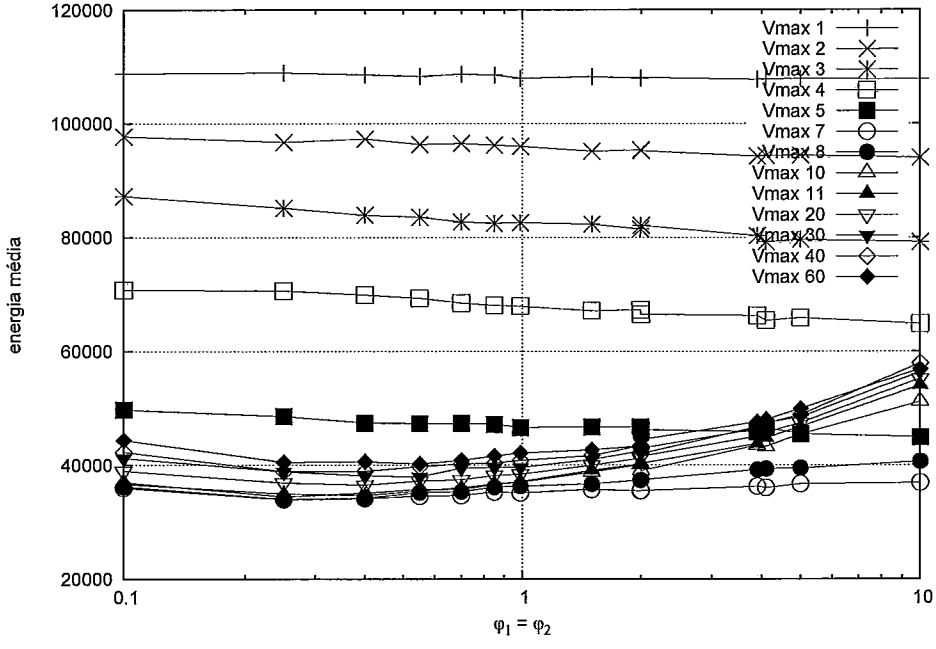

Figura B.26: Resultados: grupo *local 2*, $S = 180$, $N = 3$

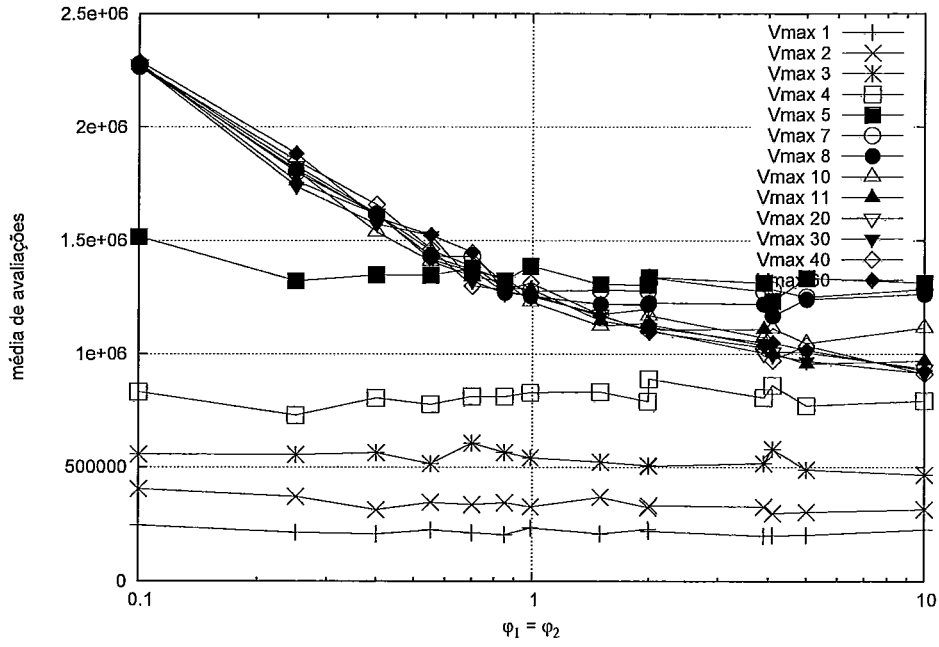

Figura B.27: Resultados: grupo *local 2*, $S = 180$, $N = 3$

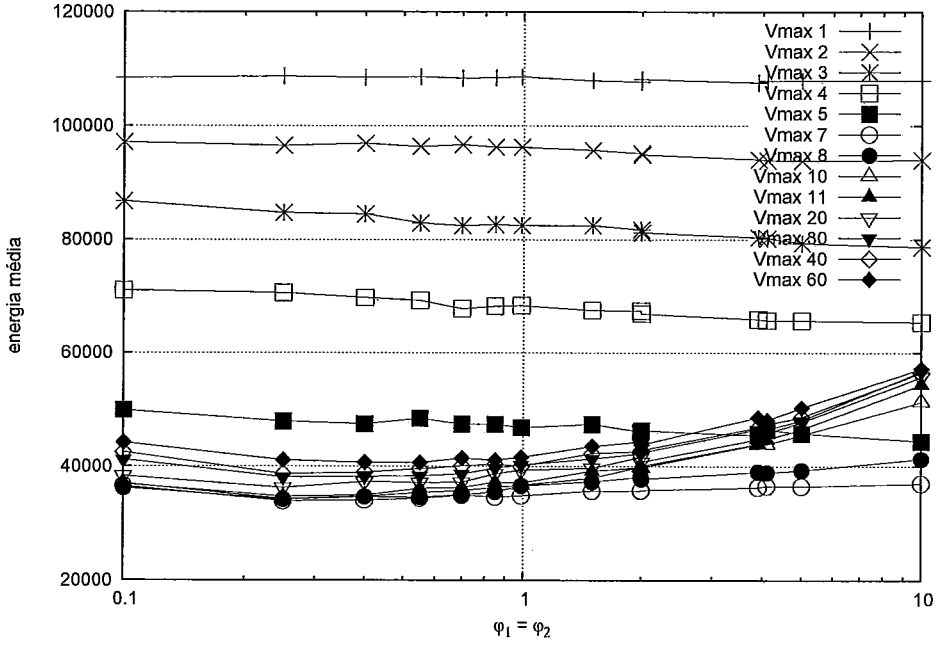

Figura B.28: Resultados: grupo *local 2*, $S = 180$, $N = 9$

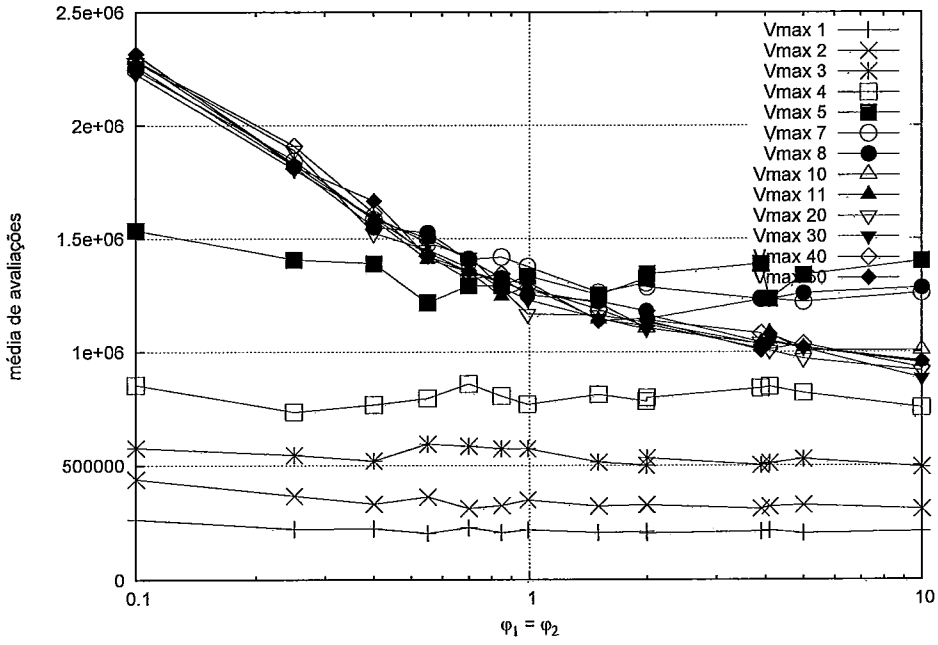

Figura B.29: Resultados: grupo *local 2*, $S = 180$, $N = 9$

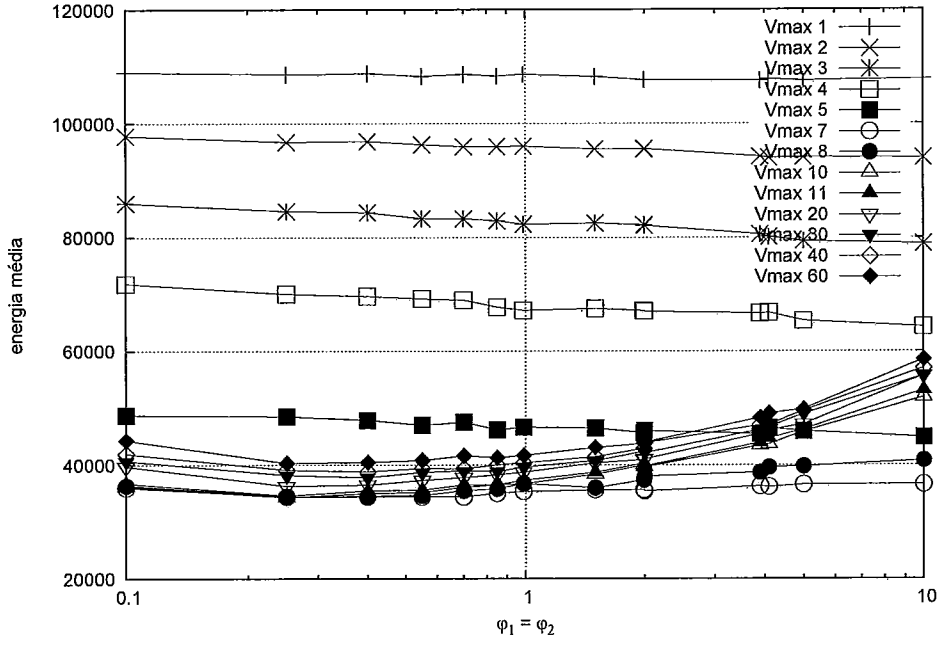

Figura B.30: Resultados: grupo *local 2*, $S = 180$, $N = 18$

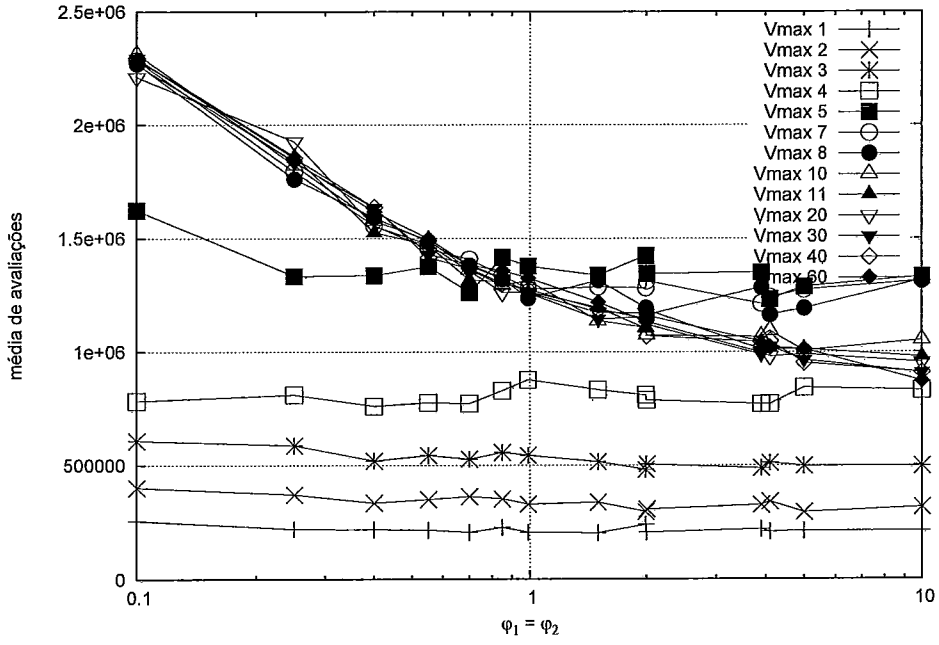

Figura B.31: Resultados: grupo *local 2*, $S = 180$, $N = 18$

B.3.3 Grupo *global 2*

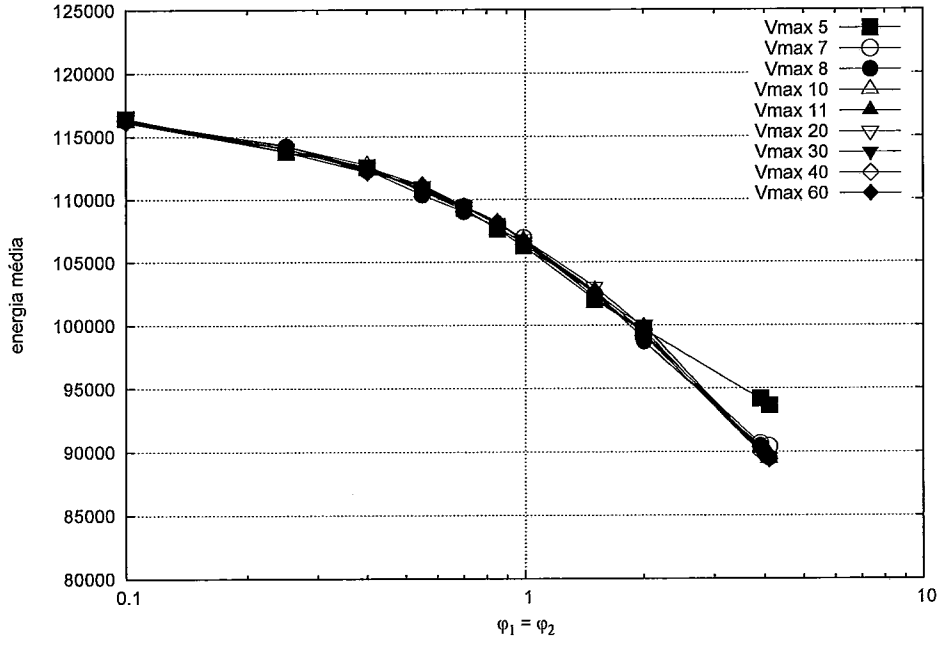

Figura B.32: Resultados: grupo *global 2*, $S = 180$, $\varphi_0 = 0.95$

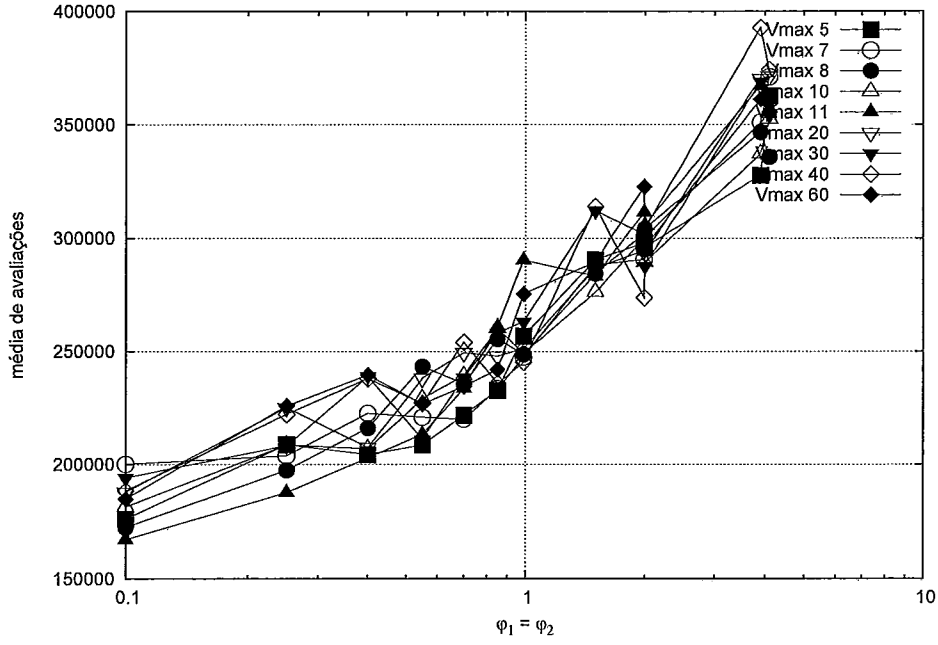

Figura B.33: Resultados: grupo *global 2*, $S = 180$, $\varphi_0 = 0.95$

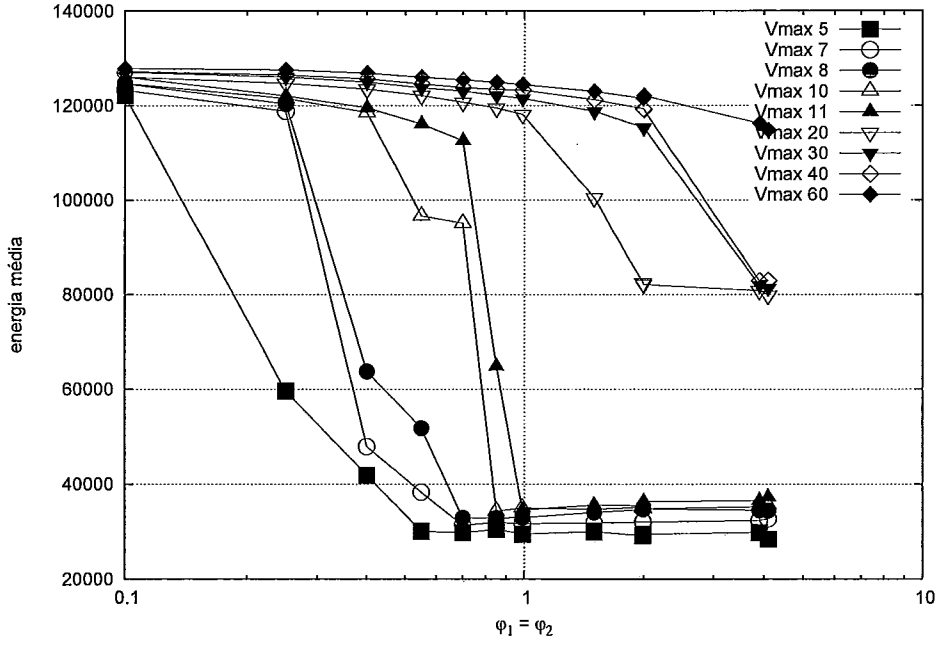

Figura B.34: Resultados: grupo *global 2*, $S = 180$, $\varphi_0 = 1.05$

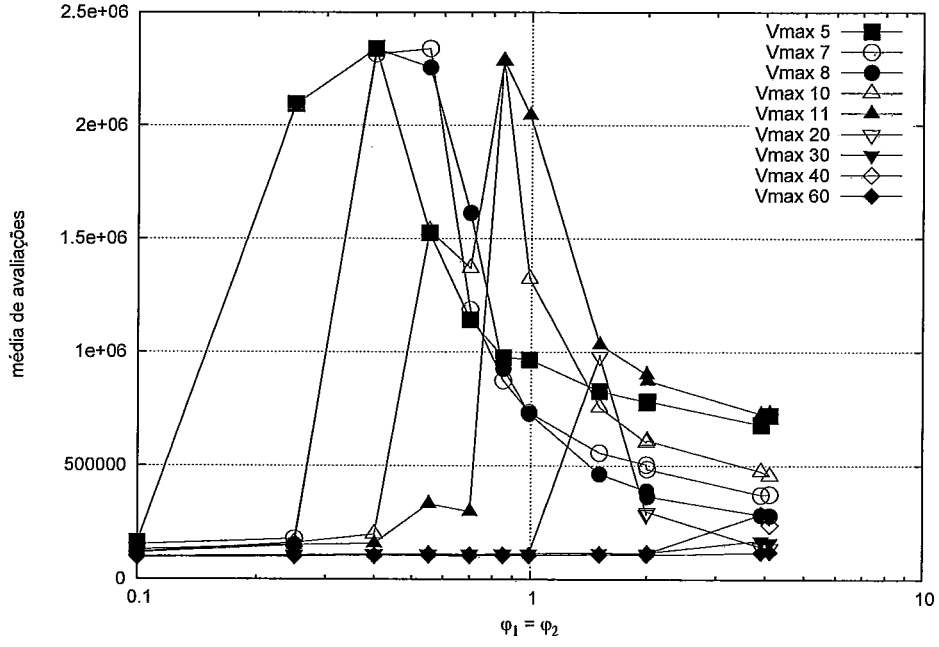

Figura B.35: Resultados: grupo *global 2*, $S = 180$, $\varphi_0 = 1.05$

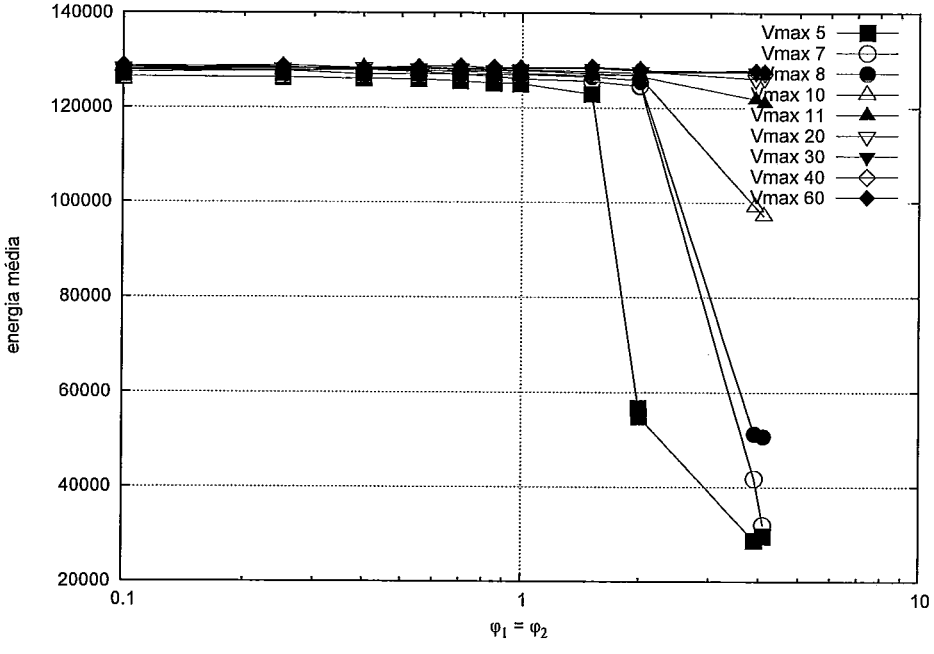

Figura B.36: Resultados: grupo *global 2*, $S = 180$, $\varphi_0 = 2$

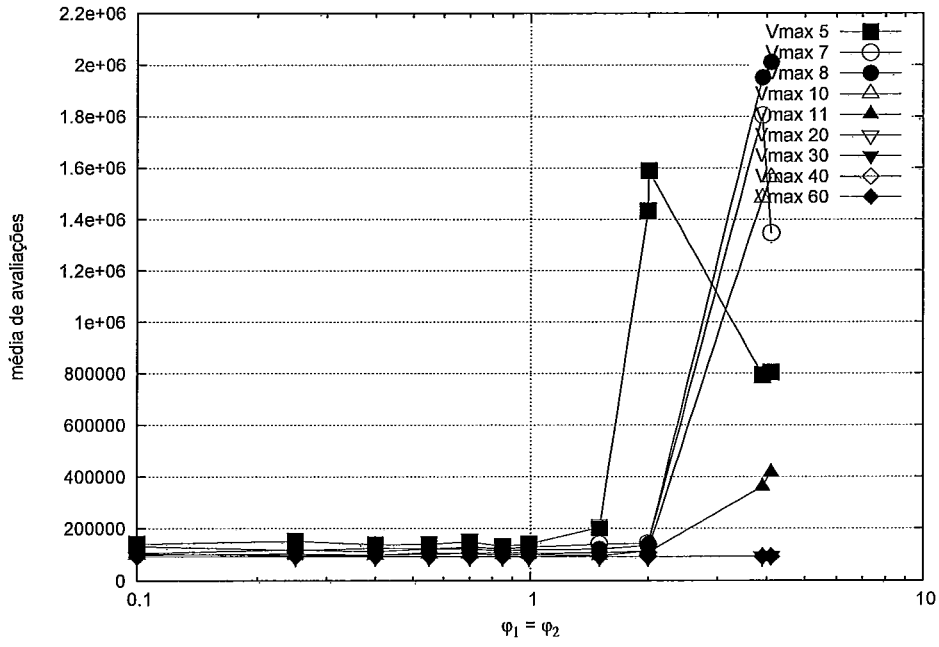

Figura B.37: Resultados: grupo *global 2*, $S = 180$, $\varphi_0 = 2$

B.3.4 Grupo *global 3*

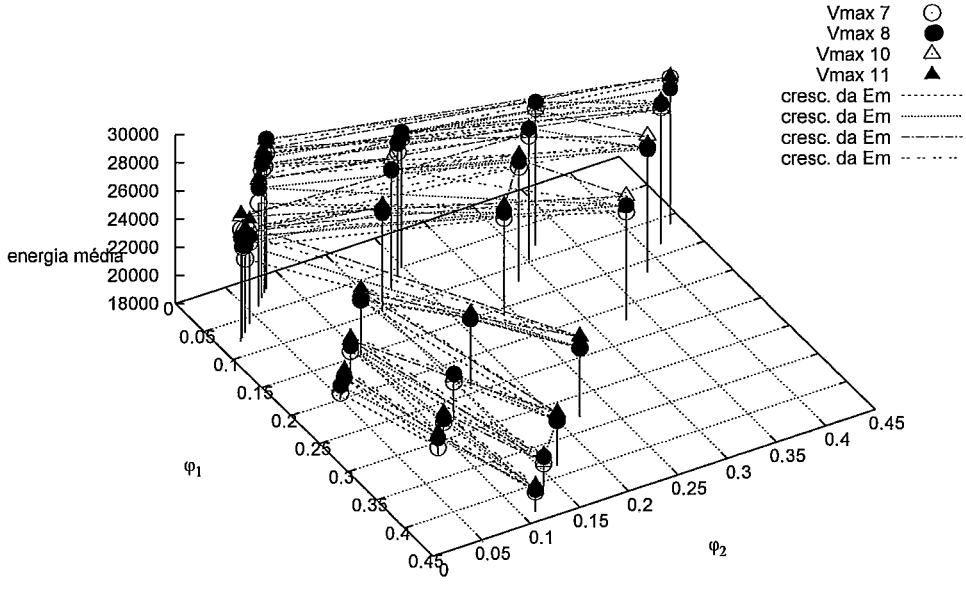

Figura B.38: Resultados: grupo *global 3*, $S = 180$

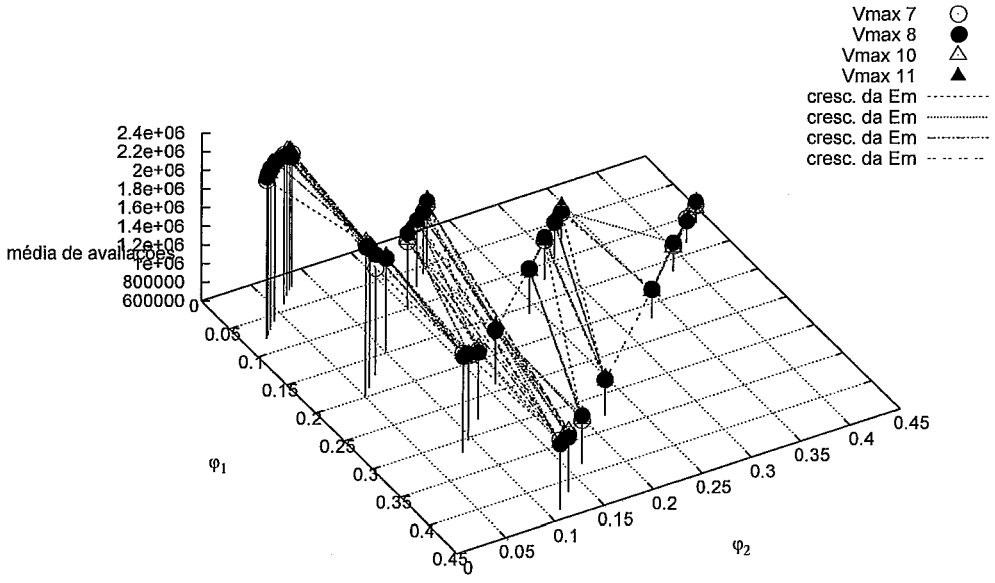

Figura B.39: Resultados: grupo *global 3*, $S = 180$

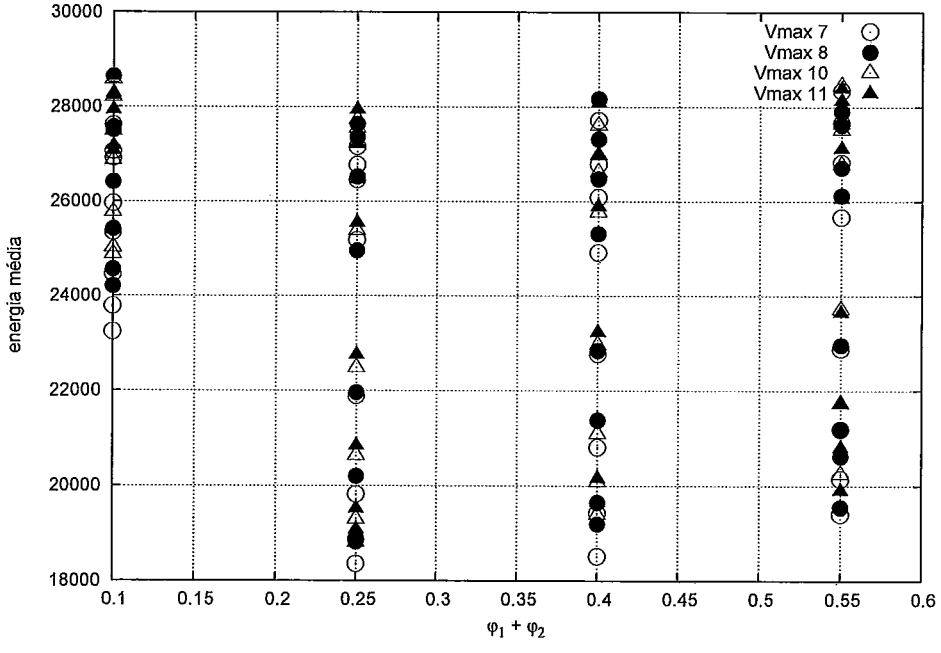

Figura B.40: Resultados: grupo *global 3*, $S = 180$, agrupados por φ

B.3.5 Grupos *wglobal 1* e *wglobal 2*

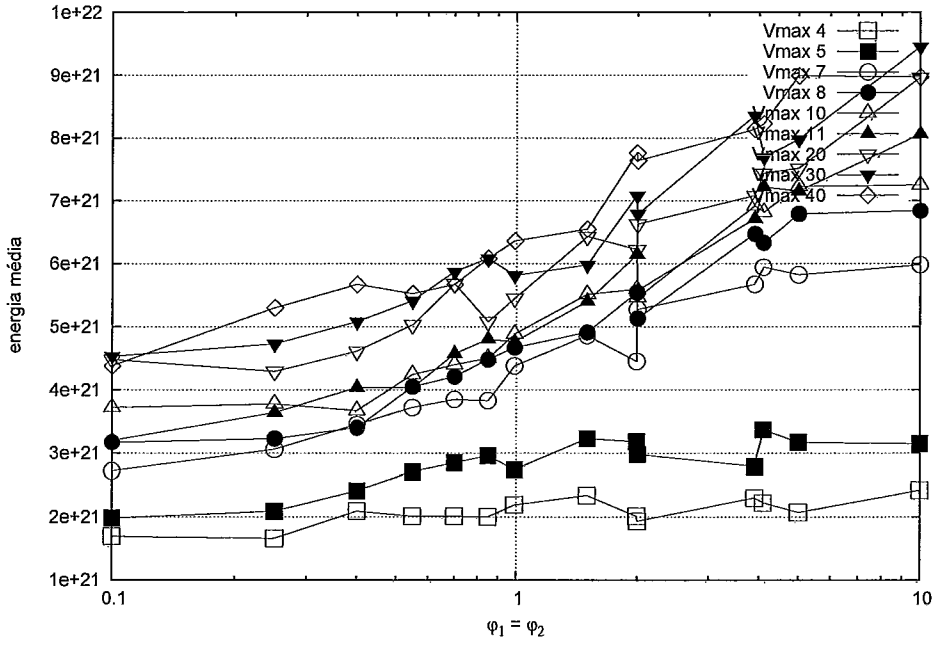

Figura B.41: Resultados: grupo *wglobal 1*, $S = 180$

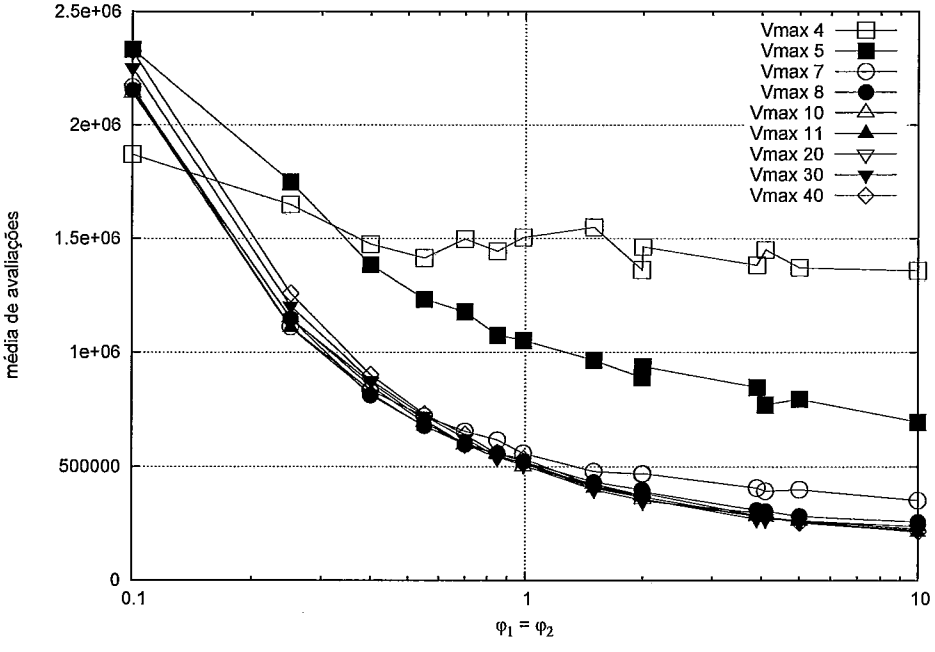

Figura B.42: Resultados: grupo *wglobal 1*, $S = 180$

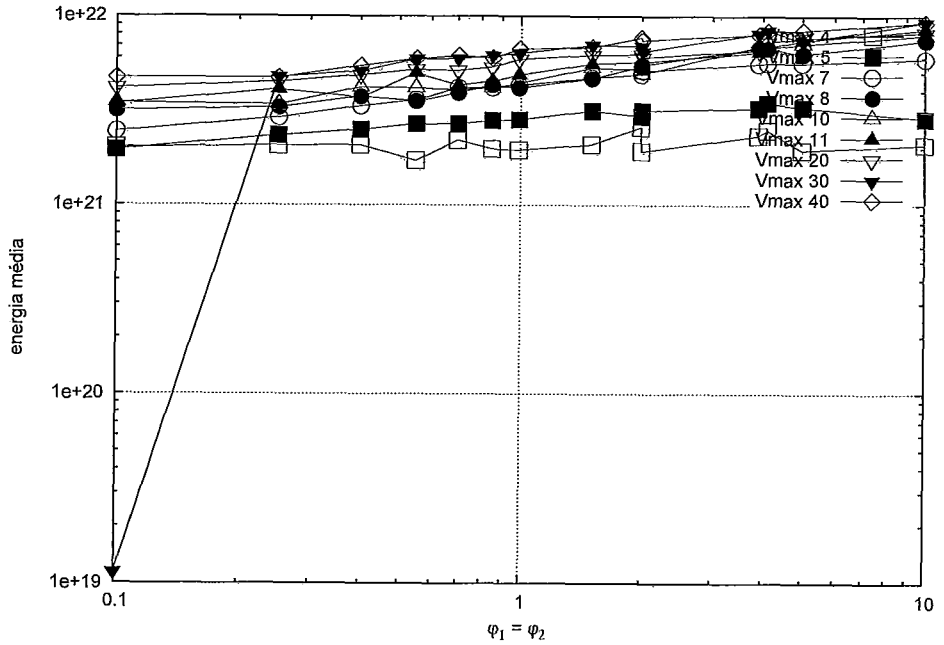

Figura B.43: Resultados: grupo *wglobal 2*, $S = 180$

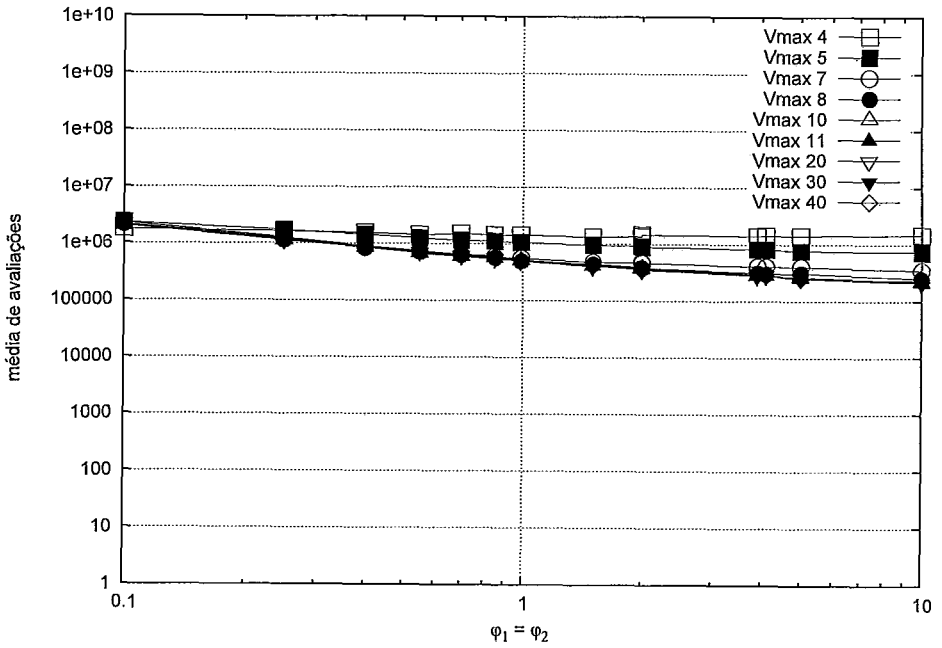

Figura B.44: Resultados: grupo *wglobal 2*, $S = 180$