Uma Introdução à Busca Tabu

Uma Introdução à Busca Tabu

André Gomes

Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo, SP, Brasil

RESUMO

Este artigo discorre sobre os fundamentos e principais características de uma meta heurística que vem mudando profundamente (para melhor) nossa habilidade de resolver uma enorme série de problemas em ciência aplicada, negócios e engenharia: a Busca Tabu.

Um de seus aspectos mais úteis é a facilidade com que é possível adaptar uma implementação rudimentar para absorver diversas características adicionais. O método pode ser evoluído a uma grande variedade de níveis de sofisticação e aplicação, fazendo com que a Busca Tabu tenha usos importantes em áreas como: métodos evolucionários e genéticos, melhoras de processos de redes neurais e aplicações práticas em campos variados como economia, física, transporte, dentre outros.

Palavras Chave: Busca Tabu, Heurísticas, Meta-Heurísticas, Otimização.

ABSTRACT

This article explains the grounds and main characteristics of a meta heuristics which is deeply changing (for better) our ability to solve a large series of problems in applied sciences, business and engineering: The Tabu Search.

One of its most useful aspects is the ease with which is possible to adapt a rudimentary implementation to absorb many additional characteristics. The method could be evolved to a great variety of levels of sophistication and applications, doing that Tabu Search has important uses in areas like evolutionary and genetic methods, improvement of neural network processes and practice applications in varied fields like economy, physics, transportation and many others.

Key words: Tabu Search, Heuristics, Meta-Heuristics, Optimization.

1. INTRODUÇÃO

A meta heurística conhecida como Busca Tabu é um procedimento adaptativo auxiliar, que guia um algoritmo de busca local na exploração contínua dentro de um espaço de busca.

Diferente do que pode vir a ocorrer com outros procedimentos, a Busca Tabu não é confundida pela ausência de vizinhos aprimorantes. Isso significa que o método evita retornar a um ótimo local visitado previamente, de forma a superar a otimalidade local e atingir um resultado ótimo ou próximo ao ótimo global.

Partindo de uma solução inicial (figura 1.1), que pode ser escolhida de acordo com algum dentre os vários critérios possíveis (que serão abordados mais adiante neste texto), a busca move-se, a cada iteração, para a melhor solução na vizinhança (figura 1.2), não aceitando movimentos que levem a soluções já visitadas (figura 1.3) por permanecerem armazenadas em uma lista tabu. A lista permanece na memória guardando soluções já visitadas (tabu) durante um determinado espaço de tempo ou certo número de iterações (prazo tabu), que também abordaremos a seguir neste material. Como resultado final é esperado que se encontre um ótimo global ou a solução mais próxima deste (figura 1.4).

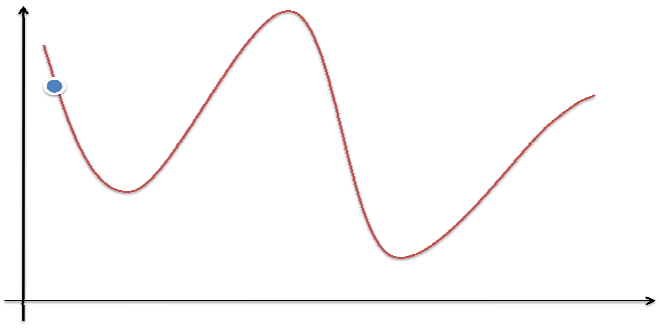

Figura 1.1 – Solução Inicial

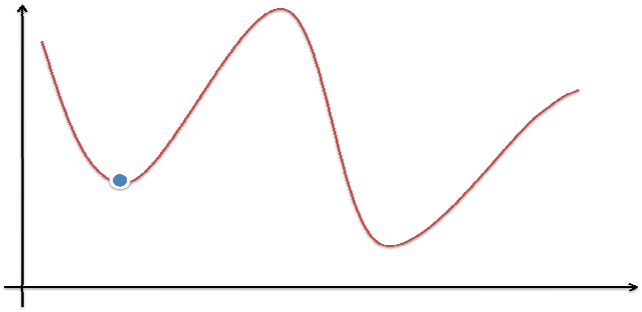

Figura 1.2 – Ótimo Local

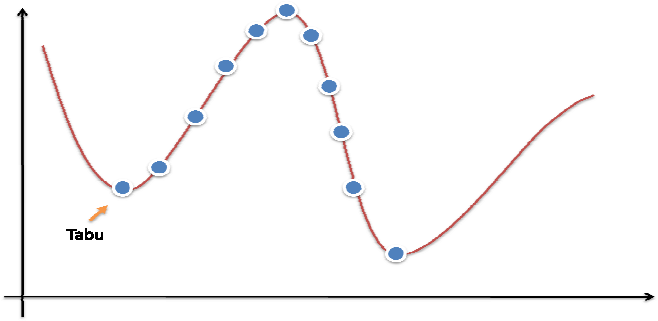

Figura 1.3 – Elementos Tabu

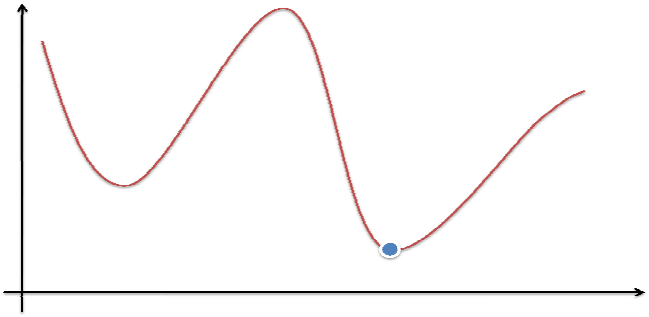

Figura 1.4 – Ótimo Global

A Busca Tabu utiliza estruturas de memória flexíveis para armazenar o conhecimento referente aos espaços que já foram percorridos. Devido ao fato de utilizar intensivamente técnicas de memória adaptativa, a técnica consegue proporcionar "boas" soluções, onde "boas" denota soluções geralmente ótimas ou perto do ótimo global em um tempo de processamento aceitável.

Outra importante característica do método é que a solução final tem pouca ou nenhuma dependência da escolha feita para a solução inicial. Isso graças aos mecanismos implementados pelo método, que fogem de ótimos locais.

1.1 Breve Histórico

Com suas raízes no fim dos anos 60 e começo dos anos 70, a Busca Tabu foi proposta na sua forma atual por Glover [1] em 1986, tornando-se rapidamente em uma forma bem estabelecida de otimização, sendo utilizada hoje em diversas áreas e campos do conhecimento.

Entretanto, aspectos teóricos da Busca Tabu, que elucidassem e comprovassem formalmente sua eficácia foram investigados somente alguns anos após a definição da sua forma atual [3], [4], permanecendo por muito tempo inexplicado o motivo da sua eficiência na busca de soluções ótimas.

A origem da palavra tabu remonta ao Tongan, um idioma da Polinésia, onde era usada pelos nativos da ilha Tonga para indicar objetos que não podiam ser tocados por serem sagrados.

De acordo com o dicionário Aurélio da Língua Portuguesa a palavra também significa "uma proibição imposta por costumes sociais como uma medida de proteção" ou de algo "banido por constituir risco". Estes sentidos atuais e mais pragmáticos da palavra definem melhor a própria Busca Tabu: O risco a ser evitado neste caso é o de seguir um caminho não produtivo, incluindo um que leve ao aprisionamento sem chances de escapatória. Por outro lado, bem como no contexto social onde "proibições protetoras" são passíveis de suspensão quando a ocasião exige, os "tabus" da Busca podem ser retirados quando a situação assim exige como bem veremos em seções posteriores.

A mais importante associação com o uso tradicional da palavra tabu, entretanto, vem do fato que tabus são normalmente transmitidos através da memória social, que está sujeita a mudanças ao longo do tempo. Isto cria a conexão fundamental para o significado de "tabu" em Busca Tabu: os elementos proibidos da Busca Tabu recebem seus status pela dependência em uma memória evolutiva, que permite a este status mudar de acordo com o momento e circunstância.

1.2 META HEURÍSTICAS

São técnicas usadas em situações que podem ser modeladas como problemas de maximizar (ou minimizar) uma função cujas variáveis tem certas restrições.

São estratégias comumente utilizadas para resolver problemas NP - Difíceis por oferecerem melhores soluções e geralmente com tempo de processamento menor do que por outros tipos de técnicas.

De forma geral, utilizam combinação de escolhas aleatórias e conhecimento histórico (dos resultados anteriores adquiridos pelo método) para se guiarem e realizar suas buscas pelo espaço de pesquisa em vizinhanças dentro do espaço de pesquisa, o que evita paradas prematuras em ótimos locais.

1.3 ESPAÇO DE PESQUISA

O espaço de todas as soluções possíveis é chamado de espaço de pesquisa. A procura por uma solução é equivalente a procurar por algum ponto extremo (mínimo ou máximo) neste espaço.

2. DEFINIÇÃO DA BUSCA TABU EM TERMOS GERAIS

De forma simplificada, podemos ver o método como uma técnica iterativa que explora um conjunto de soluções ***X*** de um problema, repetidas vezes, fazendo movimentos de uma solução **s** para outra solução ***s'*** localizada na vizinhança ***N(s)***. (***N*** vem do inglês: neiborhood).

Estes movimentos são executados com o objetivo de alcançar, eficientemente, uma solução qualificada como "boa" (aqui “boa” refere-se a qualquer solução ótima ou próxima desta) pela avaliação de algumas funções objetivas ***f(s)*** a serem minimizadas.

Como ponto de partida inicial, podemos realizar uma comparação da Busca Tabu com um simples método descendente, que é formulado como segue:

(a)

*Escolha uma solução inicial **s** em **X***

***stop** = falso;*

(b)

Faça

Gere um exemplo de soluções V^* *em* $N(s)$

Encontre o melhor s' *em* V^* *(onde* $f(s') \leq f(\bar{s})$ *para qualquer s em* $\bar{V}^*$*).*

Se $f(s') \geq f(s)$ ***então***

$stop$ *= verdade;*

Se não

$s = s'$*;* (2.1)

até $stop$ *== verdade;*

Ao limitar a escolha de s ao melhor elemento de V^* estamos restringindo a classe de métodos descendentes a um subconjunto próprio de coleção maior, algumas vezes chamados de "métodos de melhora".

A forma mais direta de realizar a implementação da aproximação anterior é utilizar $V^* = N(s)$, ou seja, vasculhar toda a vizinhança de s. Em algumas circunstâncias, entretanto, este tipo de processo é impraticável pois possui uma grande complexidade no consumo de tempo. Porém, nestes casos estipulamos que um conjunto $V^* \subset N$ com $|V^*| \ll |N(s)|$ seja analisado. Como forma alternativa de realizar a implementação da solução, podemos fazer com que $|V^*| = 1$, eliminando uma comparação direta entre elementos e não fazendo mais distinção entre o melhor elemento e um membro arbitrário de V^*. Esta ultima forma de amostragem reflete a política adotada pela meta heurística Recozimento Simulado (*simulated annealing*): esta pode ser vista como uma forma de restringir um método descendente, empregando critérios de rejeição que podem resultar na geração de um número de conjuntos com $|V^*| = 1$.

Indo além do caso onde $|V^*| = \mathbf{1}$, a técnica descendente anterior pode ser refinada ao se considerar sua dependência implícita em se resolver um problema de otimização local (tanto por iterização ou por outros métodos mais sofisticados), para achar um s' que leve ao $\mathbf{min.}\{ f(s') \mid s' \in V^* \subseteq N(s) \}$, possivelmente de maneira aproximada. Construído a partir deste ponto de vista, um conceito chave da Busca Tabu é obrigar a definição do problema de otimização local em um modo estratégico – fazendo uso sistemático da memória para explorar conhecimento além do contido na função $f(s)$ e a vizinhança $N(s)$ sobre V^* para ser mais proximamente representativo do mínimo sobre $N(s)$, sujeito à reestruturação especial de $N(s)$. Isso também leva a uma orientação geral à qual o conjunto V^* é para ser gerado estrategicamente de forma um tanto quanto randômica, com o objetivo de permitir que o mínimo de $f(s')$ sobre V^* seja representativamente mais próximo do mínimo sobre $N(s)$, sujeito à reestruturação especial de $N(s)$ através da dependência da memória. Neste sentido,

nós podemos chamar o método de uma meta heurística, pois a cada passo ele usa uma heurística para mover-se de uma solução para a próxima, guiando a busca em ***X***.

Uma busca guiada oferece vantagens sobre as heurísticas descendentes padrão. Por exemplo, a primeira dificuldade que pode surgir é, se a aproximação descendente é aplicada como descrito em (2.1) ela vai ser (na melhor das hipóteses) levada a um mínimo local de ***f*** onde vai ser presa.

Para contornar este efeito, um procedimento guia (*guidance procedure*) deve ser capaz de aceitar um movimento de ***s*** para ***s'*** (em **V***) mesmo se **f**(**s'**) > **f**(**s**). Mas quando uma solução ***s'*** pior que ***s*** pode ser aceita, um *looping* pode ocorrer, levando novamente o processo a ser preso por retornar repetidamente à mesma solução de ótimo local (talvez após alguns intervalos de soluções intermediárias). O Recozimento Simulado tenta guiar o procedimento por aceitar uma piora de **s'** com certa probabilidade que depende de ***f***(***s***), ***f***(***s'***) e um parâmetro identificado como temperatura.

Por outro lado, a técnica da Busca Tabu tenta evitar antecipadamente o perigo de ficar preso ao incorporar uma estrutura de memória que proíbe ou penaliza certos movimentos que poderiam retornar a uma solução visitada recentemente. A variante chamada Busca Tabu Probabilística (*Probabilistic Tabu Search*) cria probabilidades de aceitar movimentos como uma função mono tônica de avaliações criada por uma penalidade do método.

A noção do uso de memória para proibir certos movimentos (i.e. por torná-los tabu) pode ser geralmente formalizada dizendo que a vizinhança da solução depende da linha do tempo, portanto da iteração de número ***k***. Isto é, ao invés de ***N***(***s***) nós podemos nos referir a uma vizinhança denotada ***N***(***s*** , ***k***). (Uma notação mais precisa poderia indicar que ***N***(***s*** , ***k***) depende de soluções e vizinhanças encontradas em iterações prévias a ***k***). Por exemplo, suponha que a memória é empregada para recordar soluções transitórias em um determinado período de tempo e cria ***N***(***s*** , ***k***) por deletar de ***N***(***s***) cada solução que tem um predecessor imediato de ***s*** em uma destas transições. Podemos expressar a forma do procedimento que usa os vizinhos modificados como segue:

(a)

Escolha uma solução inicial ***s*** *em* ***X***

s* = s *e* ***k = 1****;*

(b)

Enquanto *a condição de parada não é atingida, faça*

k = k + 1;

gere $V^* \subseteq N(s, k)$

$s = s'$

Se f(s ') < f(s*)

$s^* = s'$ *(2.2)*

Fim Enquanto

Existem inúmeras condições de parada em uso pelas diversas aplicações disponíveis atualmente. Entretanto, as mais significativas estão descritas a seguir. Normalmente todas as demais condições em uso são variações ou combinações destas:

- Uma solução ótima é encontrada.
- $N(s, k + 1) = \emptyset$.
- k é maior que o número máximo de iterações permitido.
- O número de iterações executadas desde que s^* mudou pela última vez é maior que o número máximo de iterações especificado.
- Pára-se quando o valor da melhor solução chega a um limite inferior conhecido (ou próximo dele). Isto evita execuções desnecessárias ao encontrar uma solução ótima suficientemente "boa".

No procedimento acima, um exemplo não-trivial e estrategicamente gerado da vizinhança da solução é examinado a cada passo e o melhor elemento não tabu do exemplo é selecionado. O objetivo é fazer movimentos de melhora até a máxima extensão permitida pela estrutura de $N(s, k)$, balanceando as soluções fornecidas entre qualidade do resultado e esforço computacional ao examinar grandes amostras.

Caracterizada desta maneira, a Busca Tabu pode ser vista como um método de vizinhança variável: cada passo redefine a vizinhança pela qual a próxima solução vai ser desenhada, baseada nas condições que classificam certos movimentos como tabu.

Um aspecto crucial do procedimento envolve a escolha de uma definição apropriada de $N(s, k)$. Devido à exploração da memória, $N(s, k)$ depende da trajetória seguida ao se mover de uma solução para a próxima. Como ponto de partida, considere uma forma de memória encarnada em uma lista tabu T que se recorda das $|T|$ soluções visitadas mais recentemente, levando $N(s, k) = N(s) - T$.

Uma aproximação baseada em dados recentes na memória (*recencia da memória*) irá prevenir que ciclos de tamanho menor ou igual à $|T|$ ocorram na trajetória da busca.

Esta formulação normalmente tem somente interesse teórico, pois manter as soluções mais recentes de $|T|$ pode consumir um grande volume de espaço na memória. (Isto também representa uma forma relativamente limitada de memória baseada em dados recentes).

Como base para identificar uma estrutura de memória mais prática e efetiva podemos melhorar nossa definição de T ao definir a vizinhança $N(s)$ em termos de movimentos que transformam a solução s em novas soluções. Especificamente, para cada solução **s** considere um conjunto $M(s)$ de movimentos permitidos m que podem ser aplicados a s para obter uma nova solução $s' = s \oplus m$. Então temos $N(s) = \{ s' \mid \exists\, m \in M(s)$ com $s' = s \oplus m \}$. Geralmente é útil empregar um conjunto de movimentos $M(s)$ que sejam reversíveis, isto é, para cada $m \in M(s)$, $\forall\, m^{-1} \in M(M(s))$ tal que $(s \oplus m) \oplus m^{-1} = s$. Com esta estipulação, T pode consistir dos últimos $|T|$ "movimentos reversos" associados com os movimentos já executados.

Ao invés de uma única lista tabu T pode ser mais conveniente usar diversas listas T_j de m (ou de s) para indicar que estes movimentos já efetuados são atualmente proibidos para compor um novo movimento. Geralmente o status tabu de um movimento é uma função do status tabu dos atributos que mudam ao ir da solução atual para a próxima (como variáveis que mudam os valores, ou elementos que mudam posições, ou itens que são transferidos de um conjunto a outro).

Isto leva a uma coleção de condições Tabu da forma

$$t_j(m, s) \in T_i \ (i = 1, \ldots, p) \qquad (2.3)$$

Um movimento m aplicado às soluções correntes vai ser então um movimento tabu (isto é, implicitamente um membro da lista T) se todas as condições em (2.3) são satisfeitas.

A seguir é mostrado o pseudocódigo do algoritmo da Busca Tabu na Tabela 2.1:

PROCEDIMENTO DA BUSCA TABU

procedimento BT

01. *Seja s_0 solução inicial;*

02. *$s^* \leftarrow s$; {Melhor solução obtida até então}*

03. *Iter $\leftarrow 0$; {Contador do número de iterações}*

04. *MelhorIter $\leftarrow 0$; {Iteração mais recente que forneceu s^*}*

05. *Seja BTmax o número máximo de iterações sem melhora em s^*;*

06. *T ← ∅; {Lista Tabu}*

07. *Inicialize a função de aspiração A;*

08. *enquanto (Iter – MelhorIter ≤ Btmax) faça*

09. *Iter ← Iter + 1;*

10. *Seja s' ← s ⊕ m o melhor elemento de V ⊆ N (s) tal que o movimento m não seja tabu (m ∉ T) ou s' atenda a condição de aspiração (f(s') < A(f(s)));*

11. *Atualize a Lista Tabu T;*

12. *s ← s';*

13. *se f(s) < f(s*) então*

14. s ← s;*

15. MelhorIter ← Iter ;

16. *fim-se;*

17. *Atualize a função de aspiração A;*

18. *fim - enquanto;*

19. *Retorne s*;*

fim BT;

Tabela 2.1 – Procedimento Busca Tabu

3. EFICIÊNCIA DE MÉTODOS DE SOLUÇÃO ITERATIVOS

Ao trabalhar com métodos de solução iterativos temos de ter uma mente que a sua eficiência depende, em sua maior parte, da qualidade da modelagem. Por exemplo, um ajuste fino dos parâmetros do método nunca será capaz de balancear uma má escolha de estrutura de vizinhança da função objetivo. Por outro lado, uma modelagem eficiente deve levar a técnicas robustas que não sejam tão sensíveis a configurações distintas de parâmetros. Nos próximos tópicos serão abordados dois assuntos essenciais para o desenho eficiente de métodos de solução iterativos: Modelagem e Computação eficientes.

3.1 MODELAGEM EFICIENTE

Métodos de solução iterativos podem ser vistos como passeios em um grafo de estados do espaço (*state space graph*) ***G(S , A)*** onde o conjunto de vértices ***S*** é o conjunto de possíveis soluções e existe uma aresta ***(i , j)*** ∈ ***A*** de ***i*** para ***j*** se ***j*** ∈ **N(i)**. A escolha de uma solução inicial é equivalente a gerar um vértice em ***G*** e um passo do procedimento iterativo consiste em mover do vértice corrente ***i*** a um vértice

adjacente a ***i***. Para um problema de otimização dado, normalmente existem várias maneiras para definir o grafo de estados do espaço ***G*** e é essencial escolher um que satisfaça a seguinte condição:

*Dado qualquer **i** em **S**, deve existir um caminho de **i** a uma solução ótima **i***.* (*)

Se uma solução que não satisfaz esta condição é visitada durante o processo iterativo, então uma solução ótima nunca será alcançada.

Para ilustrar este fato, vamos considerar o problema de coloração de grafos: dado um grafo ***H*** = (***V*** , ***E***) temos que encontrar uma coloração de vértices com a menor quantidade de cores possível de tal forma que dois vértices unidos por uma aresta tenham a mesma cor. Podemos definir o conjunto ***S*** de soluções possíveis como o conjunto de colorações possíveis que não usam mais do que um dado número de cores; por exemplo, este limite superior pode ser o número de cores usadas na solução inicial. Vamos definir a vizinhança ***N***(***i***) da solução ***i*** como o conjunto de possíveis colorações que podem ser obtidas por ***i*** ao mudar a cor de exatamente um vértice. Isto induz um grafo de estados do espaço que não satisfaz a condição (*). De fato, considere o exemplo na figura 3.1: se o método de solução iterativo visita a possível cor $\mathbf{c_1}$ e se o limite superior no numero de cores que pode ser usado é três, então uma coloração ótima usando somente duas cores nunca será alcançada. Uma definição melhor do grafo de estados dos espaços ***G*** será descrita na seção 5.

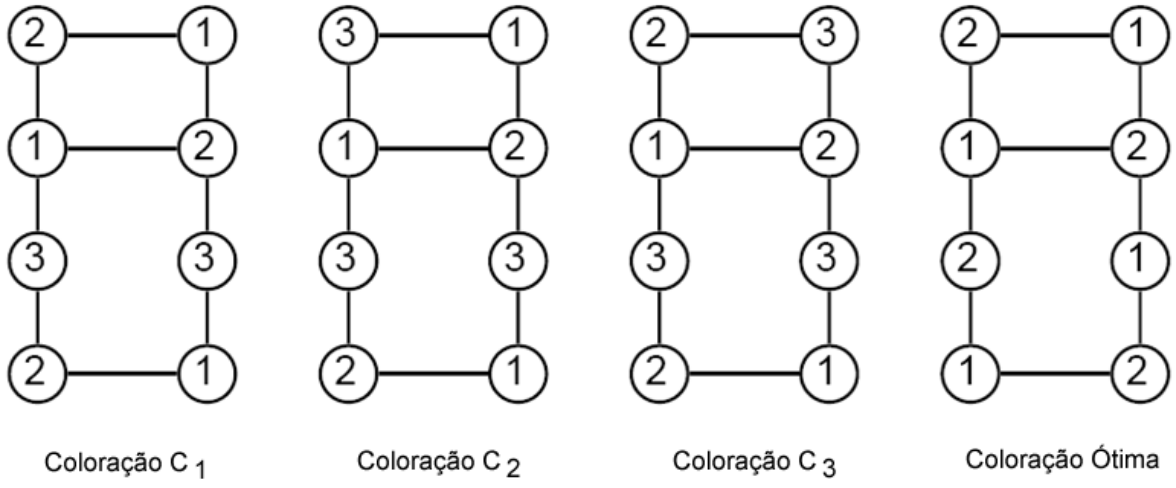

Figura 3.1 – Exemplo de soluções

Neste ponto, é importante relembrar que o conjunto ***S*** de possíveis soluções não é necessariamente o conjunto de soluções do problema original: na verdade, em alguns casos, pode ser muito difícil redefinir a estrutura da vizinhança que satisfaça a

condição **(*)**. Se as soluções do problema original precisam satisfazer um conjunto C de restrições, então algumas vezes é sensato definir S como um conjunto de soluções que satisfaça um subconjunto próprio $C' \subset C$ de restrições. Por exemplo, para o problema de coloração dos grafos, uma adaptação eficiente da Busca Tabu será descrita na seção 5 onde a restrição que dois vértices ligados por uma aresta devem receber cores diferentes é relaxada. Claro, a violação de uma restrição relaxada é penalizada na função objetivo.

Cada vértice i do grafo G tem um valor $f(i)$. Considerando estes valores como altitudes, isso induz uma topologia com vales contendo ótimos locais. É importante definir f de forma que não existam muitos vales ou grandes planaltos. Nestes casos é muito difícil guiar a busca pela solução ótima, que é o topo de um dos vales mais profundos.

Para o problema de coloração de grafos, definir o valor $f(i)$ de uma coloração i como o número de cores que são usadas em i induz planaltos muito grandes se todas as soluções tiverem um valor inteiro que pertence ao pequeno intervalo $[f(i^*), u]$, onde $f(i^*)$ é o valor para uma solução ótima e u é o limite superior.

Como mencionado anteriormente, a função objetivo f pode ser alterada durante a busca. Estas mudanças correspondem à modificação da topologia. Durante a fase de diversificação, por exemplo, é tentado substituis os vales já visitados por altas montanhas. Assim a busca é direcionada através de regiões inexploradas.

3.2 COMPUTAÇÃO EFICIENTE

A cada passo de um método de solução iterativo, muitas soluções têm de ser avaliadas e é importante executar essa computação de uma forma eficiente.

Para uma solução i e um movimento m, é mais simples computar $\Delta(i, m)$ definido como $f(i \oplus m) - f(i)$ do que $f(i \oplus m)$. Em alguns casos é possível armazenar os valores $\Delta(i, m)$ para todo $m \in M(i)$. Neste caso, se um movimento m pode ser aplicado a duas soluções consecutivas i e j, $(m \in M(i) \cap M(j))$. Acontece freqüentemente que $\Delta(j, m)$ pode ser computado de forma ainda mais fácil. Por alguns problemas, dada uma solução i e um movimento $m \in M(i)$, é difícil de computar $f(i \oplus m)$. Por exemplo, para o problema de roteamento de veículos, vamos definir uma solução i como uma distribuição dos usuários aos veículos e um movimento m como a transferência de exatamente um usuário de um veículo para outro. Se o objetivo é minimizar a distância total viajada, a computação $f(i \oplus m)$ é equivalente a resolver vários problemas do caixeiro viajante. Como é sabido, o problema do caixeiro viajante é um problema NP - difícil.

Nestes casos é necessário utilizar métodos de heurística velozes (que não tenham grande complexidade de tempo) para avaliar as soluções na vizinhança. Uma vez que a melhor solução vizinha tenha sido determinada, uma técnica mais elaborada pode ser utilizada para melhorar o resultado. Por exemplo, para o problema de roteamento de veículos, um método simples de inserção pode ser usado para encontrar a melhor vizinhança $j \in N(i)$. Todas as rotas de i modificadas para obter j são então re-otimizadas usando métodos de melhora como o r-opt [17] ou mesmo métodos exatos quando não existem muitos usuários nas rotas.

No caso onde todos os valores $\Delta(i, m)\ \forall m \in M(i)$ podem ser armazenados, às vezes é possível manter uma lista sortida destes valores com um pequeno esforço computacional.

4. USO EFICIENTE DE MEMÓRIA

No fim da seção 2 é dada uma descrição do procedimento da Busca Tabu. Nela é possível perceber que o uso intensivo de memória é uma característica essencial deste método. As condições Tabu podem, usualmente, ser consideradas como memória de curto prazo que previne ciclagem (*loopings)* por alguma extensão na iterização. Agora, nesta seção, descrevo algumas políticas eficientes para o gerenciamento de listas tabu. Também será mostrado como o uso de memória pode ajudar a intensificar a busca em regiões "boas" ou diversificar a busca através de regiões inexploradas.

4.1 LISTAS TABU DINÂMICAS

As Listas Tabu Dinâmicas tem como principal objetivo evitar que o algoritmo entre em processo de ciclagem (*looping)*. Esta situação pode ocorrer devido a inúmeros fatores. Por exemplo, se a lista tabu tem tamanho três, mas somente no quinto passo é achado um valor de melhora. Neste caso a lista retorna ao ótimo local prévio e começa a busca novamente. (Neste caso a busca seria encerrada após algumas repetições deste movimento, pois atingiria o critério de parada que diz que após certo número de iterações sem melhora o algoritmo deve ser encerrado) .

Para evitar isso, as listas dinâmicas têm tamanho variável no intervalo $[t_{min}, t_{max}]$. O tamanho deve ser mudado periodicamente. Um valor muito utilizado é realizar a mudança a cada $2t_{maxiterações}$.

4.2 PASSAGENS POR REGIÕES PLANAS

Passar por regiões planas pode levar o algoritmo a pensar que não há melhora significativa na qualidade das soluções e atingir o critério de parada (não encontrar soluções de melhora por um grande número de iterações, por exemplo). Mas muitas vezes isso é apenas uma conclusão artificial e após uma longa região plana podem haver resultados aprimorantes.

Para tentar evitar esta situação é utilizada uma técnica que aumenta o tamanho da lista tabu enquanto o algoritmo estiver passando pela região plana e volta a reduzir seu tamanho (ao original, por exemplo), quando houver uma mudança no valor da função de avaliação.

4.3 INTENSIFICAÇÃO

São técnicas utilizadas para concentrar os esforços da pesquisa em regiões consideradas promissoras. É muito comum que métodos de Busca Tabu incluam estratégias de intensificação para melhorar seu desempenho e a qualidade da resposta fornecida.

As duas principais estratégias são:

- Retornar a uma solução já visitada para explorar sua vizinhança de forma mais efetiva e,
- Incorporar atributos das melhores soluções já encontradas durante o progresso da pesquisa e estimular componentes dessas soluções a tornar parte da solução corrente.

São consideradas livres no procedimento de busca local apenas as componentes não associadas às boas soluções, permanecendo as demais componentes fixas.

Um procedimento de intensificação com estas características é a Reconexão por Caminhos (*Path-Relinking*).

4.4 DIVERSIFICAÇÃO

Métodos baseados em Busca Tabu incluem também, estratégias de diversificação. O objetivo dessas estratégias, que tipicamente utilizam uma memória de longo prazo, é redirecionar a pesquisa para regiões ainda não suficientemente exploradas do espaço de soluções.

Estas estratégias procuram, ao contrário das estratégias de intensificação, gerar soluções que têm atributos significativamente diferentes daqueles encontrados nas melhores soluções obtidas até o momento.

A diversificação, em geral, é utilizada somente em determinadas situações, como, por exemplo, quando dada uma solução ***s***, não existem movimentos ***m*** de melhora para ela, indicando que o algoritmo já exauriu a análise naquela região.

4.5 PATH RELINKING

É uma estratégia de intensificação que objetiva incorporar atributos de soluções de boa qualidade (chamadas de soluções elite), favorecendo a seleção de movimentos que as contenham. Foi originalmente proposta por Fred Glover em 1996 no contexto de Busca Tabu ou *Scatter Search*.

Basicamente, consiste em gerar e explorar caminhos no espaço de soluções partindo de uma ou mais soluções elite e levando a outras soluções elite.

Para tanto, são selecionados movimentos que introduzem atributos das soluções guia na solução corrente. O caminho, então, é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o melhor movimento é selecionado.

5. EXEMPLOS

Métodos clássicos de otimização muitas vezes encontram grande dificuldade quando são confrontados com a tarefa de resolver problemas de otimização difíceis que façam parte do mundo real. Aplicações vitais em negócios, engenharia, economia e ciência não conseguem ser abordadas com esperança razoável de sucesso dentro de janelas de tempo práticas, por métodos de solução que tenham tido foco predominante em pesquisa acadêmica nas três décadas passadas [6].

Por outro lado, a Busca Tabu está ampliando consideravelmente nossa habilidade de resolver problemas com significados práticos reais. Aplicações atuais da Busca Tabu abrangem ramos de planejamento de recursos, telecomunicações, VLSI design, análise financeira, agendamento, planejamento de espaço, distribuição de energia, engenharia molecular, logística, classificação de padrões, fabricação flexível, gerenciamento de gastos, exploração de minérios, análises biomédicas, preservação ambiental, dentre

muitas outras aplicações. Nos últimos anos, diversas publicações em uma grande variedade de áreas têm disponibilizado artigos, tutoriais e estudos computacionais documentando sucessos da Busca Tabu em estender a fronteira de problemas que podem ser lidados de forma eficiente - levando a soluções cuja qualidade se sobrepõe significantemente a aquelas obtidas por métodos aplicados previamente. A tabela 5.1 mostra um catálogo parcial de exemplos de aplicações. Uma lista mais compreensiva, incluindo descrições de ganhos atingidos pelas implementações práticas, bem como relatórios de implementações recentes de BT podem ser encontradas em [15].

Aplicações da Busca Tabu	
Design	**Scheduling**
Design assistido por computador	Flow-Time Cell Manufacturing
Redes tolerantes a falhas	Heterogeneous Processor Scheduling
Design de redes de transporte	Planejamento de força de trabalho
Architectural Space Planning	Escalonamento de Cursos
Coerência de diagramas	Escalonamento em Máquinas
Fixed Charge Network Design	Flow Shop Scheduling
Problemas de Corte Irregular	Job Shop Scheduling
	Sequencing and Batching
Produção, Inventário e Investimento	
Flexible Manufacturing	**Telecomunicações**
Produção Just-in-Time	Roteamento de Chamadas
Capacitated MRP	Bandwidth Packing
Part Selection	Hub Facility Location
Multi-item Inventory Planning	Path Assignment
Volume Discount Acquisition	Network Design for Services
Fixed Mix Investment	Arquitetura Imune a Falhas
	Synchronous Optical Networks
Roteamento	
Roteamento de Veículos	**Lógica e Inteligência Artificial**
Capacitated Routing	Maximum Satisfiability
Roteamento de Janelas de Tempo	Lógica Probabilística
Roteamento Multi Modo	Clusterização
Mixed Fleet Routing	Reconhecimento / Classificação de
Caixeiro Viajante	Padrões Integridade de Dados
Traveling Purchaser	Redes Neurais: Treinamento e Design
Otimização de Grafos	**General Combinational Optimization**
Particionamento de Grafos	Programação Zero - Um
Coloração de Grafos	Fixed Charge Optimization
Particionamento de Clique	Nonconvex Nonlinear Programming
Problemas de Clique Máximo	Redes Tudo-ou-Nada
Maximum Planner Graphs	Programação Bilevel

Problemas P-Medianos	General Mixed Integer Optimization
Tecnologia	**Location and Allocation**
Seismic Inversion	Multicommodity Location/Allocation
Distribuição de Energia Elétrica	Quadratic Assignment
Engineering Structural Design	Quadratic Semi-Assignment
Minimum Volume Ellipsoids	Multilevel Generalized Assignment
Space Station Construction	Planejamento de Lay-Out
Circuit Cell Placement	Exploração de Petróleo em Alto Mar

Tabela 5.1 – Aplicações da Busca Tabu

A seguir, alguns dos conceitos anteriores serão relembrados com exemplos simples onde o procedimento da Busca Tabu produziu soluções muito boas.

5.1 O PROBLEMA DE COLORAÇÃO DE GRAFOS

Mais detalhes sobre a resolução deste problema podem ser encontrados em [5].

Dado um grafo $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ temos que encontrar uma coloração de seus vértices com a menor quantidade de cores possível. A única restrição é que dois vértices ligados por uma aresta não recebam a mesma cor.

Em outras palavras, uma coloração de vértices de $\boldsymbol{G}$ em $\boldsymbol{k}$ cores é uma partição do conjunto de vértices $\boldsymbol{V}$ em $\boldsymbol{k}$ conjuntos independentes $\boldsymbol{V_1}, \ldots, \boldsymbol{V_k}$. Para definir o conjunto **S** de possíveis soluções, a única restrição foi relaxada e Busca Tabu utilizada para encontrar se é possível uma coloração em um dado número **k** de cores. Isto é, uma solução possível é qualquer partição $(\boldsymbol{V_1}, \ldots, \boldsymbol{V_k})$ de $\boldsymbol{V}$ em $\boldsymbol{k}$ conjuntos. Ao definir $\boldsymbol{E}(\boldsymbol{V_v})$ como o conjunto de arestas tendo as duas pontas em $\boldsymbol{V_v}$, o objetivo é minimizar $\sum_{\boldsymbol{r=1}}^{\boldsymbol{k}} \boldsymbol{E} \mid \boldsymbol{V}(\boldsymbol{r}) \mid$.

Portanto uma solução de valor **zero** corresponde a uma coloração de $\boldsymbol{G}$ em $\boldsymbol{k}$ cores. Como mencionado anteriormente, definindo $\boldsymbol{S}$ como o conjunto de todas as partições de **V** em conjuntos independentes (que é a única restrição do problema não relaxado) e a função objetivo como o número de cores usadas implica que um número maior de movimentos (que mudam a cor de um único vértice) tem o mesmo valor (e muito poucos de aumento). É difícil guiar a busca nesta situação.

Na modelagem eficiente da Busca Tabu descrita acima a vizinhança $\boldsymbol{N}(\boldsymbol{i})$ de uma solução $\boldsymbol{i}$ consiste em todas as soluções geradas por $\boldsymbol{i}$ pela modificação local: mova um vértice que tem ponta em uma aresta monocromática (ambas as pontas estão no mesmo $\boldsymbol{V_v}$) para um conjunto $\boldsymbol{V_q}$ $(\boldsymbol{q} \neq \boldsymbol{r})$. Quando um vértice é movido de $\boldsymbol{V_v}$ à $\boldsymbol{V_q}$ o par

(***v*** , ***r***) é introduzido na lista tabu ***T***. Isto significa que é proibido introduzir ***v*** em V_v durante **|*T*|** iterações.

5.2 Problemas de Escalonamento de Cursos

Mais detalhes sobre este problema podem ser encontrados em [8], [9], [14].

Basicamente, problemas de escalonamento de cursos podem ser formulados como problemas de coloração de grafos: os cursos são os vértices de um grafo, os horários correspondem às cores e dois vértices são conectados por uma aresta se as matérias não podem ser dadas ao mesmo tempo. Na vida real muito mais restrições devem ser levadas em conta e a maioria das técnicas de otimização mal pode lidar com os requerimentos de cada escola (o que caracteriza este problema como NP – Difícil).

A BT vem sido aplicada com sucesso a este problema através de uma extensão da sua adaptação do problema de coloração de grafos. Uma solução viável é definida como um escalonamento de aulas que satisfaça um subconjunto C de restrições e uma solução vizinha é obtida ao modificar o horário de uma matéria. Se todas as matérias duram um horário, isto é equivalente a mudar a cor de um vértice. Para decidir se uma restrição específica deve ser incluída em C, as seguintes orientações podem ser usadas: uma matéria nunca deve ser atribuída a um período se esta atribuição induz somente escalonamentos de cursos que não satisfazem todas as restrições. Além disso, um conflito entre duas matérias não deveria ser uma condição suficiente para dificultar uma atribuição.

Por exemplo, vamos considerar dois cursos c_1 e c_2 envolvendo estudantes em comum. Se $\mathbf{c_1}$ não pode ser atribuído no período ***p*** por alguma razão (talvez o professor não esteja disponível para aquele horário, por exemplo), então não devemos nunca visitar uma solução onde c_1 é dada no período ***p***. Entretanto, se esta restrição não existe e c_2 é atribuído ao período ***p***, então nós devemos permitir a atribuição de $\mathbf{c_1}$ no período ***p***, mesmo se isto violar a restrição de não sobreposição de cursos envolvendo estudantes em comum. A razão é que c_1 é possivelmente dado no período ***p*** em um escalonamento ótimo e a violação da restrição deve ser cancelada movendo a matéria c_2 para outro período. Como mencionado em [9], este modelo pode lidar com problemas de escalonamento de cursos com matérias opcionais, janelas no horário e pré-requisitos para as matérias, restrições de espaço e precedência e os cursos podem ter durações diferentes.

5.3 O PROBLEMA DO CONJUNTO INDEPENDENTE MÁXIMO

Maiores detalhes sobre a implementação deste problema podem ser encontrados em [12] e [13].

Dado um grafo $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ temos que determinar um subconjunto $\mathbf{X}$ de $\mathbf{V}$ tão grande quanto possível tal que $\mathbf{E}(\mathbf{X}) = \emptyset$ (onde $\mathbf{E}(\mathbf{X})$ é um conjunto de arestas tendo ambas as terminações em $\mathbf{X}$).

A Busca Tabu foi adaptada para tentar determinar um conjunto independente em $\mathbf{G}$ de um tamanho dado $\mathbf{k}$. Mais uma vez, a única restrição do problema foi relaxada. Uma solução é qualquer subconjunto $\mathbf{X}$ de $\mathbf{V}$ de tamanho $\mathbf{k}$ e a função objetivo é $|\mathbf{E}(\mathbf{X})|$. A vizinhança consiste em trocar um vértice $\mathbf{v} \in \mathbf{X}$ com um vértice $\mathbf{w} \in \mathbf{V} - \mathbf{X}$. O tamanho da vizinhança da solução é igual à $\mathbf{k}(\mathbf{n} - \mathbf{k}) \in \mathbf{O}(\mathbf{nk})$ onde $\mathbf{n} = |\mathbf{V}|$.

Para um vértice $\mathbf{v} \in \mathbf{V}$, deixe-nos denotar $\Gamma_{(\mathbf{X})}(\mathbf{v})$ o conjunto de vértices $\mathbf{w}$ em $\mathbf{X}$ que são conectados à $\mathbf{v}$ por uma aresta. Os vértices $\mathbf{v}$ em $\mathbf{X}$ são distribuídos de acordo com os valores não crescentes de $\boldsymbol{\Gamma}$ e os vértices $\mathbf{w} \in \mathbf{V} - \mathbf{X}$ são ordenados de acordo com valores não decrescentes de $\boldsymbol{\Gamma}$. Manter estas duas listas ordenadas requer tempo $\mathbf{O}(\mathbf{n})$. Três listas Tabu são usadas. A primeira $\mathbf{T}_1$ armazena as ultimas soluções visitadas, a segunda $\mathbf{T}_2$ contém os últimos vértces introduzidos em $\mathbf{X}$ e a última $\mathbf{T}_3$ contém todos os últimos vértices removidos de $\mathbf{X}$. Como mostrado em [12], ao usar técnicas de hashing para implementar $\mathbf{T}_1$ e escolhendo um valor de constante pequeno para $|\mathbf{T}_2|$ e $|\mathbf{T}_3|$, um melhor vizinho pode ser encontrado na pratica em tempo constante. Por isso um passo de BT leva tempo $\mathbf{O}(\mathbf{n})$ ao invés de $\mathbf{O}(\mathbf{nk})$.

Uma adaptação completamente diferente de BT para o problema do conjunto independente máximo foi descrita em [13]. BT é usada para obter limites em um algoritmo de "*branch & bound*". Em cada nó do algoritmo definimos dois conjuntos $\boldsymbol{I}$ e $\boldsymbol{O}$ de vértices que são forçados a estar respectivamente dentro e fora de um conjunto interdependente máximo. Seja $\boldsymbol{X}$ o maior conjunto encontrado até agora. BT é usada para cobertura com $|\boldsymbol{X}| - |\boldsymbol{I}|$ cliques para tantos vértices quanto possível no grafo $\boldsymbol{H}$ induzido por $\boldsymbol{U} = \boldsymbol{V} - (\boldsymbol{I} \cup \boldsymbol{O})$. Deixe $\boldsymbol{W}$ ser o conjunto de vértices cobertos pelos cliques. Se $|\boldsymbol{W}| = |\boldsymbol{U}|$ então *backtracking* ocorre desde que $\boldsymbol{H}$ não contenha um conjunto estável de tamanho maior que $|\boldsymbol{X}| - |\boldsymbol{I}|$. De outra maneira, nós geramos um novo subproblema par cada vértice $\boldsymbol{v}$ pertence a $\boldsymbol{U} - \boldsymbol{W}$ ao introduzir $\boldsymbol{v}$ em $\boldsymbol{I}$ e por adicionar em $\boldsymbol{O}$ todos os vértices adjacentes à $\boldsymbol{v}$.

Usando uma técnica menos elaborada para cobrir os vértices de H por cliques requer menos tempo, mas o número de ramificações gerados em cada nó do algoritmo "*branch* & *bound*" é geralmente muito maior. Foi mostrado que vale a pena gastar tempo adicional usando BT em cada nó para reduzir o número de ramificações. Mais detalhes podem ser encontrados em [13].

5.4 FORMULAÇÃO QAP

Prosseguimos analisando o Problema da Atribuição Quadrática [2] (*quadratic assignment problem* ou *QAP*):

N itens devem ser atribuídos à N diferentes localizações. A distância a_{ij} entre as localidades i e j é dada bem como o valor b_{pq} de um fluxo (circulação de partes) entre os itens p e q.

É procurada uma distribuição dos itens às localidades que minimize a soma dos produtos das distâncias e valores de fluxo associados. Matematicamente, o problema consiste em encontrar uma permutação Φ de {1, ... , N} para minimizar

$$f(\Phi) = \sum \sum a_{ij}\, b_{\phi(i)\,\phi(j)} \qquad (2.4)$$

Aplicando nossa notação anterior neste problema, X consiste nas $N!$ permutações de {1, ... , N}; um movimento m em $M(\Phi)$ pode consistir na mudança de dois itens p e q. A permutação π obtida por Φ que pode mover m é dada por

$$\left.\begin{aligned} &\pi = \Phi \oplus m \\ &\pi(i) = \Phi(i) \\ &\pi(p) = \Phi(q) \\ &\pi(q) = \Phi(p) \end{aligned}\right\} \quad \text{para todo } i \neq p, q.$$

A diferença $f(\Phi \oplus m) - f(\Phi)$ pode ser computada facilmente, permitindo exploração de toda a vizinhança $N(\Phi)$ em tempo $O(N^2)$ [7]

Uma possibilidade para criar uma memória baseada em eventos recentes é definir atributos de um movimento pelo par não ordenado de itens (***p*** , ***q***). O atributo par que precisa ser proibido para prevenir o movimento reverso da mesma forma é (***p*** , ***q***) (não permitindo estes dois itens a serem trocadas novamente, constatando que uma mudança imediata deste tipo poderia revisitar a solução já abandonada). Fazendo deste atributo par a base de uma restrição Tabu permite que cada p e q sejam trocadas de cada vez com outros intens. Infelizmente, entretanto, este tipo de restrição não satisfaz a propriedade de proibir um retorno à soluções já visitadas, desde que a seqüência de movimentos (**p** , **s**) (**q** , **s**) (**p** , **r**) (**p** , **q**) (**r** , **s**) não muda nada.

Outra maneira de identificar atributos de um movimento de mudança é introduzir informação adicional, referindo-se não somente a itens alterados, mas a posições ocupadas por estes itens na hora da mudança. Fazer isto sem incluir peso na memória por introduzir vetores de quatro dimensões, nós imaginamos um movimento que será composto de dois meio movimentos (***p***, Φ(***q***)), o primeiro colocando ***p*** na localização ***f***(***q***) e o segundo colocando ***q*** na localização **Φ**(***p***). O modo reverso é prevenido por não permitir que os dois pares de atributos (***p***, Φ(***p***)) e (***q***, Φ(***q***)) ocorram simultaneamente e, são estes dois pares que são armazenados na lista tabu da implementação do QAP.

Mais precisamente, definimos um movimento ***m*** (troca dos itens ***p*** e ***q***) a serem tabu se o movimento manda ambos, ***p*** e ***q***, a localidades que ocuparam dentro das ultimas ***t*** interações. Esta condição tabu admite alguns movimentos prevenidos pela condição ilustrada previamente e também proíbe alguns movimentos permitidos pela condição prévia, por isso não é também uniformemente mais restritiva ou menos restritiva.

A restrição atual tem o apelo de ser baseada não somente em atributos de soluções (correspondendo a itens localizados em posições específicas). Em adição, permitimos o tamanho ***t*** da lista tabu primária, consistindo de pares (***item***, ***localização***), a serem alterados durante o processo de solução.

6. Comparação com Outras Meta Heurísticas

Outra forma de definir o poder e a versatilidade da técnica é realizar uma comparação com outras meta heurísticas igualmente bem estabelecidas e aceitas.

Como técnicas diferentes se comportam ao fornecer soluções para os mesmos problemas? A Busca Tabu fornece respostas com mais ou menos precisão que outras técnicas? Em mais ou menos tempo?

São estas perguntas e comparações que tentaremos elucidar logo a seguir.

6.1 Recozimento Simulado X Busca Tabu

A seção 2 estabelece a Busca Tabu à partir de uma construção por comparação com o Recozimento Simulado. Para mais detalhes, consulte a seção 2.

6.2 Uma comparação de Algoritmos Miméticos, Busca Tabu e Colônia de Formigas para o QAP

[16] realizou um extenso estudo comparativo entre as técnicas descritas acima.

No artigo foi apresentado um algoritmo mimético para resolver várias instâncias do QAP. Foram descritos os ingredientes, operadores evolucionários e busca de vizinhança local do algoritmo memético. Em particular, um novo e altamente eficiente operador de recombinação foi proposto.

O desempenho do algoritmo mimético foi investigado em um conjunto de instâncias do QAP e comparado em desempenho em três técnicas de heurísticas muito boas em resolver QAP: Busca Tabu reativa, Busca Tabu Robusta e o *Fast ant Colony System*. O Algoritmo Mimético foi capaz de superar estas heurísticas em todas as instâncias de QAP de interesse prático.

Sendo assim, nem sempre todas as implementações de Busca Tabu encontram as melhores soluções no melhor tempo possível. Entretanto, os tempos de resposta são geralmente muito bons.

7. Conclusões

A Busca Tabu é uma técnica extremamente versátil e poderosa. Muito provavelmente, grande parte da sua versatilidade advém do fato da técnica se parecer mais com uma aproximação de engenharia para problemas grandes e complexos do que com um algoritmo matemático simples e elegante.

Por não oferecer uma implementação rígida como única resposta ou uma receita pronta para seu uso, é possível aplicar a BT à quase qualquer problema de otimização com resultados ótimos, ou próximos do ótimo por um custo aceitável. Mesmo que não seja sempre a técnica mais indicada para todos os tipos de problemas.

É uma técnica relativamente recente e, apesar de robusta, ainda se encontra em processo de amadurecimento: apesar das várias aplicações tem ainda potencial para muitas outras. E a cada nova aplicação se descobre um pouco mais sobre seu poder.

REFERÊNCIAS

[01] F. Glover; Future paths for integer programming and links to artificial intelligence.

[02] E. Taillard; Robust taboo search for the quadratic assignment problem.

[03] Faigle U., Kern W.; Some Convergence Results for Probabilistic Tabu Search; ORSA Journal on Computing 4, (1992), pg. 32-37.

[04] Fox B.L.; Integrating and accelerating Tabu Search, simulated annealing and genetic algorithms; Annals of Operations Research 41, (1993) pg. 47-67.

[05] Hertz A., de Werra D.; Using Tabu Search Techniques for Graph Coloring; Computing 39, (1987), pg. 345-351.

[06] Glover F.; Tabu Search, Part I - ORSA Journal on Computing 1, (1989).

[07] Glover F.; Tabu Search, Part II - ORSA Journal on Computing 2, (1990).

[08] Hertz A.; Tabu Search for Large Scale Timetabling Problems; European Journal of Operational Research 54/1, (1991), pg. 39-47.

[09] Hertz A.; Finding a Feasible Course Schedule Using Tabu Search; Discrete Applied Mathematics 35, (1992), pg. 255-270.

[10] Hertz A., de Werra D.; Using Tabu Search Techniques for Graph Coloring; (1987), pg. 345-351.

[11] Hertz A., de Werra D.; The Tabu Search Metaheuristic: how we used it; Annals of Mathematics and Artificial Intelligence 1, (1990), pg. 111-121.

[12] Friden C., Hertz A., de Werra D.; STABULUS: a technique for finding stable sets in large graphs with tabu search; Computing 42, (1989) pg. 35-44.

[13] Friden C., Hertz A., de Werra D.; TABARIS: an exact algorithm based on tabu search for finding a maximum independent set in a graph; Computers and Operations Research 17, (1990), pg. 437-445.

[14| Costa D.; A Tabu Search Algorithm for Computing an Operational Time Table; European Journal of Operational Research 76, (1994), pg. 98-110.

[15] Relatórios de implementações recentes de BT: http://www.upt.pt/tabusearch.

[16] Mertz Peter, Freisleben Bernd; A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem.

[17] Lin S., Kernighan B.W.; An Effective Heuristic Algorithm for the Traveling-Salesman Problem.

SUMÁRIO