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

Giovany Frossard Teixeira
Universidade Federal do Espírito Santo – UFES
Vitória ES
giovanyfrossard@gmail.com

Arlindo Gomes de Alvarenga
Universidade Federal do Espírito Santo – UFES
Vitória ES
agomes@inf.ufes.br

Attílio Provedel
Universidade Federal do Espírito Santo – UFES
Vitória ES
attilio@terra.com.br

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.
Palavras-Chave: Metaheurísticas; MAX-SAT; Multiplex, *Simulated Annealing*.

Abstract

To compute the great solution for an unit of problem MAX-SAT (weighted maximum satisfiability) it is difficult same if each clause only contains two literal. In that article, we will describe the implementation of a new metaheuristic applied to units of problem of the type MAX-SAT, but easily to use in other problems .Our objective is to generate approximate solutions and to compare their results with the one of other heuristic developed already for that problem type, among them the considered "state of the art".
Key words: *Metaheuristic; MAX-SAT; Multiplex; Simulated Annealing.*

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.

2. O PROBLEMA MAX SAT PONDERADO

Assuma que B é o alfabeto de uma linguagem proposicional $\llcorner$. Fórmulas proposicionais, denotadas pelas letras gregas $\alpha, \beta, \Phi, \ldots,$ 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 $\Phi = C_1 \Lambda C_2 \Lambda \ldots \Lambda C_m$, 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:

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

Onde:
L_i : p_i *ou* $\neg p_i$
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:

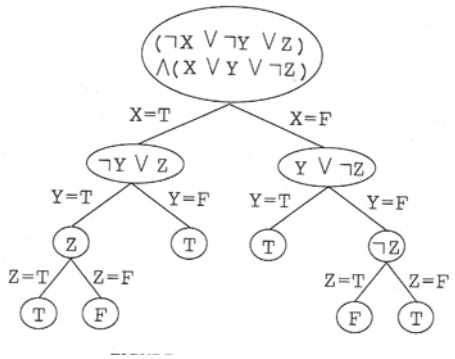


Figura 2.1 - 3-SAT

Neste exemplo, teríamos duas cláusulas:

$\neg x \vee \neg y \vee z$ (cláusula 1)
$x \vee y \vee \neg z$ (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 $k_1, \ldots, k_j, \ldots, k_m$, 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. SIMULATED ANNEALING

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.

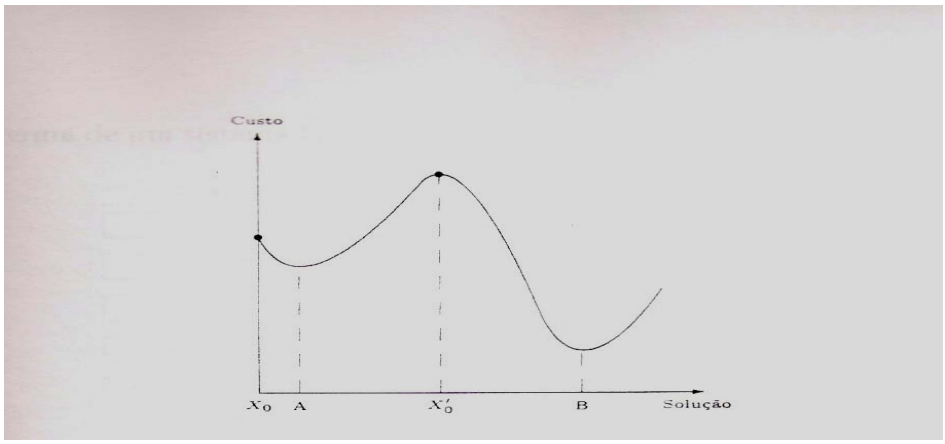


Figura 3.4.1 – Gráfico de Mínimo Local

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*.

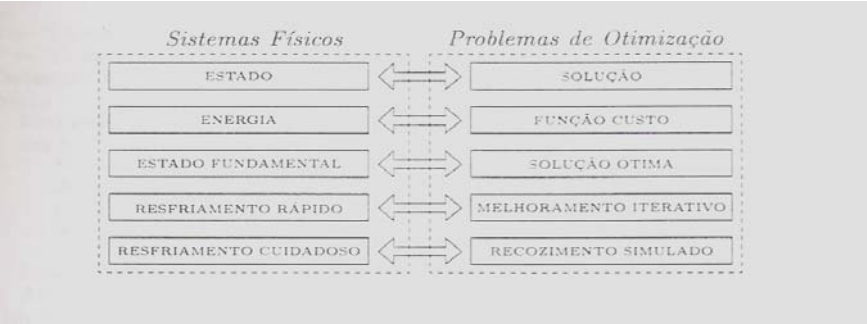


Figura 3.4.2 – Comparação do Problema Físico com o Algoritmo *Simulated Annealing*

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**

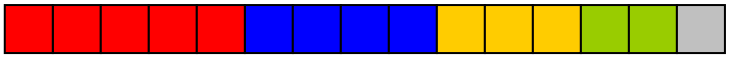
É utilizado um algoritmo para geração de soluções iniciais. Cada solução possui um valor específico que será denominado *rank*, valor este que define a qualidade de uma dada solução. De posse deste valor é possível promover uma ordenação entre as soluções iniciais geradas.

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:



1ª Divisão 2ª Divisão 3ª Divisão 4ª Divisão 5ª Divisão

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.

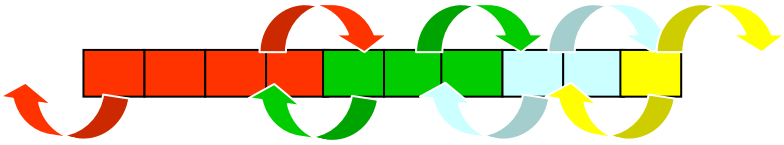


Figura 4.2.3 – Transferência de unidades de processamento móveis

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.

- **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

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.

6. 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.

Os resultados obtidos são apresentados nos gráficos a seguir, analisando apenas o aspecto quantitativo das soluções, uma vez que, a diferença percentual destas soluções é muito pequena.

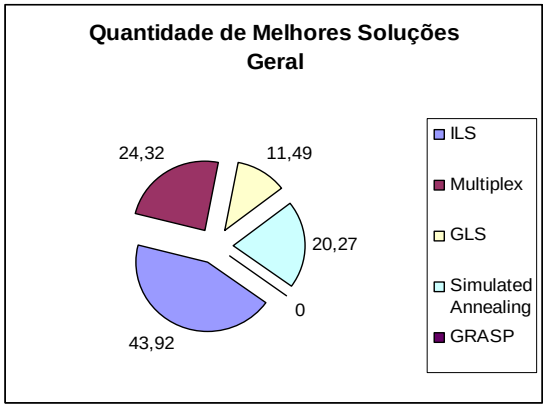


Gráfico 6.1 – Melhores Soluções Encontradas

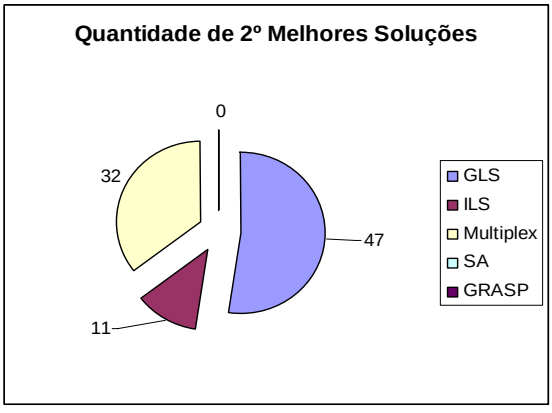


Gráfico 6.2 – 2º Melhores Soluções Encontradas

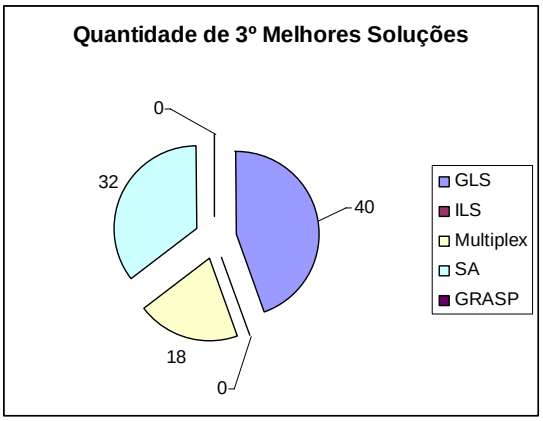


Gráfico 6.3 – 3º Melhores Soluções Encontradas

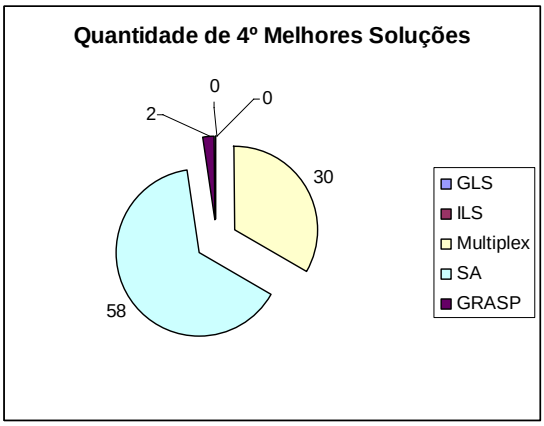


Gráfico 6.4 – 4º Melhores Soluções Encontradas

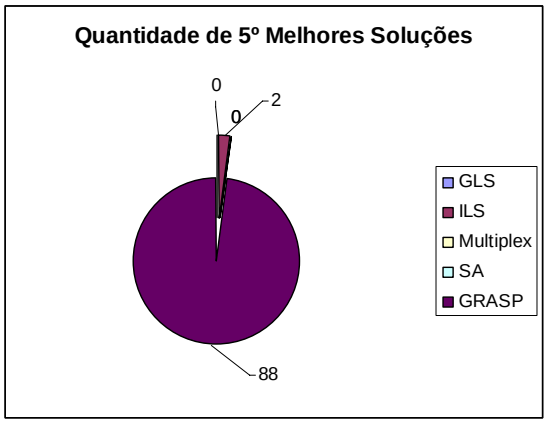


Gráfico 6.5 – 5º Melhores Soluções Encontradas

7. 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 BIBLIOGRÁFICAS

[1] CORMEN, T., **Introduction to algorithms**. Cambridge, EUA: Massachusetts Institute of Tecnology, 2001.
[2] JOHNSON D., **Approximation algorithms for combinatorial problems**, Journal of Computer and System Sciences, 9 (1974), pp 256-278.
[3] 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.
[4] 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.
[5] GOEMANS M., WILLIAMSON D., **A new ¾ approximation algorithm for the maximum satisfability problem**, SIAM Journal of Discrete Mathematics, 7 (1994), pp. 656-666.
[6] GLOVER, F., **Tabu Search**: 1. ORSA. Journal on Computing, 1(3):190-206, Summer 1989.
[7] GOLDBERG, D., **Genetic Algorithms**. Addison Wesley, Reading, 1989.
[8] 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.
[9] KIRKPATRICK, S., GELLATT, D., VECCHI, P. **Simulated Annealing**. Science, 220(671), 1983.
[10] LOURENÇO, H., MARTIN O., STUTZLE T., **A Beginner´s Introdution to Iterated Local Search,** MIC2001, 4th Metaheuristics International Conference, 2001.
[11] MILLS, P., TSANG K. **Guided Local Search for solving SAT and weighted MAX-SAT problems**. JAR: Journal of Automated Reasoning 24, 2000.
[12] RESENDE, M., **Approximate solution of weighted max-sat problems using grasp**. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, 35:393-405, 1996.
[13] www-or.amp.i.kyoto-u.ac.jp/members/yagiura/index-e.html
[14] YAGIURA, M., IBARAKI T. **Efficient 2 and 3-Flip neighborhood search algorithms for the MAX-SAT**. Lecture Notes in Computer Science, 1449, 1998.
[15] YANNAKAKIS M., **On the approximation of maximum satisfability**, in Proceedings of the Third ACM-SIAM Symposium of Discrete Algorithms, 1992, pp 1-9.