GIOVANY FROSSARD TEIXEIRA

MULTIPLEX: UM PROCEDIMENTO BASEADO EM *SIMULATED ANNEALING* APLICADO AO PROBLEMA MAX-SAT PONDERADO

Orientadores: Arlindo Gomes Alvarenga
Attílio Provedel

Vitória – ES – 2006

SUMÁRIO

RESUMO

Computar a solução ótima para uma unidade de problema MAX-SAT Ponderado (*weighted maximum satisfiability*) é difícil mesmo se cada cláusula contiver apenas dois literais. Neste trabalho, será descrita a implementação de uma nova heurística aplicada a instâncias de problema do tipo MAX-SAT Ponderado, mas perfeitamente extensível a outros problemas. Para comparação, serão geradas soluções para uma quantidade significativa de problemas e seus resultados serão comparados com os de outras heurísticas já desenvolvidas para esse tipo de problema, dentre elas as heurísticas consideradas "estado da arte", ou seja, heurísticas que têm obtido os melhores resultados no universo das heurísticas existentes.

1. INTRODUÇÃO

O problema de restrições de satisfatibilidade (*constraint-satisfaction problem – CSP*) envolve um conjunto de variáveis e para cada uma delas um domínio de possíveis valores. Além disso, é aplicado um conjunto de restrições que determina se uma combinação de variáveis é válida ou não. Uma solução, então, pode ser definida como uma atribuição de valores para as variáveis que não viola as restrições de satisfatibilidade.

Um caso especial de problema de restrição de satisfatibilidade é o SAT, que consiste em determinar um conjunto de valores verdadeiro ou falso (*true or false*) para um conjunto de variáveis e que possuem como restrições de satisfabilidade conjunções de cláusulas. Uma cláusula é definida como uma disjunção de literais e um literal representa uma variável ou a negação de uma variável em uma cláusula.

No SAT, todas as cláusulas obrigatoriamente devem ser satisfeitas para que o problema seja resolvido. Uma variação desta abordagem é o MAX-SAT (*Maximum Satisfability Problem*) que consiste em buscar satisfazer a maior quantidade de cláusulas possível, entretanto é aceitável que alguma cláusula não seja satisfeita, pois existem problemas em que não existe possibilidade de satisfazer todas as cláusulas ao mesmo tempo. O MAX-SAT Ponderado (*Weighted Maximum Satisfatibility Problem*) consiste no MAX-SAT com atribuição de pesos para as cláusulas fazendo com que certas cláusulas passem a ter mais importância que outras.

MAX-SAT e MAX-SAT Ponderado são problemas NP-hard. Para uma visão mais detalhada, considerando os problemas SAT e MAX-SAT e seus algoritmos, ver Gu (1997). Uma das abordagens possíveis diante desse

contexto, consiste em utilizar meta-heurísticas como: *simulated annealing* (Kirkpatrick, et al. 1983), algoritmos genéticos (Goldberg, 1989), *tabu search* (Glover, 1989), colônia de formigas (Dorigo e Di Caro, 1999).

Neste trabalho é proposta a criação de um novo procedimento baseado em *Simulated Annealing* (Kirkpatrick, et al. 1983), aplicando-o ao problema MAX-SAT Ponderado, comparando seu desempenho com o de outras meta-heurísticas utilizadas para este tipo de problema como: ILS (Yagiura e Ibaraki, 1998; Lourenço, et al. 2001), GLS (Mills e Tsang, 2000), GRASP (Resende, 1996), S*imulated Annealing* (Kirkpatrick, et al. 1983). Os dois primeiros são considerados o "estado da arte" para esse tipo de problema.

O trabalho está organizado da seguinte forma: na seção 2 será feita uma análise do problema, destacando sua descrição para a programação linear. Na seção 3, será feita uma breve revisão bibliográfica das heurísticas que serão comparadas ao Multiplex. Na seção 4, será descrito o novo procedimento elaborado. Na seção 5, serão mostrados os resultados experimentais. Finalmente, na seção 6, serão apresentadas as conclusões obtidas, bem como novas possibilidades para futuras pesquisas.

2. O PROBLEMA MAX-SAT PONDERADO

Assuma que *B* é o alfabeto de uma linguagem proposicional ∟ . Fórmulas proposicionais, denotadas pelas letras gregas α, β, Φ, ..., são construídas pelo uso de variáveis proposicionais existentes no universo *Q* e por conectivos lógicos. Na lógica proposicional clássica, variáveis proposicionais p_1 ,..., p_i, ..., p_n podem assumir os valores *"true"* (verdadeiro) ou *"false"* (falso). Cada interpretação é uma atribuição de *"true"* ou *"false"* para cada uma das variáveis proposicionais. Pode-se usar *W* para denotar o conjunto de todas as interpretações. Um literal é uma variável proposicional p_i ou a negação de uma variável proposicional $\neg p_i$.

O problema de satisfabilidade (SAT) é um problema da lógica proposicional e o objetivo dele é determinar um conjunto de valores (*true* ou *false*) para as variáveis proposicionais que faça uma dada CNF (forma normal conjuntiva) satisfatível ou mostrar que nenhum conjunto é possível. Em outras palavras, o objetivo do SAT é encontrar uma interpretação que satisfaça à forma normal conjuntiva $\Phi = C_1 \wedge C_2 \wedge ... \wedge C_m$, com um conjunto de $S_\Phi = \{ C_j \mid j = 1, 2, ..., m\}$. Uma cláusula C_k pertencente a S_Φ pode ser definida como uma disjunção de literais, sendo representada da seguinte forma:

Seja a função:

$$I_k(v) = \begin{cases} \textit{false}, & \text{se a variável } v \text{ não aparece na cláusula } k; \\ \textit{true}, & \text{caso contrário.} \end{cases}$$

Então:

$$C_k = (I_k(v_1) \wedge L_1) \vee (I_k(v_2) \wedge L_2) \ldots V (I_k(v_i) \wedge L_i) \vee \ldots \vee (I_k(v_n) \wedge L_n)$$

Onde:

L_i *:* p_i *ou* ┐ p_i

v_i*: variável de índice i, com i = {1 ... n}*

n: número de variáveis

Exemplo 2.1

Um exemplo de problema 3-SAT, com sua solução, é mostrado na figura abaixo:

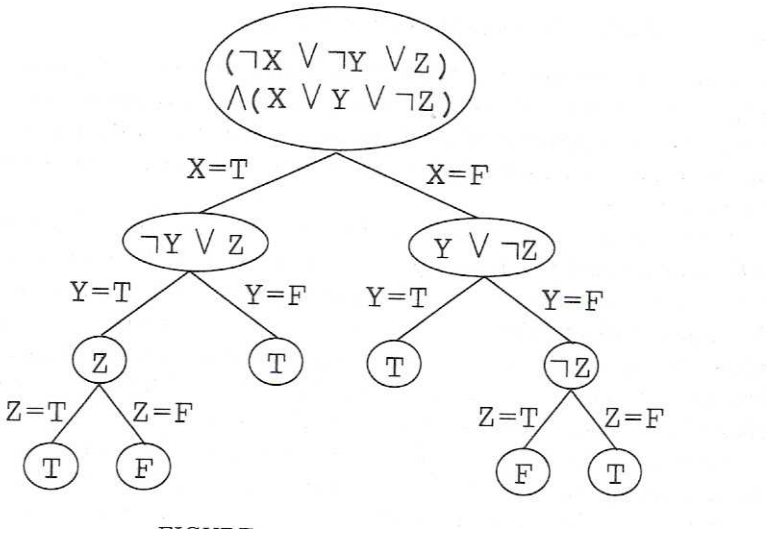

Figura 2.1 - 3-SAT

Neste exemplo, teríamos duas cláusulas:

┐ *x V* ┐ *y V z* (cláusula 1)

x V y V ┐ *z* (cláusula 2)

É feita então a montagem do grafo de possibilidades. Como *x* apareceu nas duas cláusulas, em uma cláusula negado e na outra não, então se *x* for T

(*true* – verdadeiro) a cláusula 2 é satisfeita, caso contrário a cláusula 1 será a satisfeita. O próximo passo é atribuir os valores possíveis T (*true* – verdadeiro) e F (*false* – falso) para as variáveis *y* e *z*. Ao final teremos o grafo de possibilidades, os nós folha que terminaram com o valor F indicam que o problema não foi satisfeito e os nós folha terminados em T indicam que o problema foi satisfeito. Desta forma, podemos dizer que o caminho $x = T, y = T$ *e* $z = T$ é solução para o problema e, $x = T, y = T$ *e* $z = F$ não é solução para o problema.

O problema SAT é o núcleo de uma classe ampla de problemas NP – completos. Na verdade o problema SAT foi um dos primeiros problemas provado ser NP – completo (Resende, 1996). O problema MAX-SAT está intimamente relacionado ao problema SAT, e informalmente pode ser definido como: dada uma coleção de cláusulas, determinar uma interpretação (atribuição de valores) que maximize o número de cláusulas satisfeitas. O problema MAX-SAT Ponderado é uma unidade do problema MAX-SAT que atribui um peso a cada cláusula. Serão denotados esses pesos por $k_1, ..., k_j, ..., k_m$, onde k_j é o peso da cláusula C_j, para $j = 1, ..., m$.. Nosso objetivo é determinar uma interpretação ou atribuição que maximize a soma dos pesos das cláusulas satisfeitas.

2.1. FORMULAÇÃO DO PROBLEMA MAX-SAT

Segundo Resende (1996), o problema MAX-SAT Ponderado é um problema de programação linear que pode ser descrito da seguinte forma:

$$max\ F(y, z) = \sum_{i=1}^{m} w_i\, z_i$$

Sujeito a:

$$\sum_{j \in \Gamma^{+}_{i}} y_j + \sum_{j \in \Gamma_i} (1 - y_j) \geq z_i \qquad i = 1 \ldots m$$

$$y_j \in \{0, 1\} \qquad j = 1 \ldots n$$

$$0 \leq z_i \leq 1 \qquad i = 1 \ldots m$$

Onde:

Γ^{+}_{i} : Conjunto das variáveis da cláusula i (literais) que aparecem não negadas.

Γ_i : Conjunto das variáveis da cláusula i que aparecem negadas.

y_j : Literal j da cláusula i.

z_i: Indicador se a cláusula i foi satisfeita ou não.

m: Número de cláusulas.

n: Número de variáveis.

w_i : Peso da cláusula i.

Na busca para resolver o problema MAX-SAT foram elaborados algoritmos aproximativos. Também, segundo Resende (1996), tais algoritmos garantem encontrar no mínimo certa proporção da solução ótima para qualquer problema MAX-SAT. O primeiro algoritmo de aproximação foi elaborado por Johnson (1974), em que um algoritmo de 50% de aproximação era apresentado, ou seja, por pior que fosse a unidade de problema, tal algoritmo garantia uma solução, no mínimo, 50% do ótimo; ter 50% do ótimo significa que o algoritmo encontrará uma solução cujo valor será, no mínimo, metade do valor da solução ótima. Novos algoritmos de aproximação foram elaborados chegando a quase 76% do ótimo para problemas MAX-SAT (Yannakakis, 1992; Goemans e Williamson, 1994) e aproximadamente 93 % do ótimo para problemas MAX-2SAT, problemas nos quais é garantido existirem exatamente dois

literais por cláusula (Feige e Goemans, 1995). Desta forma, pode-se afirmar que para problemas do tipo MAX-2SAT o “estado da arte” de algoritmos de aproximação é bastante satisfatório, entretanto, no que tange os problemas MAX-SAT Ponderado, os algoritmos de aproximação ainda não garantem qualidade de solução acima dos 76% do ótimo, sendo assim, torna-se bastante justificável o uso de algoritmos heurísticos. O “estado da arte” para algoritmos heurísticos aplicados ao MAX-SAT são o ILS (*Iterated Local Search*), o GLS (*Guided Local Search*) e variantes do SA (*Simulated Annealing*), portanto, esses algoritmos serão os algoritmos usados para a comparação com a proposta que será abordada nos capítulos seguintes. Além dos algoritmos chamados “estado da arte” foi acrescido à comparação o GRASP (*Greedy Randomized Adaptative Search Procedure*) por ser um algoritmo tradicionalmente aplicado ao MAX-SAT Ponderado e que encontra resultados consideráveis.

3. ABORDAGENS PARA O PROBLEMA

A análise de desempenho de heurísticas pode ser bastante complexo, uma vez que, não se conhecendo o valor ótimo, não há nenhuma garantia da qualidade da solução encontrada. Em vista disso, torna-se importante submeter a heurística proposta a uma significativa base de testes experimentais. Comparando, quando possível, os resultados desta nova heurística com os de outras pré-existentes.

Neste tópico, serão discutidas as heurísticas que serão utilizadas para comparação com o algoritmo Multiplex, que é a nova heurística proposta.

3.1.*GREEDY RANDOMIZE ADAPTATIVE PROCEDURE* - GRASP

Greedy Randomize Adaptative Procedure (Resende, 1996) é um procedimento iterativo *multistart* em que cada iteração é dividida em duas fases:

A **Fase de Construção** é caracterizada por um algoritmo iterativo, adaptativo, randômico e guloso. Nesta etapa uma solução é iterativamente construída, um elemento por vez, onde a ordem de escolha dos elementos é determinada por uma função de construção gulosa.

Por outro lado, na **Fase de Busca Local**, novas soluções são geradas a partir de uma solução inicial através de um algoritmo de busca local. Os algoritmos de busca local armazenam apenas o estado atual (baixo uso de memória), e não vêem além dos vizinhos imediatos do estado, contudo

muitas vezes são os melhores métodos para tratar problemas reais complexos (espaço contínuo).

Podemos então resumir o algoritmo GRASP da seguinte forma:

```
Algoritmo GRASP (RCLSize, num_iteracoes_max, random_seed)
{
    Ler Dados ()
    Inicializar Estruturas de Dados ()
    Melhor Solução Encontrada = 0
    Para k = 1 .. num_iteracoes_max faça
    {
        Algoritmo de Construção (RCLSize, random_seed)
        Algoritmo de Busca Local (Melhor Solução Encontrada)
    }
          Retorna Melhor Solução Encontrada
}
```

Algoritmo 3.1.1 - GRASP

```
Algoritmo de Construção (RCLSize, random_seed)
{
    Para k = 1 .. num_variáveis faça
    {
        FazerRCL (RCLSize)
        s = Selecione um índice (random_seed)
        Setar a variável de índice s para "true" ou "false"
        Função Adaptação Gulosa(s)
    }
}
```

Algoritmo 3.1.2 – Algoritmo de Construção

Nos algoritmos 3.1.1 e 3.1.2, RCLSize define a lista de escolhas para atribuição de variáveis, no caso do MAX-SAT, especificamente, essa lista é definida por {0, 1}. A função de adaptação gulosa impõe restrições à montagem da lista RCL (de possíveis valores para uma dada variável) na iteração subseqüente. O algoritmo de busca local pode ser construído de

diversas formas, uma vez que, o único parâmetro passado é a melhor solução encontrada até então. Uma possível abordagem seria utilizar algoritmos do tipo 1-flip, 2-flip, ver Yagiura e Ibaraki (1998) para mais detalhes.

3.2. *ITERATED LOCAL SEARCH* - ILS

Iterated Local Search (Yagiura e Ibaraki, 1998; Lourenço, et al. 2001) é um dos algoritmos considerados "estado da arte" no que tange a algoritmos de busca local para o problema MAX-SAT Ponderado. A idéia básica consiste em aplicar uma perturbação na melhor solução encontrada até um dado momento, aplicar ao resultado dessa perturbação um algoritmo de busca local e finalmente aplicar uma função de avaliação de aceitação. O algoritmo ILS pode, então, ser ilustrado da seguinte forma:

```
Algoritmo ILS( num_iteracoes_max)
{
    S0 = Gere uma solução inicial
    S* = Aplique algum algoritmo de busca local( S0)
    Para i = 1 .. num_iteracoes_max faça
    {
        S´ = Perturbação( S*, histórico)
        S*´ = Aplique algum algoritmo de busca local(S´)
        S* = Aplique algum critério de Aceitação (S*, S*´, histórico)
    }
}
```

Algoritmo 3.2.1 - ILS

A melhor forma de elaboração/calibração das funções de Perturbação e do critério de aceitação e a forma de atualização do histórico, definem o quão eficiente pode se tornar o ILS para um dado problema.

3.3. *GUIDED LOCAL SEARCH* - GLS

Guided Local Search (Mills e Tsang, 2000), assim como o ILS, é um dos algoritmos considerados "estado da arte" no que tange a algoritmos de busca local. Para seu funcionamento, o GLS define o conceito de **características**, de forma que, características consideradas desinteressantes geram **penalização** da solução. A melhor escolha das características depende do tipo de problema e também deve ser analisada no uso do algoritmo de busca local internamente utilizado no GLS.

Cada característica *fi* definida deve ter os seguintes componentes:

- Uma função de indicação, mostrando se tal característica está presente ou não:

$$I_{fi}(s) = \begin{cases} 1, \text{ se a característica } fi \text{ está presente na solução } s; \\ 0, \text{ caso contrário.} \end{cases}$$

- Uma função de custo $c_{fi}(s)$ que define o custo da solução *s* com característica *fi*.

- Uma penalidade p_{fi}, inicialmente inicializada para zero, mas usada para penalizar ocorrências de característica *fi*, em um mínimo ou máximo local.

Se o algoritmo de busca local retornar, no caso de um problema de minimização, um mínimo local, o GLS penaliza todas as características em que a solução tiver máxima utilidade, sendo esta definida da seguinte forma:

$Util(s, fi) = c_{fi}(s) / (1 + p_{fi})$

A idéia básica é primeiro penalizar características de maior custo, mas com o aumento da penalização de uma dada característica de maior custo, a utilidade será decrementada para essa característica possibilitando que outras características sejam penalizadas.

O GLS utiliza uma função para sair de mínimos locais baseada nas penalizações. A idéia básica é penalizar o mínimo local em relação a sua vizinhança, possibilitando assim que o GLS saia de um mínimo local. A expressão desta função é a seguinte:

$h(s) = g(s) + K * \sum p_{fi} * I_{fi}(s)$

Nesta expressão, temos que:

- *K* é uma variável de calibração, quanto maior, maior a diversidade de soluções;
- *g(s)* é o valor da solução corrente;
- *h(s)* é o valor da solução aplicando penalizações.

Mills e Tsang (2000) definem para o Problema SAT, que características são violações de cláusulas. Dessa forma:

$$I_{fi}(s) = \begin{cases} 1, \text{ se a cláusula não é satisfeita;} \\ 0, \text{ caso contrário.} \end{cases}$$

$c_{fi}(s) = 1$ (no problema MAX SAT Ponderado, $c_{fi}(s) = w_i$).

O Algoritmo GLS para o SAT, então, pode ser escrito de maneira simplificada da seguinte forma:

Algoritmo GLS(s, g, K, num_iter_int),
{
s = Gerar Solução Inicial
Inicializar Variáveis (for each fi, $p_{fi} = 0$)
Faça
{
$h(s) = g(s) + K * \sum p_{fi} * I_{f\,i}(s)$
s = LocalSATSearch(s, h, num_iter_int)
Para cada característica de fi (cláusula não satisfeita) de máxima utilidade, Útil(s, fi)
$p_{fi} = p_{fi} + 1$
} Enquanto h(s) > 0
}

Algoritmo 3.3.1 - GLS

O módulo LocalSATSearch pode ser definido da seguinte forma:

***Function** LocalSATSearch(**s**, g, smax)*
{
Enquanto (g(s) > 0)
{
Modifique s, pela troca da variável menos recentemente modificada, a qual decrementará a função objetivo g(s), se alguma existir.
Senão (se nenhuma variável que possa decrementar a função existir), trocar a variável menos recentemente modificada que faz com que a função objetivo não tenha seu valor acrescido nem decrescido.
Se o número de trocas feito igual à smax então retornar s
}
Retornar s
}

Algoritmo 3.3.2 - LocalSATSearch

Não é objetivo deste trabalho discutir aspectos de otimização do GLS. Assim, para mais informações sobre otimização do GLS aplicado ao SAT, ver Mills e Tsang (2000).

3.4. *SIMULATED ANNEALING* - SA

O algoritmo de ***Simulated Annealing*** (Kirkpatrick, et al. 1983) é uma evolução do método de melhoramento iterativo. A técnica de melhoramento iterativo consiste em, dada uma solução inicial, gerar soluções vizinhas até o ponto que não exista nenhuma solução vizinha de custo menor que a melhor solução encontrada até então, isto, para o caso do algoritmo estar sendo aplicado para uma minimização. O problema da técnica de melhoramento iterativo é a grande possibilidade de estagnação em um mínimo local. A figura 3.4.1 ilustra o problema dos mínimos locais. Se o algoritmo de melhoramento iterativo fosse aplicado em X_0 a melhor solução seria A, que é um mínimo local. Se o algoritmo fosse aplicado em X_0´ o mínimo seria encontrado em B, que é o mínimo efetivo.

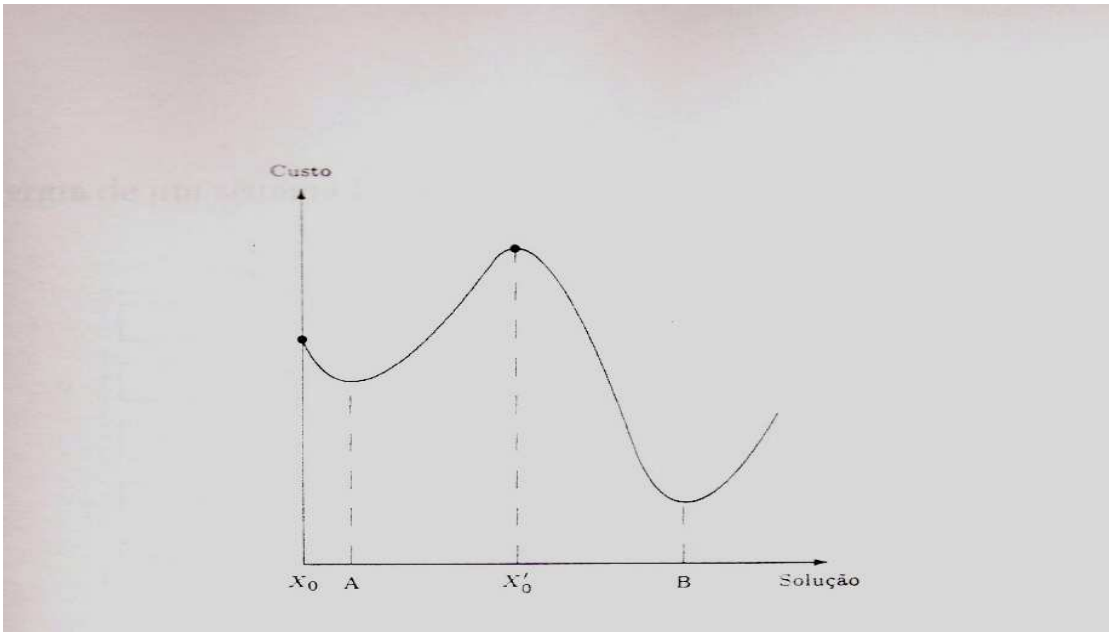

Figura 3.4.1 – Gráfico de Mínimo Local

O *Simulated Annealing* é um algoritmo de melhoramento iterativo que cria um mecanismo de saída de mínimos locais, portanto, se aplicado em X_0, na figura 3.4.1 seria possível que ele conseguisse encontrar B.

O *Simulated Annealing* foi proposto como um método para minimizar funções de variáveis e, em sua concepção, estabelece uma analogia com o processo físico de resfriamento descrito pela mecânica estatística.

Na mecânica estatística, temperaturas baixas não são uma condição suficiente para encontrar estados fundamentais (de baixa energia) de sistemas. Experimentos que determinam o estado de baixa temperatura de um sistema que forneça energia mínima são feitos por um método chamado recozimento (*annealing*). Resumidamente, o recozimento é a redução gradativa da temperatura de forma a montar um sistema de baixa temperatura. A redução rápida da temperatura pode ocasionar sistemas de alta energia devido a falta de tempo para equilíbrio térmico. A figura 3.4.2 ilustra a analogia do sistema mecânico descrito com a abordagem simulada do *Simulated Annealing*.

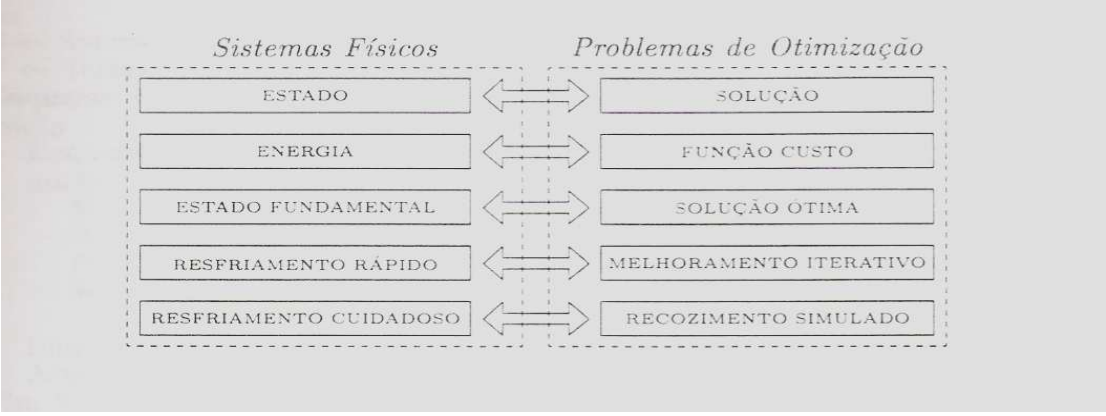

Figura 3.4.2 – Comparação do Problema Físico com o Algoritmo *Simulated Annealing*

O Algoritmo *Simulated Annealing* pode ser descrito da seguinte forma:

```
Algoritmo Simulated Annealing
{
    S0 = Gerar Solução Inicial
    T = temperatura inicial
    Enquanto critério de parada não satisfeito faça
    {
        Enquanto não encontrar o equilíbrio faça
        {
            S´ = Gerar Solução Vizinha de S
            Δ = Custo ( S´ ) – Custo ( S )
            Se Δ < 0 faça S = S´
            Senão
                Prob = Min (1, e^(−Δ/T))
                Se (randômico(0, 1) ≤ Prob) faça
                    S = S´
        }
        Atualizar T aplicando uma taxa de decrescimento
    }
    Retorna S
}
```

Algoritmo 3.4.1 – *Simulated Annealing*

4. MULTIPLEX: UM PROCEDIMENTO BASEADO EM *SIMULATED ANNEALING*

O Multiplex é um algoritmo altamente influenciado pelo Simulated Annealing. Pode-se destacar as seguintes características presentes no *Simulated Annealing* que formam base para o Multiplex:

- É gerada uma solução inicial (ou várias). Na realidade, esta é uma característica presente em praticamente todas as heurísticas construtivas;
- É aplicado algum algoritmo de geração de solução vizinha;
- É aplicado algum critério para se determinar que solução deva ser escolhida, a original ou a vizinha gerada.

4.1. VISÃO GERAL DA TÉCNICA

Basicamente o algoritmo gera um conjunto de soluções iniciais. Divide essa população de maneira ordenada em divisões de população, cada divisão então, possui uma parcela ordenada da população e não há interseção de soluções entre as divisões. As unidades de processamento são responsáveis pela geração de soluções novas, toda unidade de processamento está sempre associada a uma e apenas uma divisão em um instante qualquer, mas elas podem variar em duas formas:

- Unidades de Processamento Fixas: Pertencem a divisão e independente do momento do algoritmo permanecerão associadas a mesma divisão.

- Unidades de Processamento Móveis: podem migrar entre as divisões premiando as divisões que as recebem e penalizando as divisões que as perdem.

Cada divisão de maneira independente trabalha apenas em sua parcela da população, gerando soluções vizinhas às soluções ou atribuições que tem acesso. A escolha das atribuições para geração de vizinhas pode ser feita de maneira completamente aleatória. Faz-se então a avaliação da evolução da divisão. Caso a divisão tenha evoluído, ela recebe Unidades de Processamento Móveis de suas vizinhas mais próximas, se elas possuírem alguma, valorizando assim a evolução da divisão. Caso a divisão não tenha evoluído, ela é penalizada cedendo Unidades de Processamento Móveis para suas vizinhas mais próximas, se ela possuir Unidades de Processamento Móveis para ceder. Dessa forma busca-se aumentar o número de gerações de vizinhos em regiões que sejam mais promissoras.

Para determinar se a divisão deve manter a atribuição corrente ou ficar com uma nova vizinha gerada é utilizado o critério de escolha feito no *Simulated Annealing*, que toma como base a temperatura corrente e uma relação entre o valor da função de qualificação da atribuição corrente e da vizinha gerada.

Quando todo conjunto das divisões termina de gerar as atribuições vizinhas é aplicado um decréscimo na temperatura de forma a fazer o recozimento do *Simulated Annealing*. Para uma melhor convergência foi utilizada uma taxa variável de queda da temperatura, de forma que, quanto menor a temperatura mais lentamente ela decresce, fazendo com que o algoritmo permaneça mais tempo em temperaturas menores.

4.2. DESCRIÇÃO DA HEURÍSTICA

Neste tópico, discutiremos os aspectos relativos a implementação da metaheuristica Multiplex para o problema MAX-SAT Ponderado.

- **Geram-se as soluções iniciais, promovendo em seguida sua ordenação**

É utilizado um algoritmo para geração de soluções iniciais. Cada solução possui um valor específico que será denominado *rank*, valor este que define a qualidade de uma dada solução. De posse deste valor é possível promover uma ordenação entre as soluções iniciais geradas.

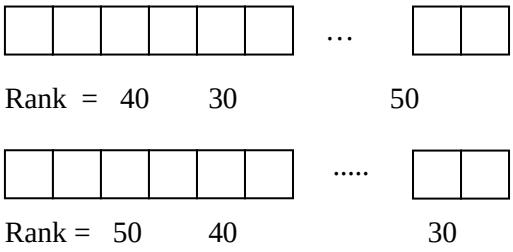

Figura 4.2.1 – Geração / Ordenação das Soluções Iniciais

- **Divide-se as atribuições em divisões de tamanho previamente estabelecido**

Suponha que se deseje ter 5 divisões: a primeira com 5 atribuições, a segunda com 4, a terceira com 3, a quarta com 2 e a quinta com uma única solução. A determinação do tamanho de cada divisão é parâmetro de entrada, possibilitando assim, uma melhor adaptação para cada tipo de problema.

Como as divisões estão ordenadas teríamos algo como:

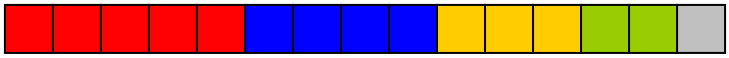

Figura 4.2.2 – Esquematização das divisões

É importante ressaltar que a quantidade de atribuições na população é igual à soma das quantidades de atribuições de cada divisão, ou seja, não existe uma atribuição sem divisão.

- **Determinação da quantidade de unidades de processamento fixas de cada divisão**

A quantidade de unidades de processamento fixas em uma divisão é um parâmetro de entrada. Sua quantidade não deve ser muito grande, isso porque seu objetivo é garantir que em um momento qualquer existe alguma unidade de processamento numa dada divisão.

Cada unidade de processamento é responsável por gerar uma solução vizinha de uma dada solução da divisão, tal escolha pode ser feita de maneira completamente aleatória.

Suponhamos a seguinte distribuição de unidades de processamento fixas:

1ª Divisão = 2 unidades de processamento fixas
2ª Divisão = 1 unidade de processamento fixa
3ª Divisão = 0 unidade de processamento fixa
4ª Divisão = 0 unidade de processamento fixa
5ª Divisão = 1 unidade de processamento fixa

Isso significa que em qualquer momento do algoritmo existem pelo menos 2 unidades de processamento na 1ª Divisão, 1 unidade de processamento na 2ª Divisão e assim sucessivamente. As 3ª e 4ª divisões, como possuem 0 unidade de processamento fixa, podem num dado instante, ser regiões não pesquisadas para geração de novas soluções vizinhas, sendo assim, é importante calibrar o algoritmo de forma que ele obtenha os melhores resultados possíveis.

- **Determinação da quantidade de unidades de processamento móveis de cada divisão**

A quantidade de unidades de processamento móveis iniciais é um parâmetro do algoritmo, entretanto as unidades de processamento móveis como o próprio nome diz, ‘passeiam’ entre as divisões, indo sempre para regiões que se mostram mais promissoras. Pode-se fazer uma analogia aos garimpeiros que buscam ouro. Quando uma mina se mostra promissora a maior parte deles tenta achar ouro nessa mina; se forem duas minas promissoras eles devem se dividir. Da mesma forma, se uma mina se mostra exaurida os garimpeiros tentem a buscar outras regiões. É nesta lógica que o algoritmo se baseia, valorizações de divisões promissoras e desvalorização de divisões que se mostram pouco produtivas.

Cada divisão utiliza suas unidades de processamento buscando encontrar novas soluções vizinhas melhores que as atuais. Ao fim desse processo faz-se a avaliação a fim de se determinar se houve ou não evolução na divisão, ou seja, se esta divisão é promissora ou não. Caso a divisão seja considerada promissora, ela recebe unidades de processamento móveis das

divisões vizinhas, caso não seja, ela cede unidades móveis às divisões vizinhas.

No caso especifico da primeira e da última divisões, ocorre o seguinte:

- Se a primeira divisão evoluiu, ela recebe unidades de processamento da divisão posterior a ela e da última divisão existente, se a primeira divisão não evoluiu ela sede unidades de processamento para a divisão posterior e para a última. Da mesma forma, se a última divisão evoluir, ela recebe unidades de processamento da sua antecessora e da primeira divisão, se ela não evoluir ela sede unidades de processamento para a antecessora e para a primeira.

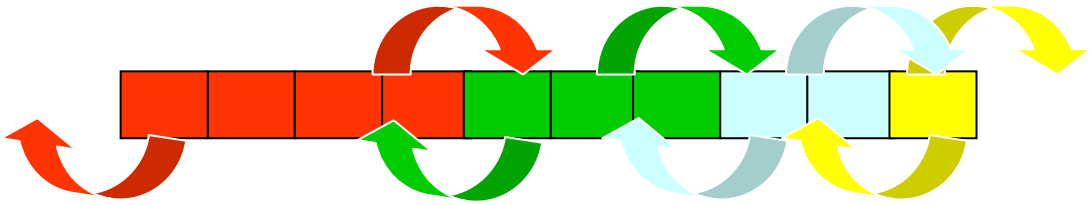

Figura 4.2.3 – Transferência de unidades de processamento móveis

A figura 4.2.3 ilustra a transferência das unidades de processamento móveis. È importante notar o comportamento de lista circular entre as divisões, pois é como se a primeira e a última divisões estivessem ligadas.

A ordem de geração de vizinhas é da primeira para última divisão. As atribuições também estão ordenadas em ordem decrescente de rank. O exemplo abaixo ilustra esse comportamento:

Exemplo 4.2.1

Considere que foram fornecidos como parâmetros de entrada, as seguintes informações:

- O algoritmo utilizará 5 divisões e 20 soluções distribuídas da seguinte forma,:
    - A primeira divisão terá 5 soluções, 2 unidades de processamento fixas e 10 unidades de processamento móveis.
    - A segunda divisão terá 4 soluções, 1 unidade de processamento fixa e 8 unidades de processamento móveis.
    - A terceira divisão será composta por 3 soluções, não terá unidades de processamento fixas e terá 10 unidades de processamento móveis.
    - Na quarta divisão haverá 2 soluções, nenhuma unidade de processamento fixa e 8 unidades de processamento móveis.
    - Na quinta e última divisão existirá apenas uma solução, que será trabalhada por 1 unidade de processamento fixa e 4 unidades de processamento móveis.

Esquematicamente pode-se ver a estrutura de divisão das soluções na figura 4.2.2.

Suponha que a 1ª Divisão tenha evoluído, assim teríamos o novo quadro:

1ª) Divisão: 2 up (unidades de processamento) fixas + 12 up móveis.
2ª) Divisão: 1 up fixa + 7 up móveis.
3ª) Divisão: 10 up móveis.
4ª) Divisão: 8 up móveis.

5ª) Divisão: 1 up fixa + 3 up móveis.

Suponhamos também que a 2ª Divisão não tenha evoluído, então:

1ª) Divisão: 2 up (unidades de processamento) fixas + 13 up móveis.
2ª) Divisão: 1 up fixa + 5 up móveis.
3ª) Divisão: 11 up móveis.
4ª) Divisão: 8 up móveis.
5ª) Divisão: 1 up fixa + 3 up móveis.

Se a 3ª Divisão evoluir verificar-se-á o seguinte:

1ª) Divisão: 2 up (unidades de processamento) fixas + 13 up móveis.
2ª) Divisão: 1 up fixa + 4 up móveis.
3ª) Divisão: 13 up móveis.
4ª) Divisão: 7 up móveis.
5ª) Divisão: 1 up fixa + 3 up móveis.

Da mesma forma se a 4ª Divisão evoluir ter-se-á:

1ª) Divisão: 2 up (unidades de processamento) fixas + 13 up móveis.
2ª) Divisão: 1 up fixa + 4 up móveis.
3ª) Divisão: 12 up móveis.
4ª) Divisão: 9 up móveis.
5ª) Divisão: 1 up fixa + 2 up móveis.

E finalmente se a 5ª Divisão não evoluir dar-se-á o seguinte:

1ª) Divisão: 2 up (unidades de processamento) fixas + 14 up móveis.

2ª) Divisão: 1 up fixa + 4 up móveis.
3ª) Divisão: 12 up móveis.
4ª) Divisão: 10 up móveis.
5ª) Divisão: 1 up fixa + 0 up móvel.

Deve-se notar que a 5ª Divisão ficou sem up móveis. Na próxima iteração, então, ela só poderá contar com a up fixa que ela possui. É importante ressaltar que o algoritmo reinicializa as up móveis após certo número de iterações a fim de evitar que algumas divisões não sejam mais visitadas e para evitar convergência para máximos locais.

- **Escolha da atribuição que será mantida**

Quando se gera uma atribuição vizinha, pode-se ficar com ela, eliminando a atribuição já existente, ficar com as duas ou ignorar a atribuição vizinha gerada. Ficar com as duas atribuições não é um bom caminho, pois fará com que a quantidade de atribuições cresça rapidamente, dificultando o controle dos limites das divisões e por conseqüência piorando o desempenho. Por outro lado, somente ignorar as atribuições originais é desinteressante, pois se pode perder muito facilmente uma excelente atribuição. Sendo assim, para se determinar com que atribuição ficar, foi escolhido o critério de seleção do *Simulated Annealing* porque tem se mostrado bastante interessante e aplicável nas mais diversas situações.

Pode-se dizer então, que quando ocorre a geração de uma atribuição vizinha a partir de uma atribuição original, aplica-se o critério do Simulated Annealing e mantem-se a atribuição escolhida por esse critério.

- **A Temperatura**

O algoritmo é iniciado com uma temperatura inicial, a qual se espera que decresça até um valor mínimo. Para essa redução da temperatura foi utilizado o conceito de taxa de decréscimo variável, de forma que, inicialmente a taxa de decréscimo seja alta, mas que no decorrer do algoritmo esta taxa vá diminuindo, fazendo com que o algoritmo trabalhe mais tempo em temperaturas mais baixas, o que experimentalmente mostrou-se mais eficiente.

A cada iteração exterior completa, ou seja, a cada vez que todas as divisões geram suas soluções vizinhas e remontam o espaço populacional, o algoritmo aplica a taxa de redução da temperatura, fazendo com que esta se aproxime do mínimo e influencie de maneira diferente na escolha de que atribuição será mantida (a atribuição vizinha ou a atribuição original).

O algoritmo pode ser descrito da seguinte forma:

```
Algoritmo Multiplex()
Inicio
    Temp Corrente = TEMPERATURA_INICIAL
    Ler Parâmetros de Entrada
    Gerar e ordenar soluções iniciais
    Guardar melhor solução encontrada
    Dividir as soluções em divisões de tamanho pré-estabelecido (lido do arquivo de
    processamento)
    Distribuir as unidades de processamento de acordo com o arquivo de processamento
    Para i de 1 até o num_iteracoes_externas faça
        Para j de 1 até o num_iteracoes_internas faça
            Para k de 1 até o número de divisões faça
                Divisão = divisão[k];
                Para p de 1 até o num_up_fixas + num_up_moveis de Divisão faça
                    Atribuição = Divisão.ObterAtribuiçãoAleatoriamente();
                    Atribuição_Vizinha = Atribuição.GerarVizinha();
                    Delta = Atribuição_Vizinha.Rank() – Atribuição.Rank();
                    Se Delta > 0
                         Se Atribuição_Vizinha.Rank() > Melhor_encontrada.Rank();
                    Atualizar_Melhor_Atribuicao_Encontrada;
                         Fim se;
                    Senão Se Randômico_0_até_1 >= exp(Delta/ Temp Corrente)
                        Despreza Atribuição Vizinha
                    Senão   Despreza Atribuição
                    Fim se
                Fim para
            Fim para
            Se Divisão.Evoluiu
                Divisão.Ganhar_Unidades_de_Processamento
            Senão
                Divisão.Perder_Unidades_de_Processamento
            Fim se
        Fim para
            OrdenarAtribuicoes();
            ReiniciarDivisoes();
            Se α < TAXA_MAXIMA
                    Temp Corrente  = Temp Corrente * α;
                    α = α * TAXA_AUMENTO
        Senão
                α = TAXA_MAXIMA;
        Fim para
        Imprime Melhor Solução Encontrada
Fim Algoritmo
```

Algoritmo 4.2.1 – Multiplex

4.3. ANÁLISE DE COMPLEXIDADE

Quando um novo algoritmo é apresentado, um questionamento imediatamente é feito: Qual é a complexidade desse algoritmo ? Neste tópico, será apresentada a complexidade do Multiplex.

Pode-se dividir o Multiplex em procedimentos menores. A soma dos tempos destes procedimentos determina o tempo de execução do Multiplex. Será feita uma análise sobre cada um dos procedimentos menores encontrados no Multiplex e ao final deste processo, o tempo de execução e a complexidade do algoritmo serão determinados. Verifica-se então o tempo de execução do algoritmo Multiplex da seguinte forma:

Tempo de execução = Tempo de Leitura dos Parâmetros de Entrada + Tempo Gerar Soluções Iniciais + número de iterações externas * número de iterações internas * $\sum$ Tempos para gerar todas as soluções de cada divisão.

Tempo de Leitura dos Parâmetros de Entrada = *2* * tempo de leitura de um número + $\sum$ *(2* + número de literais de cada cláusula) * tempo de leitura de um número.

4.3.1. Tempo de Leitura dos Parâmetros de Entrada

O primeiro elemento a ser avaliado para se determinar o tempo de execução é o tempo de leitura dos parâmetros de entrada.

- **No pior caso:**

Neste caso o número de literais de cada cláusula é o número de variáveis *n*, daí tem-se:

Tempo de Leitura dos Parâmetros de Entrada = $2 * c_1 + \sum_{j=1}^{m} (2 + n) * c_1 +$

$C = 2 * c + 2 * m + m * n + C = O(m * n)$

- **No caso médio**:

Normalmente o número de literais de uma cláusula é pequeno e muito menor que o número de variáveis. Logo, pode-se considerar que o número de literais de uma cláusula é uma constante k. Daí:

Tempo de Leitura dos Parâmetros de Entrada = $2 * c_1 + \sum_{j=1}^{m} (2 + k) * c_1 +$

$C = 2 * c + 2 * m + m * k + C = O(m)$

C: é o tempo de leitura dos parâmetros de processamento. Esses parâmetros são proporcionalmente em menor quantidade se comparados aos parâmetros de entrada, sendo assim, ordem de complexidade é constante, ou seja, *O(1).*

c_1: é o tempo de leitura de um número de um arquivo.

4.3.2. Tempo para Gerar Soluções Iniciais

O tempo de geração das soluções iniciais em si é pequeno, entretanto o tempo para avaliação dessas soluções tem um impacto mais significativo. Será mostrada uma análise deste aspecto a seguir:

- **No pior caso:**

Na geração de soluções iniciais atribui-se aleatoriamente um valor a cada uma das variáveis, assim esse algoritmo é obrigatoriamente *O(n)*, mas a função objetivo precisa calcular o quão boa foi cada solução gerada. No pior caso será necessário passar por todos os literais de todas as cláusulas, cada uma das m cláusulas com *n* literais, para determinar a qualidade de cada solução gerada:

Tempo para Gerar Soluções Iniciais = $O(n) + m * n * c_2 = O(n*m)$

- **No caso Médio:**

Tempo para Gerar Soluções Iniciais = $O(n) + m * c_3 * c_2 = O(m)$ supondo m normalmente maior que *n*.

c_2: é o tempo de avaliação de um literal.
c_3: é o número de variáveis avaliadas em uma cláusula.

4.3.3. Tempo para Gerar as Soluções de uma Divisão

Cada divisão pode ter uma quantidade de atribuições diferentes a ser trabalhada, será chamado *Q* o somatório do número de tentativas de geração de novas soluções das divisões. Por conseguinte, pode-se dizer que:

Tempo para Gerar Soluções de uma Divisão = *O(Q/número de divisões)* * Tempo para Gerar uma Solução Vizinha = $O(c_4)$ * Tempo para Gerar uma Solução Vizinha.

c_4: é uma constante determinada pela divisão de duas constantes: *Q* e o número de divisões.

Tempo para Gerar uma Solução Vizinha = c_5 + Tempo de Avaliação da Nova Solução.

c_5: é o tempo de geração de um aleatório somado com a troca de uma variável.

4.3.4. Tempo de Avaliação da Nova Solução

Para avaliar esse tempo basta testar as cláusulas nas quais aparece a variável, uma vez que na geração de uma vizinha é feita apenas a troca de uma variável.

- **No pior caso:**

Se uma variável aparece em todas as cláusulas então:

Tempo de Avaliação da Nova Solução = $c_6 * m * n = O(m * n)$ pois neste caso, seria necessário passar por todas as cláusulas e a variável seria a última a ser avaliada.

- **No caso médio:**

Tempo de Avaliação da Nova Solução = $c_6 * m / n * g = O(m / n)$ pois neste caso, pode-se considerar que cada variável aparece a mesma quantidade de vezes nas cláusulas (m / n), como cada cláusula possui quantidade de variáveis constante g, então se tem $O(m / n)$.

c_6: é o tempo de avaliação de um literal.

4.3.5. Análise de Complexidade

A análise de complexidade de uma função pode ser feita pela soma das complexidades de suas componentes, então:

- **No pior caso:**

Tempo de execução = *O(m * n) + O(m * n) + num_iter_ext * num_iter_int * num_divisoes * O(m * n) = O(m * n).*

Considerando que o número de iterações externas, internas e divisões sejam constantes.

Se supuser que elas são proporcionais ao tamanho da entrada então ter-se-á:

Tempo de execução = $O(m * n) + O(m * n) + num\_iter\_ext * num\_iter\_int * num\_divisoes * O(m * n) = O(m * n) + O(m * n) + k_1 * m * k_2 * m * k_3 * m * O(m * n) = O(m^4 * n)$

Se n for proporcional a m, então Tempo de execução = $O(m^5)$

k_1, k_2, k_3: constantes de proporcionalidade.

- **No caso médio:**

Tempo de execução = $O(m) + O(m) + num\_iter\_ext * num\_iter\_int * num\_divisoes * O(m / n) = O(m)$.

Se for considerado que normalmente $m = b * n$ onde b é uma constante, pode-se considerar que o algoritmo é: $O(m^2)$ ou $O(n^2)$.

5. RESULTADOS EXPERIMENTAIS

Nesta seção, serão apresentados resultados para um conjunto de unidades randomicamente geradas, de problemas MAX-SAT ponderado. Tais unidades se dividem em dois grupos:

- Os jnhX.sat, ver Resende (1996), todos com 100 variáveis e com 800 a 900 cláusulas, alguns satisfatíveis e outros não. Os pesos das cláusulas estão uniformemente distribuídos entre 1 e 1000 e a quantidade de literais por cláusula é variável, para estas unidades de problema é conhecido o valor ótimo.
- Os rg_200_2000_4_X.sat, que possuem 800 variáveis e 8200 cláusulas, os rndw1000aX.sat que possuem 1000 variáveis e 7700 cláusulas e os rndw1000bX.sat com 1000 variáveis e 11050 cláusulas. Todos obtidos do endereço eletrônico: www-or.amp.i.kyoto-u.ac.jp/members/yagiura/index-e.html. Para essas unidades de problema não é conhecido o valor ótimo nem a função de distribuição e a quantidade de literais por cláusula é variável.

Aqui, X é um número utilizado para diferenciação.

Todos os testes desta seção foram feitos em um Sempron 2200+ (1.5 GHz), com 512 MB de memória Ram, Placa de Vídeo Xabre 200, placa mãe PC-Chips M-847, utilizando o sistema operacional Linux Kurumin 5.0, utilizando linguagem de programação C no compilador gcc 3.3.5.

O arquivo de processamento, arquivo que define as características das divisões, possui os seguintes parâmetros:

- Existem 4 divisões e 80 soluções distribuídas entre elas.
- A primeira, definida da solução 0 até a solução 18, com 2 unidades de processamento fixas e 90 unidades de processamento móveis iniciais.
- A segunda, da solução 19 até a solução 38, com 2 unidades de processamento fixas e 20 unidades de processamento móveis iniciais.
- A terceira, da solução 39 até a solução 58, com 2 unidades de processamento fixas e 20 unidades de processamento móveis iniciais.
- A última divisão é definida da solução 59 até a solução 79, lembrando que a linguagem C começa a contagem de 0 (zero), por isso a última divisão termina na solução 79. Esta divisão também possui 2 unidades de processamento fixas e 90 unidades de processamento móveis iniciais.

5.1. ANÁLISE EXPERIMENTAL

Para analisar o desempenho do algoritmo MULTIPLEX serão feitas comparações com algoritmos normalmente utilizados na resolução de problemas do tipo MAX-SAT Ponderado. São eles: GLS - *Guided Local Search* (Mills e Tsang, 2000), ILS - *Iterated Local Search* (Yagiura e Ibaraki, 1998) (Lourenço, et al. 2001), GRASP - *Greedy Randomized Adaptative Search Procedure* (Resende, 1996), e SA - *Simulated Annealing* (Kirkpatrick, et al. 1983), as implementações para esses

algoritmos podem ser encontradas no endereço eletrônico: www-or.amp.i.kyoto-u.ac.jp/members/yagiura/index-e.html. A análise será feita levando em conta a solução média, a solução máxima e a solução mínima. Para a análise foram feitas 10 execuções de cada algoritmo para cada unidade de problema.

As únicas alterações feitas nos algoritmos obtidos foram:

- Critério de parada, levando em consideração apenas o tempo.
- Mudança da função de geração de aleatórios (utilização da função rand()) para que a semente aleatória utilizada também seja aleatória (utilização da função srand()) .
- Colocação do tempo limite máximo em 120 segundos.

A seguir são mostradas tabelas comparativas entre os algoritmos testados. Foi dado o tempo de 120 segundos de execução para cada algoritmo. As unidades de problema reconhecidamente satisfatíveis terão o sinal * entre parênteses antes do nome da unidade, a fim de diferenciar unidades satisfatíveis, que possuem o sinal, de unidades insatisfatíveis

Os melhores resultados de cada arquivo aparecerão em vermelho e os resultados médios em negrito. Em caso de empate ambas soluções serão consideradas melhores soluções. Será mostrado o percentual proporcional ao valor das médias, para cada tabela mostrada, e o maior percentual será mostrado em vermelho.

5.1.1. Arquivos rndw1000bX.sat

Estas unidades de problema podem ser consideradas as mais difíceis, pois são as que possuem maior número de cláusulas e variáveis. A seguir serão analisados os resultados que cada um dos algoritmos obteve para estas unidades de problema:

Arquivo	GLS	ILS	GRASP	SA	Multiplex
rndw1000b01.sat	5547551	5550332	5522901	5542995	5548658
rndw1000b02.sat	5517463	5518415	5494880	5512208	5517242,9
rndw1000b03.sat	5575223	5576516	5551512	5570596	5573947
rndw1000b04.sat	5500740	5499429	5476729	5494513	5499190
rndw1000b05.sat	5522645	5524353	5498424	5517280	5521422
rndw1000b06.sat	5518333	5521294	5495342	5514067	5519033
rndw1000b07.sat	5509360	5511648	5484904	5504866	5507378
rndw1000b08.sat	5486044	5487629	5462919	5484023	5487767
rndw1000b09.sat	5538680	5541278	5512672	5534756	5538722
rndw1000b10.sat	5550125	5551335	5526222	5546388	5548338
Média	**5526616**	**5528223**	**5502651**	**5522169**	**5526170**
Percentual	**99,971%**	**100%**	**99,537%**	**99,890%**	**99,963%**

Tabela 5.1.1.1 – Soluções Médias para os arquivos rndw1000bX.sat

Arquivo	GLS	ILS	GRASP	SA	Multiplex
rndw1000b01.sat	5547551	5550332	5522901	5542995	5548658
rndw1000b02.sat	5517463	5518415	5494880	5512277	5517243
rndw1000b03.sat	5575223	5576517	5551512	5570596	5573947
rndw1000b04.sat	5500740	5499429	5476729	5494513	5499190
rndw1000b05.sat	5522645	5524353	5498424	5517280	5521422

rndw1000b06.sat	5518333	5521294	5495342	5514067	5519033
rndw1000b07.sat	5509360	5511648	5484904	5504866	5507378
rndw1000b08.sat	5486044	5487629	5462919	5484023	5487767
rndw1000b09.sat	5538680	5541278	5512672	5534756	5538722
rndw1000b10.sat	5550125	5551335	5526222	5546388	5548338
Média	**5526616**	**5528223**	**5502651**	**5522176**	**5526170**
Percentual	**99,971%**	**100%**	**99,537%**	**99,890%**	**99,963%**

Tabela 5.1.1.2 – Soluções Máximas para os arquivos rndw1000bX.sat

Arquivo	**GLS**	**ILS**	**GRASP**	**SA**	**Multiplex**
rndw1000b01.sat	5547551	5550332	5522901	5542995	5548658
rndw1000b02.sat	5517463	5518415	5494880	5511586	5517242
rndw1000b03.sat	5575223	5576514	5551512	5570596	5573947
rndw1000b04.sat	5500740	5499429	5476729	5494513	5499190
rndw1000b05.sat	5522645	5524353	5498424	5517280	5521422
rndw1000b06.sat	5518333	5521294	5495342	5514067	5519033
rndw1000b07.sat	5509360	5511648	5484904	5504866	5507378
rndw1000b08.sat	5486044	5487629	5462919	5484023	5487767
rndw1000b09.sat	5538680	5541278	5512672	5534756	5538722
rndw1000b10.sat	5550125	5551335	5526222	5546388	5548338
Média	**5526616**	**5528223**	**5502651**	**5522107**	**5526170**
Percentual	**99,971%**	**100%**	**99,537%**	**99,889%**	**99,963%**

Tabela 5.1.1.3 – Soluções Mínimas para os arquivos rndw1000bX.sat

5.1.1.1. Análise dos Resultados

O ILS se mostrou superior às demais meta-heuristicas, obtendo 80 % dos melhores resultados, tanto para solução média quanto para solução máxima e mínima. Isto mostra que além de eficiente a meta-heurística é robusta. No segundo lugar ficou o GLS pois obteve 10 % das melhores soluções encontradas, assim como o Multiplex, mas obteve uma média das soluções um pouco melhor. O Multiplex obteve um bom desempenho, ficando em terceiro, mas com resultado médio praticamente igual ao do GLS. O SA ficou atrás do Multiplex, vencendo apenas o Grasp que vai se mostrar o algoritmo de pior desempenho dos testes pois as suas soluções sempre são as piores dentre os algoritmos testados.

5.1.1.2. Análise Gráfica

O gráfico 5.1.1.2.1 ilustra a superioridade do ILS em relação aos demais algoritmos no que tange quantidade de melhores soluções encontradas. Notar no gráfico 5.1.1.2.2 a proximidade das soluções de ILS, GLS e Multiplex, mostrando que apesar do ILS ser superior na maior parte das vezes a diferença normalmente é bem pequena.

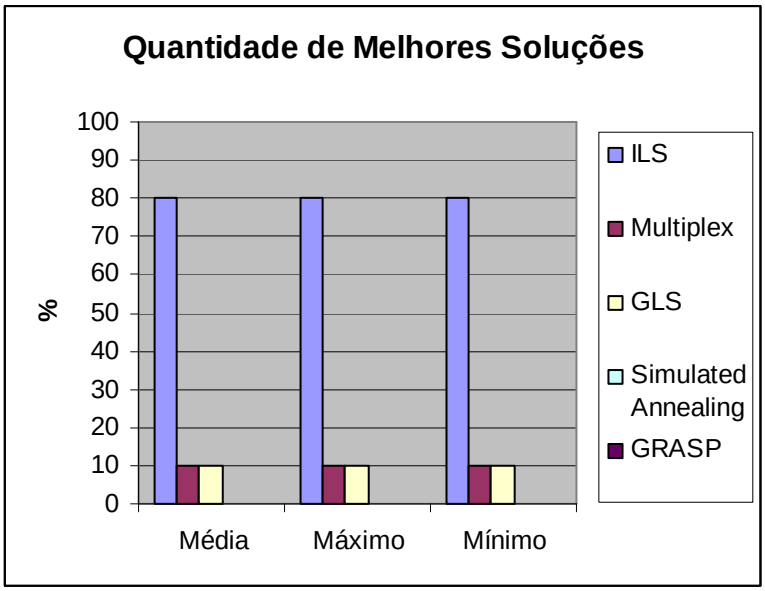

Gráfico 5.1.1.2.1 – Melhores Soluções Encontradas nos Arquivos rndw1000b[01-10]

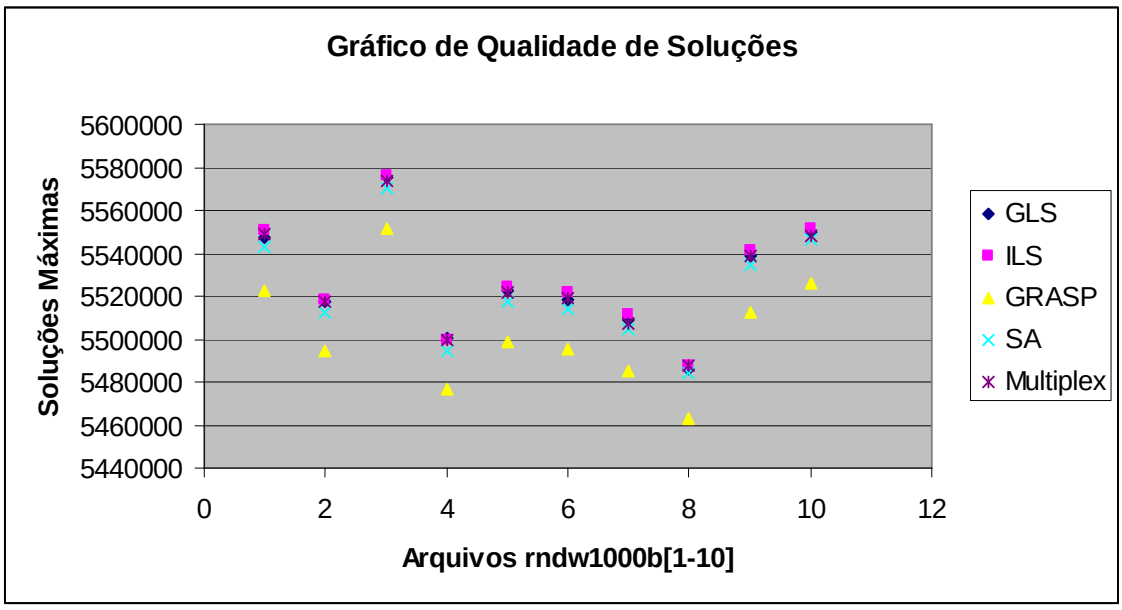

Gráfico 5.1.1.2.2 – Comparando a Qualidade das Soluções dos Arquivos rndw1000b[01-10]

5.1.2. Arquivos rndw1000aX.sat

As unidades de problema rndw1000aX.sat possuem dificuldade intermediária, pois possuem a mesma quantidade de variáveis que os problemas rndw1000bX.sat, mas menos cláusulas. A seguir serão mostrados os resultados obtidos:

Arquivo	GLS	ILS	GRASP	SA	Multiplex
rndw1000a01.sat	3867931	3868242	3854794	3866307	3868190
rndw1000a02.sat	3842717	3843004	3832186	3841315	3842860
rndw1000a03.sat	3898353	3899475	3883620	3897700	3899409
rndw1000a04.sat	3819652	3819957	3805126	3817735	3819989
rndw1000a05.sat	3858654	3859146	3844614	3856690	3859439
rndw1000a06.sat	3866246	3843859	3851289	3864424	3867496
rndw1000a07.sat	3841170	3849775	3828616	3838305	3842152
rndw1000a08.sat	3835601	3836494	3820424	3834339	3836390
rndw1000a09.sat	3882612	3883053	3875237	3880903	3882955
rndw1000a10.sat	3878681	3879191	3867652	3877106	3878977
Média	**3859162**	**3858220**	**3846356**	**3857482**	**3859786**
Percentual	**99,984%**	**99,954%**	**99,652%**	**99,940%**	**100%**

Tabela 5.1.2.1 – Soluções Médias para os arquivos rndw1000aX.sat

Arquivo	GLS	ILS	GRASP	SA	Multiplex
rndw1000a01.sat	3867931	3868242	3854794	3866307	3868200
rndw1000a02.sat	3842717	3843004	3832186	3841315	3842868
rndw1000a03.sat	3898353	3899475	3883620	3897700	3899409
rndw1000a04.sat	3819652	3819957	3805381	3817735	3820043

rndw1000a05.sat	3858654	3859146	3844614	3856690	3859439
rndw1000a06.sat	3866246	3867761	3851289	3864424	3867496
rndw1000a07.sat	3841170	3859146	3828616	3838305	3842152
rndw1000a08.sat	3835601	3836494	3820424	3834339	3836390
rndw1000a09.sat	3882612	3883053	3875237	3880903	3882955
rndw1000a10.sat	3878681	3879191	3867652	3877106	3878977
Média	**3859162**	**3861547**	**3846381**	**3857482**	**3859793**
Percentual	**99,938%**	**100%**	**99,607%**	**99,894%**	**99,955%**

Tabela 5.1.2.2 – Soluções Máximas para os arquivos rndw1000aX.sat

Arquivo	**GLS**	**ILS**	**GRASP**	**SA**	**Multiplex**
rndw1000a01.sat	3867931	3868242	3854794	3866307	3868180
rndw1000a02.sat	3842717	3843004	3832186	3841315	3842858
rndw1000a03.sat	3898353	3899475	3883620	3897700	3899409
rndw1000a04.sat	3819652	3819957	3804207	3817735	3819935
rndw1000a05.sat	3858654	3859146	3844614	3856690	3859439
rndw1000a06.sat	3866246	3819957	3851289	3864424	3867496
rndw1000a07.sat	3841170	3842279	3828616	3838305	3842152
rndw1000a08.sat	3835601	3836494	3820424	3834339	3836390
rndw1000a09.sat	3882612	3883053	3875237	3880903	3882955
rndw1000a10.sat	3878681	3879191	3867652	3877106	3878977
Média	**3859162**	**3855080**	**3846264**	**3857482**	**3859779**
Percentual	**99,984%**	**99,878%**	**99,650%**	**99,940%**	**100%**

Tabela 5.1.2.3 – Soluções Mínimas para os arquivos rndw1000aX.sat

5.1.2.1. Análise dos Resultados

Novamente o ILS se mostra superior obtendo a maioria das melhores soluções encontradas, entretanto no quesito soluções mínimas a sua média ficou inferior ao Multiplex. Aliás, neste conjunto de arquivos que o Multiplex mais se destacou obtendo 26,67% das melhores soluções, enquanto o GLS e o SA não obtiveram uma única melhor solução.

5.1.2.2. Análise Gráfica

O gráfico 5.1.2.2.1 ilustra a superioridade do ILS e do Multiplex em relação aos demais algoritmos no que tange quantidade de melhores soluções encontradas. Em relação a qualidade das soluções encontradas podemos verificar que GLS e Multiplex continuam praticamente juntos, enquanto o Simulated Annealing continua inferior a ILS, GLS e Multiplex. Observar que, para o desempenho do ILS no arquivo rndw1000a07.sat, a diferença foi bastante significativa.

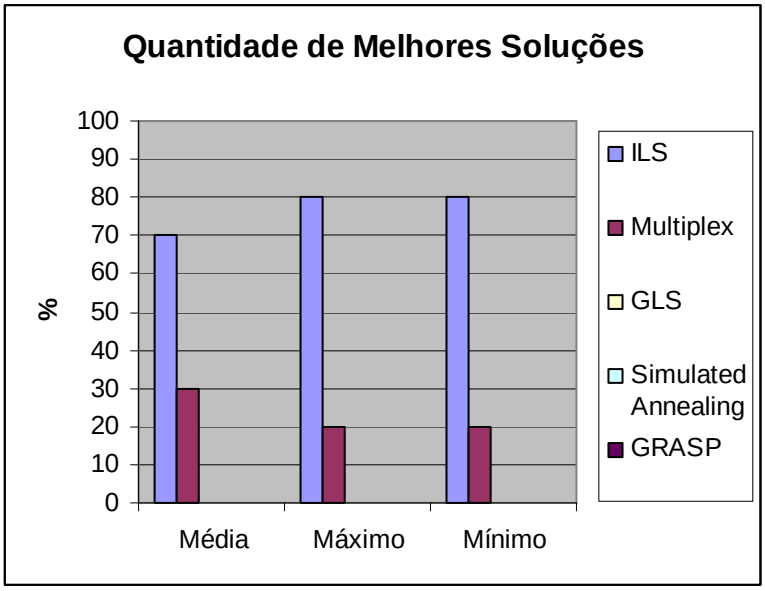

Gráfico 5.1.2.2.1 – Melhores Soluções Encontradas nos Arquivos rndw1000a[01-10]

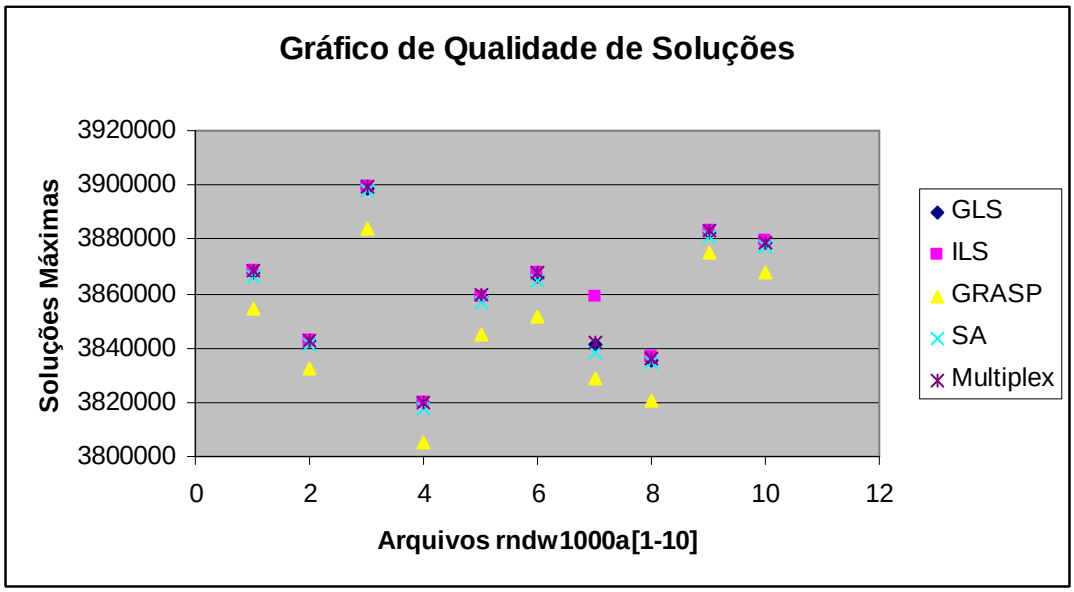

Gráfico 5.1.2.2.2 – Comparando a Qualidade das Soluções dos Arquivos rndw1000a[01-10]

5.1.3. Arquivos rg_200_2000_4_X.sat

As unidades de problema rg_200_2000_4_X.sat são bastante restritivas pois possuem mais cláusulas que as unidades do tipo rndw1000aX.sat e menos variáveis, logo pode-se considerá-las mais difíceis que as rndw1000aX.sat, mas menos difíceis que as rndw1000bX.sat, já que estas unidades de problema possuem mais variáveis e cláusulas.

Arquivo	GLS	ILS	GRASP	SA	Multiplex
rg_200_2000_4_11.sat	7012460	7057099	6901254	6974676	6965268
rg_200_2000_4_13.sat	7285463	7325087	7168991	7257043	7234973
rg_200_2000_4_15.sat	7319257	7361395	7203162	7284603	7258001
rg_200_2000_4_17.sat	7459379	7500397	7333214	7419508	7397285

rg_200_2000_4_19.sat	7436487	7471363	7306243	7394628	7375150
rg_200_2000_4_21.sat	7322869	7373491	7209917	7296924	7278260
rg_200_2000_4_23.sat	7528170	7561448	7403857	7485125	7463943
rg_200_2000_4_25.sat	6972198	7005446	6847013	6927444	6915416
rg_200_2000_4_27.sat	7796604	7835972	7672864	7749085	7738597
rg_200_2000_4_29.sat	7501510	7545798	7379995	7465516	7443450
Média	**7363440**	**7403750**	**7242651**	**7325455**	**7307034**
Percentual	**99,455%**	**100%**	**97,824%**	**98,942%**	**98,694%**

Tabela 5.1.3.1 – Soluções Médias para os arquivos rg_200_2000_4_X.sat

Arquivo	GLS	ILS	GRASP	SA	Multiplex
rg_200_2000_4_11.sat	7012460	7057099	6901254	6974676	6965268
rg_200_2000_4_13.sat	7285463	7325087	7168991	7257043	7234973
rg_200_2000_4_15.sat	7319257	7361395	7203162	7284603	7258001
rg_200_2000_4_17.sat	7459379	7500397	7333214	7419508	7397285
rg_200_2000_4_19.sat	7436487	7471363	7306243	7394628	7375150
rg_200_2000_4_21.sat	7322869	7373491	7209917	7296924	7278260
rg_200_2000_4_23.sat	7528170	7561448	7403857	7485125	7463943
rg_200_2000_4_25.sat	6972198	7005446	6847013	6927444	6915416
rg_200_2000_4_27.sat	7796604	7835972	7672864	7749085	7738597
rg_200_2000_4_29.sat	7501510	7545798	7379995	7465516	7443450
Médias	**7363440**	**7403750**	**7242651**	**7325455**	**7307034**
Percentual	**99,455%**	**100%**	**97,824%**	**98,942%**	**98,694%**

Tabela 5.1.3.2 – Soluções Máximas para os arquivos rg_200_2000_4_X.sat

Arquivo	GLS	ILS	GRASP	SA	Multiplex
rg_200_2000_4_11.sat	7012460	7057099	6901254	6974676	6965268

rg_200_2000_4_13.sat	7285463	7325087	7168991	7257043	7234973
rg_200_2000_4_15.sat	7319257	7361395	7203162	7284603	7258001
rg_200_2000_4_17.sat	7459379	7500397	7333214	7419508	7397285
rg_200_2000_4_19.sat	7436487	7471363	7306243	7394628	7375150
rg_200_2000_4_21.sat	7322869	7373491	7209917	7296924	7278260
rg_200_2000_4_23.sat	7528170	7561448	7403857	7485125	7463943
rg_200_2000_4_25.sat	6972198	7005446	6847013	6927444	6915416
rg_200_2000_4_27.sat	7796604	7835972	7672864	7749085	7738597
rg_200_2000_4_29.sat	7501510	7545798	7379995	7465516	7443450
Médias	**7363440**	**7403750**	**7242651**	**7325455**	**7307034**
Percentual	**99,455%**	**100%**	**97,824%**	**98,942%**	**98,694%**

Tabela 5.1.3.3 – Soluções Mínimas para os arquivos rg_200_2000_4_X.sat

5.1.3.1. Análise dos Resultados

O Multiplex obteve um desempenho fraco neste grupo de problemas, obtendo melhores soluções apenas se comparado ao GRASP. A diferença em relação ao SA foi pequena, mas constante. O Multiplex teve pior desempenho se comparado ao SA em todas as unidades de problema. O ILS foi o superior em todas as instâncias de problema, seguido do GLS e do SA em terceiro.

5.1.3.2. Análise Gráfica

Para este conjunto de arquivos o ILS foi incontestável, pois simplesmente obteve todas as melhores soluções. Para ilustrar esta superioridade apresenta-se o gráfico 5.1.3.2.1:

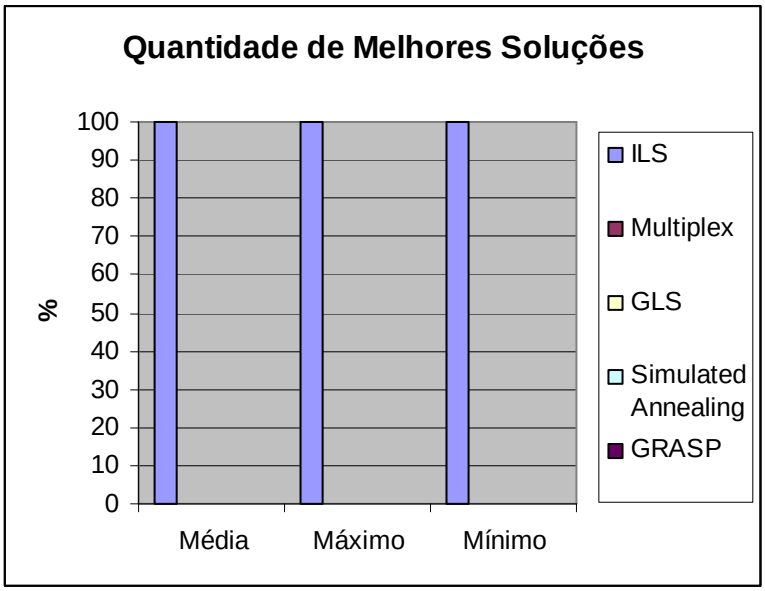

Gráfico 5.1.3.2.1 – Melhores Soluções Encontradas nos Arquivos rg_200_2000_4_[11-29]

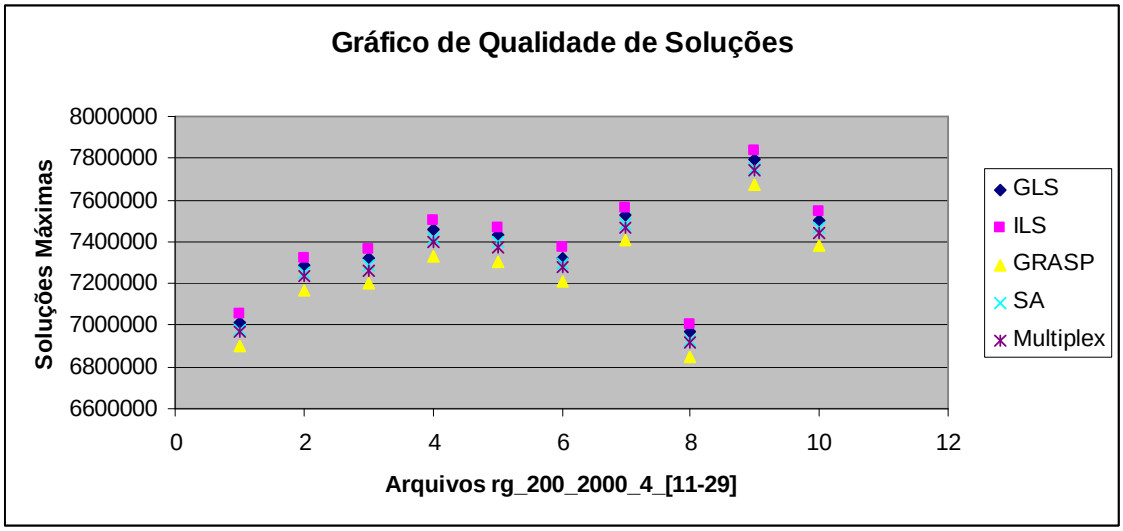

Gráfico 5.1.3.2.2 – Comparando a Qualidade das Soluções dos Arquivos rg_200_2000_4_[11-29]

5.1.4. Arquivos jnhX.sat

São as unidades de problema mais fáceis, pois possuem apenas 100 variáveis e de 800 a 900 cláusulas, seus ótimos são conhecidos e serão mostrados nas tabelas abaixo:

Arquivo	Ótimo	GLS	ILS	GRASP	SA	Multiplex
(*) jnh1.sat	420925	420892	420925	420088	420925	420925
jnh4.sat	420830	420789	420830	419091	420780	420813
jnh5.sat	420742	420742	420742	418708	420742	420742
jnh6.sat	420826	420826	420826	419307	420826	420826
(*) jnh7.sat	420925	420925	420925	418994	420925	420925
jnh8.sat	420463	420463	420463	419185	420463	420463
jnh9.sat	420592	420592	420592	418371	420529	420505
jnh10.sat	420840	420581	420840	419530	420770	420840
jnh11.sat	420753	420648	420753	419397	420740	420728
(*) jnh12.sat	420925	420706	420925	418561	420925	420925
jnh13.sat	420816	420816	420816	418843	420816	420816
jnh14.sat	420824	420636	420824	419500	420824	420763
jnh15.sat	420719	420538	420719	419068	420719	420719
jnh16.sat	420919	420914	420919	418984	420914	420914
jnh19.sat	420759	420680	420680	418807	420759	420680
(*)jnh201.sat	394238	394238	394238	393508	394238	394238
jnh202.sat	394170	393950	394170	392777	394170	394170
jnh203.sat	394199	394135	394199	392636	394135	394199
(*)jnh205.sat	394238	394238	394238	393020	394238	394238
(*)jnh207.sat	394238	394118	394238	392552	394196	394238
jnh208.sat	394159	393997	394159	392531	394159	394159
(*)jnh209.sat	394238	394226	394238	392571	394238	394328
(*)jnh210.sat	394238	394067	394238	393701	394238	394238
jnh211.sat	393979	393742	393979	392647	393742	393979
(*)jnh212.sat	394238	394227	394227	392902	394227	394227
jnh214.sat	394163	394163	394163	393024	394163	394163
jnh215.sat	394150	393951	394150	393023	394150	394150

jnh216.sat	394226	393648	394226	393056	394187	394226
(*)jnh217.sat	394238	394238	394238	393921	394238	394238
(*)jnh218.sat	394238	394238	394238	391490	394238	394238
jnh219.sat	394156	393759	394156	393109	394156	394070
(*)jnh220.sat	394238	394076	394238	393767	394164	394205
(*)jnh301.sat	444854	444790	444854	443906	444790	444790
jnh302.sat	444459	444288	444459	442254	444459	444459
jnh303.sat	444503	444299	444503	442739	444503	444423
jnh304.sat	444533	444266	444533	442945	444533	444533
jnh305.sat	444112	443822	444112	442419	444000	444112
jnh306.sat	444838	444838	444838	443995	444838	444838
jnh307.sat	444314	444314	444314	443544	444314	444314
jnh308.sat	444724	444403	444621	441895	444664	444664
jnh309.sat	444578	444578	444578	442913	444578	444578
jnh310.sat	444391	444391	444391	442387	444353	444391
Média	**415678,8**	**415565,4**	**415674,2**	**414182,5**	**415656,3**	**415666,4**
Percentual	**100%**	**99,972%**	**99,999%**	**99,640%**	**99,994%**	**99,997%**

Tabela 5.1.3.1 – Soluções Médias para os arquivos jnhX.sat

Arquivo	**Ótimo**	**GLS**	**ILS**	**GRASP**	**SA**	**Multiplex**
(*) jnh1.sat	420925	420892	420925	420088	420925	420925
jnh4.sat	420830	420789	420830	419091	420780	420813
jnh5.sat	420742	420742	420742	418708	420742	420742
jnh6.sat	420826	420826	420826	419307	420826	420826
(*) jnh7.sat	420925	420925	420925	418994	420925	420925
jnh8.sat	420463	420463	420463	419185	420463	420463
jnh9.sat	420592	420592	420592	418371	420529	420505
jnh10.sat	420840	420581	420840	419530	420770	420840
jnh11.sat	420753	420648	420753	419397	420740	420728

(*) jnh12.sat	420925	420706	420925	418561	420925	420925
jnh13.sat	420816	420816	420816	418843	420816	420816
jnh14.sat	420824	420636	420824	419500	420824	420763
jnh15.sat	420719	420538	420719	419068	420719	420719
jnh16.sat	420919	420914	420919	418984	420914	420914
jnh19.sat	420759	420680	420680	418807	420759	420680
(*) jnh201.sat	394238	394238	394238	393508	394238	394238
jnh202.sat	394170	393950	394170	392777	394170	394170
jnh203.sat	394199	394135	394199	392636	394135	394199
(*) jnh205.sat	394238	394238	394238	393020	394238	394238
(*) jnh207.sat	394238	394118	394238	392552	394196	394238
jnh208.sat	394159	393997	394159	392531	394159	394159
(*) jnh209.sat	394238	394226	394238	392571	394238	394328
(*) jnh210.sat	394238	394067	394238	393701	394238	394238
jnh211.sat	393979	393742	393979	392647	393742	393979
(*) jnh212.sat	394238	394227	394227	392902	394227	394227
jnh214.sat	394163	394163	394163	393024	394163	394163
jnh215.sat	394150	393951	394150	393023	394150	394150
jnh216.sat	394226	393648	394226	393056	394187	394226
(*) jnh217.sat	394238	394238	394238	393921	394238	394238
(*) jnh218.sat	394238	394238	394238	391490	394238	394238
jnh219.sat	394156	393759	394156	393109	394156	394070
(*) jnh220.sat	394238	394076	394238	393767	394164	394205
(*) jnh301.sat	444854	444790	444854	443906	444790	444790
jnh302.sat	444459	444288	444459	442254	444459	444459
jnh303.sat	444503	444299	444503	442739	444503	444423
jnh304.sat	444533	444266	444533	442945	444533	444533
jnh305.sat	444112	443822	444112	442419	444000	444112
jnh306.sat	444838	444838	444838	443995	444838	444838
jnh307.sat	444314	444314	444314	443544	444314	444314

jnh308.sat	444724	444403	444621	441895	444664	444664
jnh309.sat	444578	444578	444578	442913	444578	444578
jnh310.sat	444391	444391	444391	442387	444353	444391
Média	**415678,8**	**415565,4**	**415674,2**	**414182,5**	**415656,3**	**415666,4**
Percentual	**100%**	**99,972%**	**99,999%**	**99,640%**	**99,994%**	**99,997%**

Tabela 5.1.3.2 – Soluções Máximas para os arquivos jnhX.sat

Arquivo	Ótimo	GLS	ILS	GRASP	SA	Multiplex
(*) jnh1.sat	420925	420892	420925	420088	420925	420925
jnh4.sat	420830	420789	420830	419091	420780	420813
jnh5.sat	420742	420742	420742	418708	420742	420742
jnh6.sat	420826	420826	420826	419307	420826	420826
(*) jnh7.sat	420925	420925	420925	418994	420925	420925
jnh8.sat	420463	420463	420463	419185	420463	420463
jnh9.sat	420592	420592	420592	418371	420529	420505
jnh10.sat	420840	420581	420840	419530	420770	420840
jnh11.sat	420753	420648	420753	419397	420740	420728
(*) jnh12.sat	420925	420706	420925	418561	420925	420925
jnh13.sat	420816	420816	420816	418843	420816	420816
jnh14.sat	420824	420636	420824	419500	420824	420763
jnh15.sat	420719	420538	420719	419068	420719	420719
jnh16.sat	420919	420914	420919	418984	420914	420914
jnh19.sat	420759	420680	420680	418807	420759	420680
(*)jnh201.sat	394238	394238	394238	393508	394238	394238
jnh202.sat	394170	393950	394170	392777	394170	394170
jnh203.sat	394199	394135	394199	392636	394135	394199
(*)jnh205.sat	394238	394238	394238	393020	394238	394238
(*)jnh207.sat	394238	394118	394238	392552	394196	394238
jnh208.sat	394159	393997	394159	392531	394159	394159

(*)jnh209.sat	394238	394226	394238	392571	394238	394328
(*)jnh210.sat	394238	394067	394238	393701	394238	394238
jnh211.sat	393979	393742	393979	392647	393742	393979
(*)jnh212.sat	394238	394227	394227	392902	394227	394227
jnh214.sat	394163	394163	394163	393024	394163	394163
jnh215.sat	394150	393951	394150	393023	394150	394150
jnh216.sat	394226	393648	394226	393056	394187	394226
(*)jnh217.sat	394238	394238	394238	393921	394238	394238
(*)jnh218.sat	394238	394238	394238	391490	394238	394238
jnh219.sat	394156	393759	394156	393109	394156	394070
(*)jnh220.sat	394238	394076	394238	393767	394164	394205
(*)jnh301.sat	444854	444790	444854	443906	444790	444790
jnh302.sat	444459	444288	444459	442254	444459	444459
jnh303.sat	444503	444299	444503	442739	444503	444423
jnh304.sat	444533	444266	444533	442945	444533	444533
jnh305.sat	444112	443822	444112	442419	444000	444112
jnh306.sat	444838	444838	444838	443995	444838	444838
jnh307.sat	444314	444314	444314	443544	444314	444314
jnh308.sat	444724	444403	444621	441895	444664	444664
jnh309.sat	444578	444578	444578	442913	444578	444578
jnh310.sat	444391	444391	444391	442387	444353	444391
Média	**415678,8**	**415565,4**	**415674,2**	**414182,5**	**415656,3**	**415666,4**
Percentual	**100%**	**99,972%**	**99,999%**	**99,640%**	**99,994%**	**99,997%**

Tabela 5.1.3.2 – Soluções Mínimas para os arquivos jnhX.sat

5.1.4.1. Análise dos Resultados

O ILS manteve-se superior às demais meta-heurísticas, tendo a melhor média e 40 melhores soluções, seguido do Multiplex com 32 e do SA com

30. O GLS teve um desempenho um pouco inferior, com apenas 16 melhores soluções.

5.1.4.2. Análise Gráfica

Para este conjunto de arquivos o ILS foi superior, mas a diferença não foi muito grande em relação ao Multiplex e ao SA que se mostraram muito bons para problemas pequenos. Já o GLS parece não ser uma heurística muito indicada para problemas pequenos, pois obteve resultados significativamente inferiores em relação ao ILS, Multiplex e SA. A seguir será mostrado o gráfico 6.1.4.2.1 que ilustra a situação descrita:

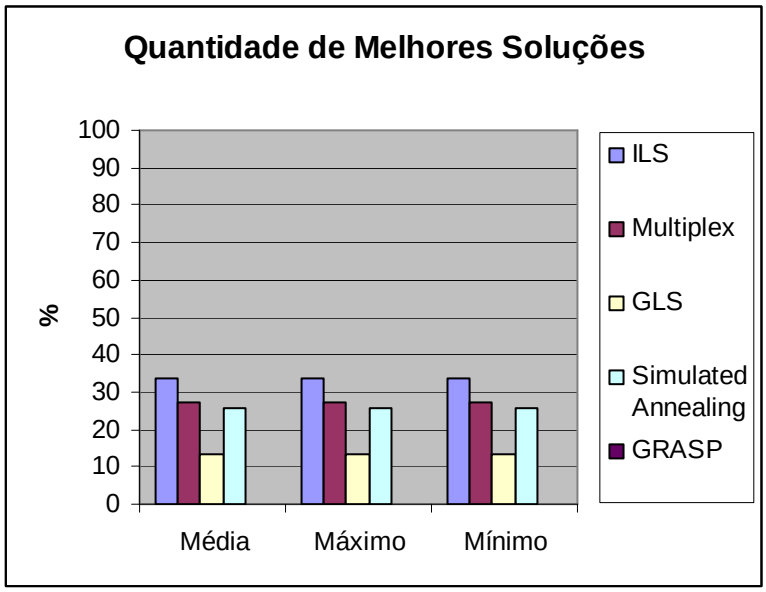

Gráfico 5.1.4.2.1 – Melhores Soluções Encontradas nos Arquivos jnh[1-310]

Nos gráficos de qualidade de solução mostrados abaixo, nota-se o equilíbrio entre ILS, Multiplex e SA e, com resultados um pouco inferiores, o GLS.

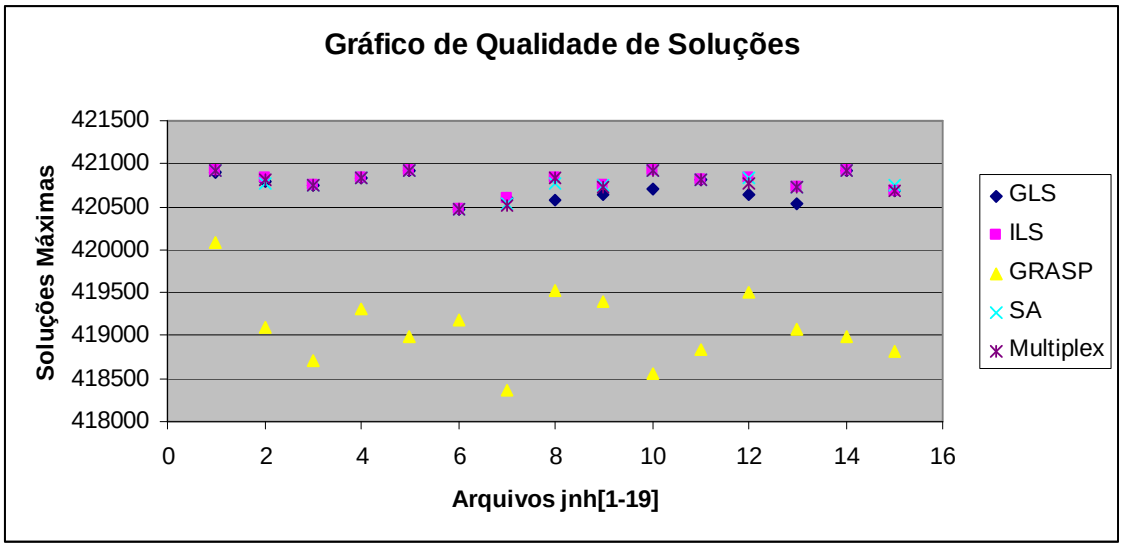

Gráfico 5.1.4.2.2 – Comparando a Qualidade das Soluções dos Arquivos jnh[1-19]

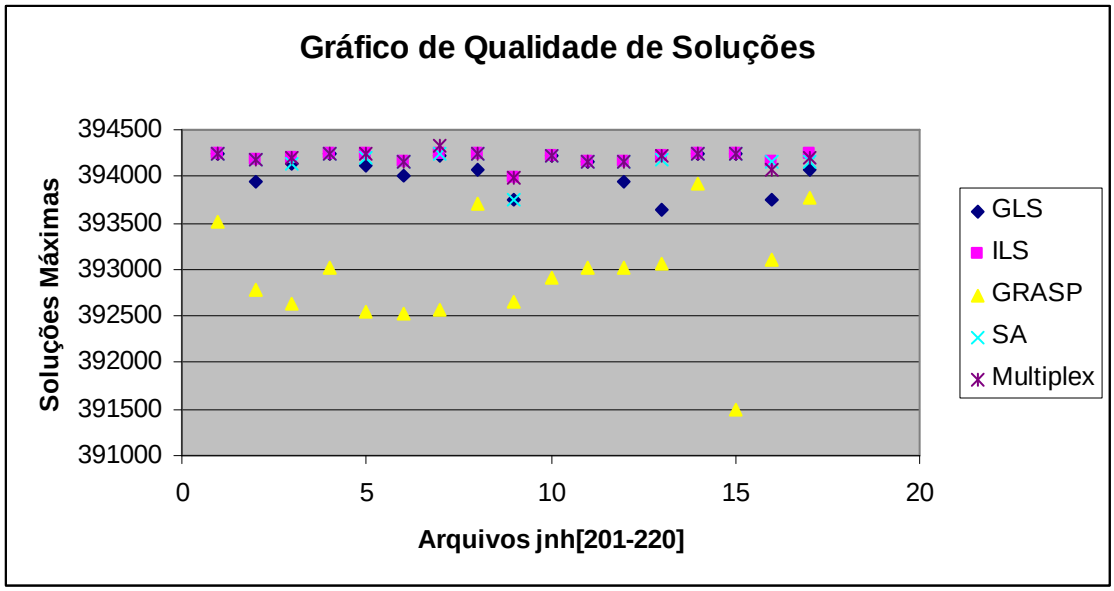

Gráfico 5.1.4.2.3 – Comparando a Qualidade das Soluções dos Arquivos jnh[201-220]

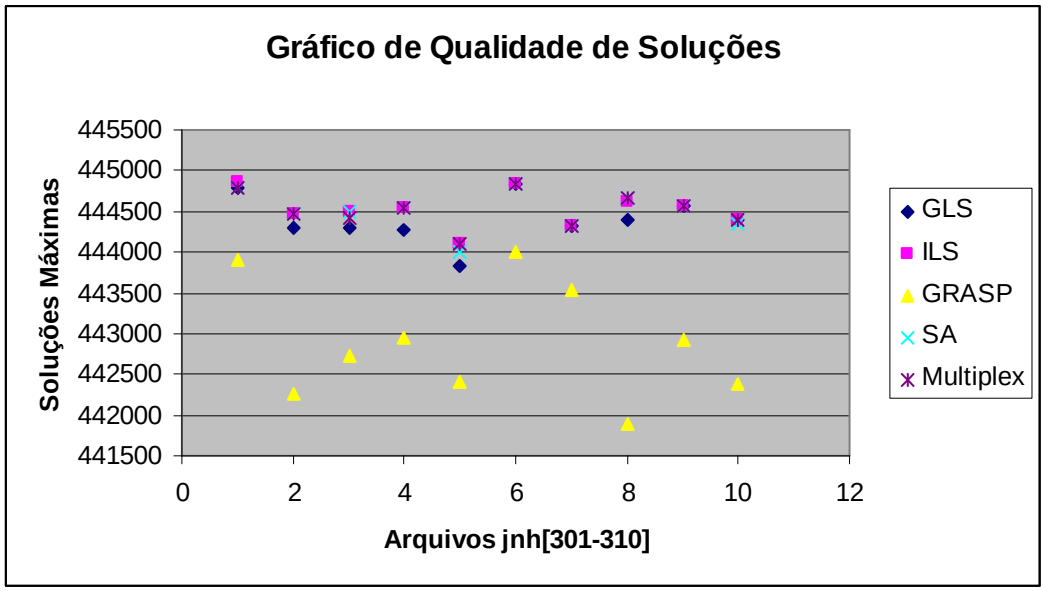

Gráfico 5.1.4.2.4 – Comparando a Qualidade das Soluções dos Arquivos jnh[301-310]

5.1.5. Análise Gráfica Geral

A comparação, levando em conta todos os arquivos, é ilustrada no gráfico 5.1.5.1. Nele podemos verificar que o ILS é realmente superior. Nesta análise será levado em conta apenas o caso médio, pois busca-se uma visão mais global dos algoritmos:

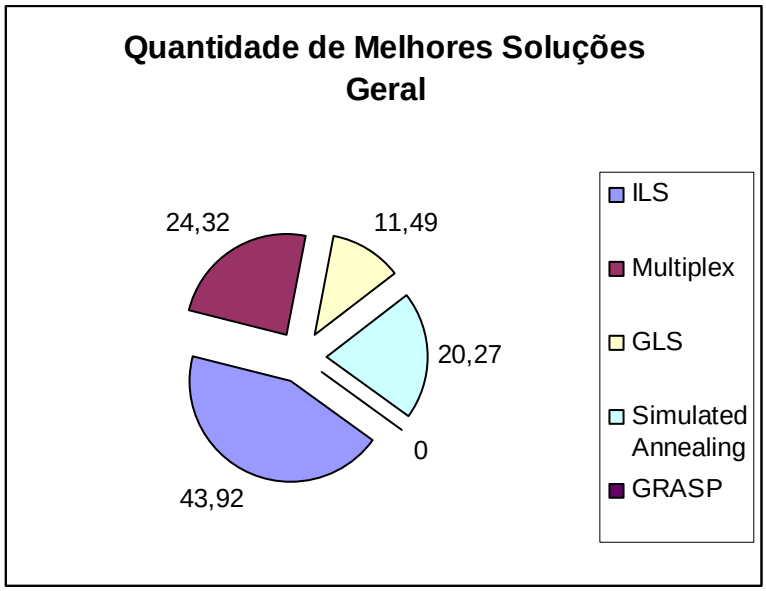

Gráfico 5.1.5.1 – Melhores Soluções Encontradas

Para uma análise ainda mais completa, será feita uma comparação dos 2º, 3º, 4º e 5º lugares para arquivos grandes, ou seja, todos exceto os jnhX.sat. A seguir, são mostrados gráficos comparando as heurísticas utilizadas:

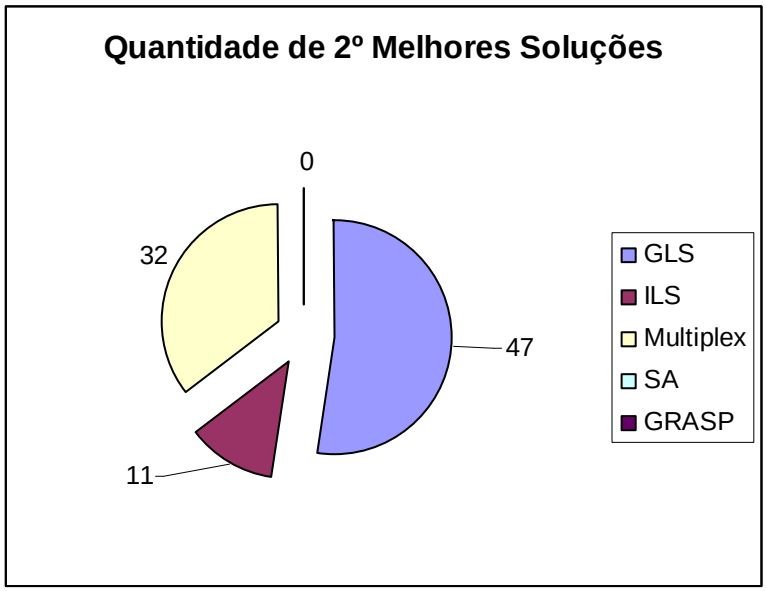

Gráfico 5.1.5.2 – 2º Melhores Soluções Encontradas

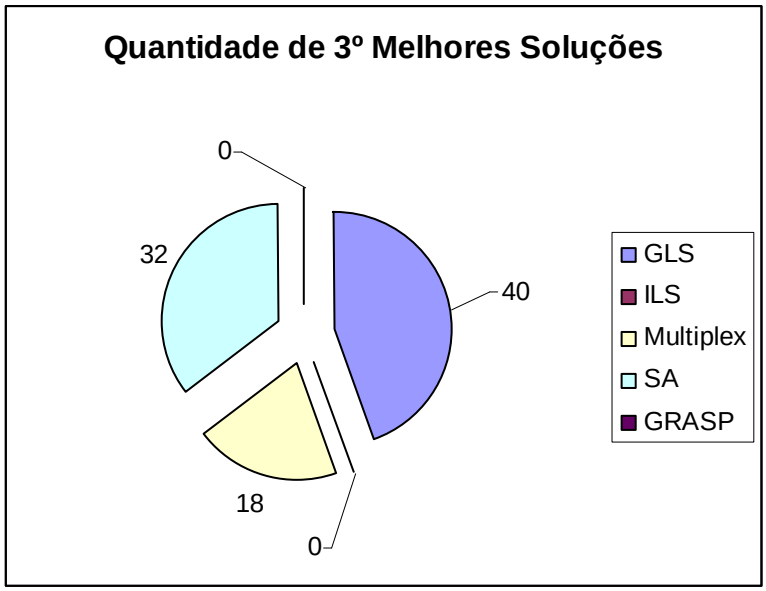

Gráfico 5.1.5.3 – 3º Melhores Soluções Encontradas

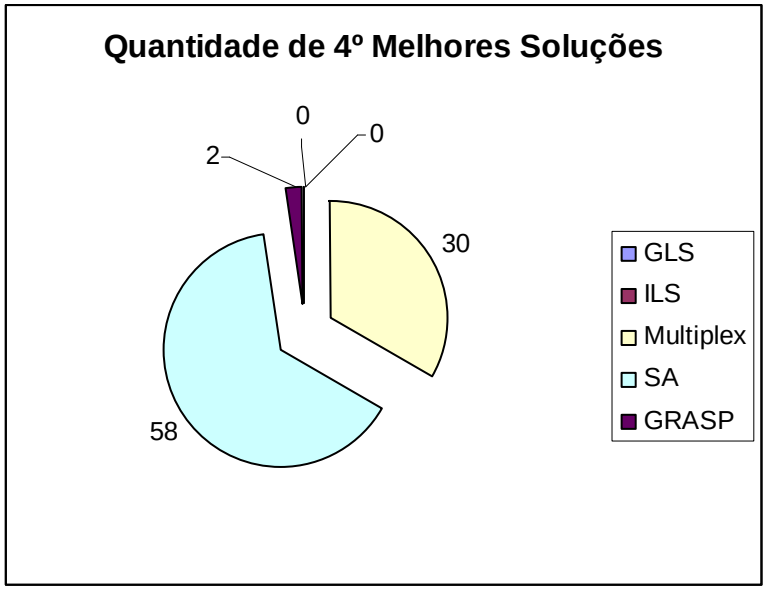

Gráfico 5.1.5.4 – 4º Melhores Soluções Encontradas

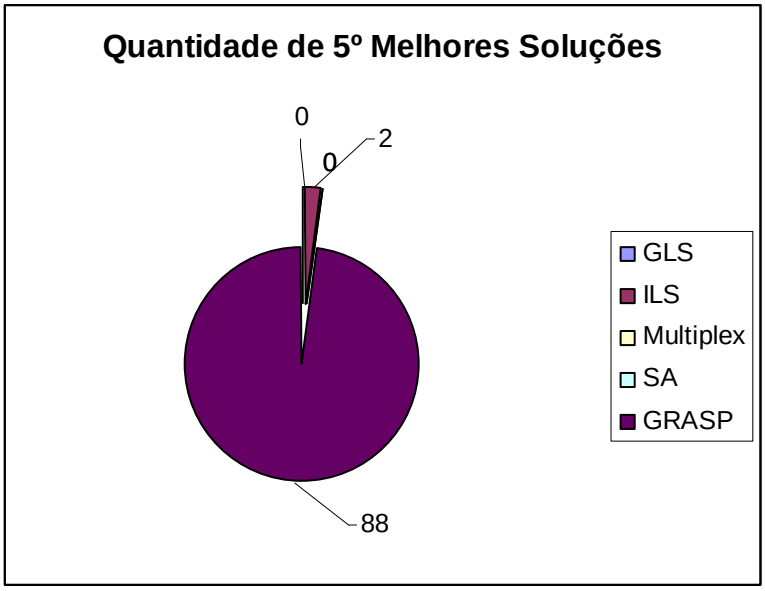

Gráfico 5.1.5.5 – 5º Melhores Soluções Encontradas

Como foram desprezados os arquivos pequenos, os jhnX.sat, os resultados do GLS o colocaram na segunda colocação, seguido do Multiplex que teve menos 3os lugares, mas teve muitos 2os, o SA vem em 4º e o GRASP ficou na última colocação. A surpresa fica por conta do ILS que possui 2 piores soluções, perdendo até mesmo para o GRASP, mas isso não é o suficiente para retira-lo da 1ª colocação.

6. CONCLUSÃO

O ILS continua sendo o melhor algoritmo, dentre os testados, aplicado ao problema MAX-SAT Ponderado. Mas o Multiplex é altamente promissor, pois obteve um desempenho superior ao GLS para os problemas maiores e superior ao SA para problemas menores.

É possível que outro arquivo de processamento obtenha resultados melhores que os relatados, pois o arquivo de processamento é quem define as divisões e a forma de ataque do algoritmo. A melhor escolha da montagem desse arquivo de processamento é objeto de estudos futuros e a melhor calibração do mesmo possibilita melhores resultados no algoritmo.

O Multiplex se mostra uma heurística bastante interessante, pois seu núcleo utiliza conceitos de algoritmos conhecidos e de resultados significativos, mas permite também um novo mecanismo de calibração de forma a adaptar o algoritmo de maneira ainda mais personalizada a cada problema que se deseja utilizá-lo. Além disso, seu desempenho também se mostrou comparável às melhores heurísticas existentes para o problema testado.

Assim como o *Simulated Annealing*, o Multiplex é uma heurística de implementação simples e facilmente adaptável a diversos tipos de problemas, entretanto seu mecanismo de calibração é ainda mais complexo que o do SA, sendo, portanto, este mecanismo ótima fonte de futuros trabalhos como já foi descrito anteriormente.

Não se pode desprezar a possibilidade de paralelização do Multiplex. Devido à relativa independência das divisões, a possibilidade de

paralelização é bastante interessante e facilitada, sendo portanto outra fonte bastante interessante para novos estudos.

Em suma, o Multiplex se mostrou uma heurística de desempenho considerável, fonte para novos estudos e aplicável a diversos problemas.

REFERÊNCIAS

CORMEN, T., **Introduction to algorithms**. Cambridge, EUA: Massachusetts Institute of Tecnology, 2001.

JOHNSON D., **Approximation algorithms for combinatorial problems**, Journal of Computer and System Sciences, 9 (1974), pp 256-278.

DORIGO, M., DI CARO, G., **Ant Optimization: A new meta-heuristic**. In Peter J. Angeline, Zbyszek Michalewicz, Marc Shoenauer, Xin Yao and Aly Zalzala, Proceedings of the Congress on Evolutionary Computation, volume 2, pages 1470-1477, Mayflower Hotel, Washington D.C., 6-9 July 1999. IEEE Press.

FEIGE U., GOEMANS M., **Approximating the proper of two proper proof systems with applications to MAX-2SAT and MAX-DICUT**, in Proceedings of the Israel Symposium on Theory of Computation and Systems, 1995, pp. 182 -189.

GOEMANS M., WILLIAMSON D., **A new ¾ approximation algorithm for the maximum satisfability problem**, SIAM Journal of Discrete Mathematics, 7 (1994), pp. 656-666.

GLOVER, F., **Tabu Search**: 1. ORSA. Journal on Computing, 1(3):190-206, Summer 1989.

GOLDBERG, D., **Genetic Algorithms**. Addison Wesley, Reading, 1989.

GU, J., et al. **Algorithms for the satisfability (SAT) problem**: a survey. In Dingzhu Du, Jun Gu, and Panos M. Pardalos, editors, Satisfability Problem: Theory and Applications, volume 35 of DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, pages 19-152.American Mathematical Society, 1997.

KIRKPATRICK, S., GELLATT, D., VECCHI, P. **Simulated Annealing**. Science, 220(671), 1983.

LOURENÇO, H., MARTIN O., STUTZLE T., **A Beginner´s Introdution to Iterated Local Search,** MIC2001, 4th Metaheuristics International Conference, 2001.

MILLS, P., TSANG K. **Guided Local Search for solving SAT and weighted MAX-SAT problems**. JAR: Journal of Automated Reasoning 24, 2000.

RESENDE, M., **Approximate solution of weighted max-sat problems using grasp**. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, 35:393-405, 1996.

www-or.amp.i.kyoto-u.ac.jp/members/yagiura/index-e.html

YAGIURA, M., IBARAKI T. **Efficient 2 and 3-Flip neighborhood search algorithms for the MAX-SAT**. Lecture Notes in Computer Science, 1449, 1998.

YANNAKAKIS M., **On the approximation of maximum satisfability**, in Proceedings of the Third ACM-SIAM Symposium of Discrete Algorithms, 1992, pp 1-9.