

Busca Dispersa e Reconexão de Caminhos

Igor Ribeiro Sucupira
(Aluno de Mestrado)

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo
São Paulo - 2005

Sumário

- Fundamentos da Busca Dispersa e da Reconexão de Caminhos.
- Estrutura Geral Unificada
 - Gerador de Diversificação
 - Conjunto de Referência
 - Geração de Subconjuntos
 - Método de Melhora
- Exemplo de Aplicação:
 - Vizinhança para a Conexão
 - Avaliação das Soluções
 - Otimização Local
 - População Inicial
 - Iteração do Algoritmo
 - Fases do Algoritmo
 - Experimentos
- Leituras Complementares

Fundamentos

- São meta-heurísticas evolutivas.
- Já foram aplicadas com sucesso a muitos problemas de otimização.

Por exemplo:

- Problema quadrático de alocação.
- Treinamento de redes neurais.
- Desenho de grafos.
- *Linear ordering*.
- Otimização contínua sem restrições.

Fundamentos

Forma Original da Busca Dispersa (1977)

1. Construa um conjunto C de soluções iniciais quaisquer, viáveis ou não, representadas como vetores numéricos.
2. Enquanto o critério de parada não for satisfeito:
 - Utilize um processo heurístico para produzir algumas soluções a partir das soluções de C . Escolha algumas das melhores para serem *soluções de referência*.
 - Construa novas soluções que sejam combinações lineares entre as soluções de referência. Devem ser realizadas combinações convexas e não-convexas, envolvendo duas ou mais soluções por vez.
 - Escolha algumas das melhores soluções novas para constituírem o novo conjunto C .

Fundamentos

Principais Motivações

- A combinação entre soluções, bem como a aplicação de processos heurísticos para a melhoria das soluções produzidas com essa estratégia, contribui significativamente para a qualidade de um método heurístico.
- Uma coleção com soluções diversas de boa qualidade freqüentemente contém informações úteis para a construção de soluções ótimas.
- Realizar combinações que envolvam mais de duas soluções possui impacto positivo em um método heurístico.

Fundamentos

Reconexão de Caminhos

- Mesma forma da busca dispersa.
- Soluções são combinadas através da geração de "caminhos" entre elas (e "passando por elas"). Esses caminhos correspondem a movimentos em um espaço de vizinhanças (em geral, não euclidiano).
- Durante a conexão entre soluções, pode-se eliminar (caso exista) a restrição de que as soluções devem ser viáveis. O algoritmo não ficará "desorientado", uma vez que a viabilidade será recuperada em algum momento.

Fundamentos

Combinação entre Soluções

- Podem ser consideradas mais de duas soluções por vez: partindo-se de uma delas, alteramos gradualmente suas características, de modo a incorporar atributos das demais soluções, dando preferência às soluções de menor custo e aos *atributos consistentes*.
- Exemplo: escolha sempre o vizinho mais próximo da solução de destino. Ao atingi-la, continue se distanciando da solução de origem, evitando se distanciar muito da solução de destino.

Fundamentos

Forma Construtiva para Combinar Soluções

- As soluções a serem combinadas são chamadas de *soluções-guia*.
- As novas soluções são construídas a partir de uma solução parcial vazia.
- Para incluir um novo componente na solução que está sendo construída, devem ser considerados, principalmente:
 - Os componentes que ocorrem com maior frequência nas soluções-guia. São dados pesos maiores para as ocorrências em soluções de menor custo.
 - Os componentes que ainda não ocorreram nas soluções já construídas.

Estrutura Geral Unificada

Elementos Principais

1. Gerador de Diversificação: a partir de uma ou mais "soluções-semente", produz uma coleção de soluções significativamente distintas.
2. Método de Aperfeiçoamento: recebe uma solução **s** e produz uma solução que seja pelo menos tão boa quanto **s**.
3. Método de Manutenção do Conjunto de Referência: capaz de construir e atualizar o *conjunto de referência*, que é uma coleção composta por **b** soluções que estejam entre as melhores já encontradas.
O valor de **b** é tipicamente pequeno (por exemplo: 20), se comparado aos tamanhos comuns das populações em outras meta-heurísticas evolutivas.
5. Método de Geração de Subconjuntos: recebe um conjunto de soluções e produz diversos subconjuntos deste. A combinação entre soluções será guiada por esses subconjuntos.

Estrutura Geral Unificada

FORMA BÁSICA

1. Construa um conjunto C_0 contendo soluções quaisquer.
2. Utilize o gerador de diversificação para produzir um conjunto C_1 a partir de C_0 .
3. Aplique o método de aperfeiçoamento às soluções de C_1 e construa o conjunto de referência com **b** soluções distintas.
4. Se necessário, volte ao passo 1.
5. Enquanto o critério de parada não for satisfeito:
 - i. Utilize o método de geração de subconjuntos.
 - ii. Para cada conjunto gerado no passo anterior, construa algumas combinações entre suas soluções.
 - iii. Aplique o método de aperfeiçoamento a cada solução do passo anterior, utilizando o método de manutenção do conjunto de referência para incluir novas soluções caso seja desejável.

Estrutura Geral Unificada

Observações

- Um possível critério de parada é terminar o algoritmo quando uma iteração do passo 5 não alterar o conjunto de referência. Porém, o conjunto de referência pode ser parcialmente reconstruído nessas situações.
- O conjunto de referência pode ser atualizado dinamicamente, de forma que as soluções incluídas possam imediatamente ser combinadas.
 - Uma forma simples de fazer isso é retornar ao início do passo 5 sempre que o conjunto de referência for alterado.
 - Porém, muitas soluções não terão a oportunidade de se combinarem com outras, mesmo estando presentes no conjunto de referência em algum momento.

Estrutura Geral Unificada

Observações

- A busca dispersa e a reconexão de caminhos são freqüentemente implementadas em conjunção com a busca tabu.
- De fato, esses métodos compartilham (não por acaso) uma importante característica: baseiam-se em intensificação e diversificação (conceitos que surgiram no contexto da busca tabu) e, portanto, evitam o emprego de aleatoriedade.
- A diversificação pode ser bastante explorada nessas meta-heurísticas evolutivas. Por exemplo: o gerador de diversificação pode ser utilizado durante os passos *5.ii* ou *5.iii*.

Gerador de Diversificação

Exemplo 1 (Vetores Binários)

- Sejam: n o tamanho fixo dos vetores; \mathbf{x} a solução inicial.
- Para cada k tal que $1 < k < n$, produza vetores que diferem de \mathbf{x} em posições com distância k entre elas e que valem 0 (ou o valor original em \mathbf{x}) nas demais posições.
- Por exemplo: se $\mathbf{x} = (0, 0, \dots, 0)$, serão construídas:
 - $(1, 1, 1, \dots)$.
 - $(1, 0, 1, 0, \dots)$, $(0, 1, 0, 1, \dots)$.
 - $(1, 0, 0, 1, 0, 0, \dots)$, $(0, 1, 0, 0, 1, 0, 0, \dots)$,
 $(0, 0, 1, 0, 0, 1, 0, 0, \dots)$.
 - ...
- Gere o complemento de cada uma das soluções resultantes do passo anterior.
- Se necessário, gere mais soluções, que difiram de \mathbf{x} nas posições $c+1$, $c+2$, $c+k+1$, $c+k+2$, $c+2k+1$, ..., com $1 < k < n$ e $0 < c < k$.

Gerador de Diversificação

Exemplo 2 (Permutações)

- Criar distintas permutações de $(1, 2, \dots, n)$.
- Sejam:
 - $S(k, j) = (j, j+k, j+2k, \dots)$, para todo k e todo j com $1 < j \leq k < n$.
 - $P(k) = (S(k, k), S(k, k-1), \dots, S(k, 1))$, para todo $k < n$.
- Por exemplo, se $n = 7$, teremos:
 - $P(3) = (3, 6, 2, 5, 1, 4, 7)$.
- Seja $P^*(k)$ o reverso de $P(k)$, para todo k .
- Gere $P(k)$ e $P^*(k)$ para a maior quantidade possível de valores de k , especialmente os que estiverem próximos à raiz de n .

Conjunto de Referência

Para a inicialização do conjunto de referência (após a produção de soluções com o gerador de diversificação e o método de melhora), há duas alternativas principais:

- Simplesmente selecionar as ***b*** melhores soluções distintas.
- Selecionar as ***m*** melhores soluções (por exemplo, $m = 5 \times b$), definir uma medida de distância e, então:
 - Incluir a melhor solução no conjunto de referência.
 - Enquanto o conjunto não tiver tamanho ***b***, incluir nele a solução mais diferente das soluções já incluídas.

Conjunto de Referência

- A princípio, o conjunto de referência deve ser atualizado sempre que surgir uma solução com custo inferior ao da pior solução do conjunto.
- Porém, o conjunto de referência não admite duplicações.
- Além disso, pode-se levar em conta a distância da nova solução às demais, para que a diversidade não seja prejudicada.
- Quando for necessário reconstruir o conjunto de referência, podemos remover suas **$b/2$** piores soluções e acrescentar novas, com o mesmo algoritmo utilizado para a geração do conjunto de referência inicial.

Geração de Subconjuntos

Obviamente, não é desejável gerar todos os subconjuntos do conjunto de referência. Os autores da meta-heurística sugerem:

- Gerar todos os subconjuntos com duas soluções.
- Para cada conjunto $\{w, x\}$ produzido no passo anterior, gerar um conjunto $\{w, x, y\}$ em que y é a melhor solução diferente de w e de x (ou uma das melhores).
- De forma análoga, gerar conjuntos com quatro soluções.
- Para cada $i > 4$, gerar um conjunto com as i melhores soluções do conjunto de referência.

Método de Melhora

- Naturalmente, pode-se utilizar qualquer algoritmo de busca local.
- Uma alternativa seria implementar uma meta-heurística de busca por entornos, mas o processo se tornaria muito longo e dominaria a execução.
- Os autores propõem um algoritmo com características intermediárias em relação às duas opções acima.
- Algoritmo de filtragem e espalhamento (*filter and fan*): combinação entre as estratégias de espalhamento seqüencial (*sequential fan*) e de filtragem, utilizadas na busca tabu.

Método de Melhora

Algoritmo

1. Geração da primeira lista de soluções candidatas:
 - i. Se algum dos vizinhos da solução corrente é melhor que ela, selecione o de menor custo, atualize a solução corrente e execute este passo novamente.
 - ii. Caso contrário, construa o conjunto $V(1)$, com os v_0 melhores vizinhos da solução corrente, faça $L = 1$ e vá para o passo 2.
2. Enquanto $L \leq MAXL$, faça:
 - i. Se algum dos vizinhos das soluções de $V(L)$ é melhor que a solução corrente, selecione o melhor deles e volte ao passo 1.
 - ii. Caso contrário, construa $V(L+1)$, com os v_0 melhores vizinhos das soluções de $V(L)$, e faça $L = L + 1$.

Exemplo de Aplicação

- Problema Generalizado de Atribuição (*GAP* - *Generalized Assignment Problem*).
- É um problema clássico NP-difícil.
- Dados:
 - Um conjunto de tarefas $I = \{1, \dots, n\}$.
 - Um conjunto de agentes $J = \{1, \dots, m\}$.
 - $p(i, j)$: lucro obtido ao associarmos a tarefa i ao agente j .
 - $r(i, j)$: consumo de recursos de i quando associada a j .
 - $c(j)$: capacidade do agente j .
- Associar cada tarefa a exatamente um agente, respeitando as capacidades.
- Maximizar o lucro.

Exemplo de Aplicação (GAP)

- Os melhores algoritmos exatos atualmente resolvem, em geral, instâncias com cerca de 100 tarefas.
- Algumas meta-heurísticas já aplicadas ao problema:
 - Algoritmos genéticos.
 - Recozimento simulado.
 - Busca tabu.
- Em 2002, Alfandari et al. desenvolveram um algoritmo de reconexão de caminhos.

Aplicação da Reconexão de Caminhos

- Combinações entre pares de soluções. Uma conexão entre duas soluções x' e x'' nunca remove associações (*tarefa, agente*) que sejam comuns a x' e x'' (atributos consistentes).
- A distância entre duas soluções x' e x'' é o número de tarefas associadas a diferentes agentes em x' e x'' .

Vizinhança para a Conexão

- Para conectar uma solução x' a uma solução x'' , é realizado um processo iterativo que se inicia com $x = x'$ e, em cada passo, substitui x por um de seus vizinhos que esteja mais próximo de x'' .
- Uma solução é vizinha de outra se apenas uma tarefa está associada a um agente diferente ou se as duas soluções se tornam iguais trocando-se os agentes de duas determinadas tarefas em uma das soluções.
- De outros experimentos com meta-heurísticas, sabe-se que as trocas têm um papel importante na resolução do problema.

Avaliação das Soluções

- Função objetivo modificada:

$$v(\mathbf{x}, \mathbf{a}) = \mathbf{f}(\mathbf{x}) - \mathbf{a} \mathbf{g}(\mathbf{x})$$

- \mathbf{f} é a função objetivo (lucro).
- \mathbf{a} é o fator de penalização de soluções inviáveis.
- $\mathbf{g}(\mathbf{x})$ é a média das violações $h(\mathbf{x}, j)$ sobre todos os agentes j .

$$h(\mathbf{x}, j) = \max\{0, z(\mathbf{x}, j)/c(j) - 1\}$$

- $z(\mathbf{x}, j)$ são os recursos consumidos no agente j , na solução \mathbf{x} .
- A variável \mathbf{a} é dinâmica. Seu valor é ajustado para controlar a quantidade de soluções viáveis no conjunto de referência.

Otimização Local

- Em cada caminho construído, seleciona-se a melhor solução e realiza-se otimização local a partir dela.
- O ótimo local é encontrado a partir de uma busca local gulosa.
- A definição de vizinhança é a mesma utilizada na conexão.
- A otimização local é importante para evitar duplicações, uma vez que o ótimo local em relação a uma solução \mathbf{x}_k do caminho entre \mathbf{x}' e \mathbf{x}'' pode ser uma destas duas.

População Inicial

1. Inicialmente, é produzida uma solução de boa qualidade (espera-se):
 - i. O problema é formulado como programação inteira e, então, relaxado para programação linear e resolvido como tal.
 - ii. As variáveis da solução são transformadas em inteiras.
 - iii. Em seguida, realiza-se otimização local.
2. A seguir, constróem-se m soluções $\{x_1, \dots, x_m\}$ com diversidade máxima (nenhuma associação (*tarefa, agente*) em comum).

População Inicial (Continuação)

3. Suponha que $b = n/m$ é inteiro (o método pode ser facilmente adaptado se isso não for verdade).
4. Seja $I_j = \{(j-1)b + 1, \dots, (j-1)b + b\}$, para todo agente j .
5. Definindo $x_k(i)$:
 - i. Seja I_j o conjunto ao qual i pertence.
 - ii. Se $j < k$, $x_k(i) = m - k + j + 1$.
 - iii. Caso contrário, $x_k(i) = j - k + 1$.
6. Por fim, realiza-se otimização local a partir de cada uma das m soluções.

Iteração do Algoritmo

- Primeira iteração: para cada par (x', x'') de soluções distintas no conjunto de referência inicial, se $v(x', a) \leq v(x'', a)$, conecte x' a x'' e acrescente ao conjunto o ótimo local relativo à melhor solução do caminho, desde que esse seja uma solução nova.
- Demais iterações: para cada par (x', x'') em que ao menos uma solução é nova (do passo anterior), conecte x' a x'' , se $v(x', a) \leq v(x'', a)$.
- Alternativas experimentadas:
 - Sem a restrição $v(x', a) \leq v(x'', a)$.
 - Com a restrição $v(x'', a) \leq v(x', a)$.
 - Cada uma das três versões anteriores, com a restrição $d(x', x'') \geq DMIN$. Pequeno ganho se $DMIN = 3$. Resultados diversos com outros valores (trabalhos futuros: $DMIN$ dinâmico).

Fases do Algoritmo

- Fase 1: as iterações são executadas como descrito anteriormente, até que o tamanho do conjunto de referência atinja $TMAX$.
- Fase 2: quando o conjunto de referência passa a conter $TMAX$ soluções, seu tamanho se torna fixo. A inclusão de uma nova solução S_N substituirá a pior solução do conjunto e só ocorrerá se S_N for melhor que esta última.
- Ajuste do parâmetro \mathbf{a} :
 - Durante a fase 1, \mathbf{a} é ajustado para que o conjunto de referência sempre contenha ao menos uma solução viável e uma inviável.
 - Durante a fase 2, \mathbf{a} é ajustado para que haja ao menos uma solução inviável e para que ao menos 25% das soluções sejam viáveis.
- Critério de parada: MAX_ITER iterações da fase 2 ou uma iteração sem alteração do conjunto de referência.

Experimentos

- Conjunto de 60 instâncias (com soluções ótimas conhecidas) já utilizadas por diversos outros autores.
- Comparação com as melhores implementações de meta-heurísticas:
 - RS/BT: implementação híbrida (recozimento simulado e busca tabu).
 - BT1 e BT2: duas implementações da busca tabu.
 - GA: um algoritmo genético.
- Distância média do ótimo:
 - RS/BT: 0,210.
 - BT1: 0,070.
 - BT2: 0,004.
 - GA: 0,009.
 - Reconexão de caminhos: 0,004 (gastando um tempo de CPU maior que o gasto por BT2).

Leituras Complementares

- GLOVER, F. Heuristics for Integer Programming Using Surrogate Constraints. 1977.
- GLOVER, F. A Template for Scatter Search and Path Relinking. 1998.
- GLOVER, F., LAGUNA, M., MARTÍ, R. Scatter Search and Path Relinking: Foundations and Advanced Designs. 2004.

Leituras Complementares

- CUNG, V-D. et al. Scatter Search for the Quadratic Assignment Problem. 1996.
- FLEURENT, C. et al. A Scatter Search Approach for Unconstrained Continuous Optimization. 1996.
- KELLY, J., RANGASWAMY, B., XU, J. A Scatter Search-Based Learning Algorithm for Neural Network Training. 1996.
- LAGUNA, M., MARTÍ, R. GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization. 1997.
- LAGUNA, M., MARTÍ, R., CAMPOS, V. Tabu Search with Path Relinking for the Linear Ordering Problem. 1997.
- ALFANDARI, L., PLATEAU, A., TOLLA, P. A Path-Relinking Algorithm for the Generalized Assignment Problem. 2002.

Leituras Complementares

- OSMAN, I. Heuristics for the Generalized Assignment Problem: Simulated Annealing and Tabu Search Approaches. 1991.
- WILSON, J. A Genetic Algorithm for the Generalized Assignment Problem. 1997.
- CHU, P., BEASLEY, J. A Genetic Algorithm for the Generalized Assignment Problem. 1997.
- DÍAZ, J., FERNÁNDEZ, E. A Tabu Search Heuristic for the Generalized Assignment Problem. 2001.

Esta Apresentação

A partir da minha página:
<http://www.ime.usp.br/~igorrs>