Universidade Federal do Paraná – UFPR
Curso de Ciência da Computação
CI801 – Tópicos em Inteligência Artificial

Busca Local Iterada

Kelly Rodrigues Abreu
Federico Luis Losco

28 de Maio de 2007.

Sumário

Introdução
Busca Local Iterada
Aplicações
Relações com outras metaheuristicas
Conclusões

Introdução

- Metaheurística deve ser simples, eficiente e mais genérica possível.
- Problema específico deve ser incorporado à metaheurística. Portanto deve ser composta por:
    1. função geral e;
    2. parte que varia de acordo com o problema.

- A heurística incorporada estará em um módulo (“caixa-preta”).

- A Busca Local Iterada:

    Constroi iterativamente uma seqüência de soluções geradas por uma heurística incorporada;

    Guia para melhores soluções do que se usasse testes aleatórios repetidos.

- Heurística incorporada: Busca local

Algoritmo - Estrutura Geral

- ***C*** função de custo a ser minimizada.
- Soluções candidatas são rotuladas ***s,*** e um conjunto por ***S.***
- Busca local: determinística e sem memória
- Usa estrutura de vizinhança. Assim, é possível mover-se de uma solução ***s*** para uma melhor ainda por um caminho inteligente. Algo que não seria possível se ***S*** fosse apenas um conjunto.
- Então, como reduzir o custo sem abrir a “caixa-preta”?

Algoritmo

Random Restart

- A maneira mais simples de melhorar o custo é repetir a BuscaLocal de outro ponto inicial.
- Cada s* gerada será independente.
- Perde a utilidade medida que o espaço de busca aumenta.
- Em instâncias muito genéricas levam o custo a: ter uma média que é uma taxa fixa que excede o custo ótimo; ter uma *distribuição* que torna-se arbitrariamente acima da média quando o tamanho da instância tende a infinito.
- É necessário então, uma amostra parcial.

Algoritmo

Buscando em S*

- Evita grandes espaços de busca
- É feito de forma recursiva.
- Gera uma hierarquia de busca locais aninhadas

Como formular a busca local no nível mais baixo da hierarquia?

- Busca local requer uma estrutura de vizinhança;
- O maior problema é como definir os vizinhos de ***S**** para que sejam numerados e acessados de forma eficiente.
- Aplicar busca estocástica em S*

Algoritmo

Busca Local Iterada

Para explorar ***S**** sem a noção de vizinhança, aplica-se uma perturbação que leva ***s**** a um estado intermediário ***s'***, aplica-se a busca local em ***s'***, gerando ***s'**, se este passar pelo teste de aceitação ele passar a ser o próximo elemento do caminho em ***S****.

- Memória: a maioria ainda não utiliza, mas se precisar de *s** previamente:
  Então podem ser usados: diversificação, intensificação, tabu, perturbações adaptativas, critério de aceitação etc.

- O desempenho da ILS depende muito da perturbação e do critério de aceitação escolhido.

Algoritmo

procedure Iterated Local Search

s0 = GenerateInicialSolution

s* = LocalSearch(s0)

repeat

s' = Perturbation(s*, history)

s*'= LocalSearch(s*)

s* = AcceptanceCriterion(s*, s*', history)

until termination condition met

end

Algoritmo – Melhorando a performance

Componentes básicos:

- Gera solução inicial
- Busca local
- Perturbação
- Critério de aceitação

Algoritmo – Melhorando a performance

Solução Inicial

Começar com uma boa solução pode ser importante se soluções de alta qualidade estão sendo buscadas o mais rápido possível.

- Há dois tipo de inicialização: Aleatória ou Gulosa.

  Vantagens da gulosa:

  - Combinada com a busca local resulta em soluções s_0^* de melhor qualidade
  - Uma busca local a partir de uma solução gulosa, requer menos tempo de CPU.

Algoritmo – Melhorando a performance

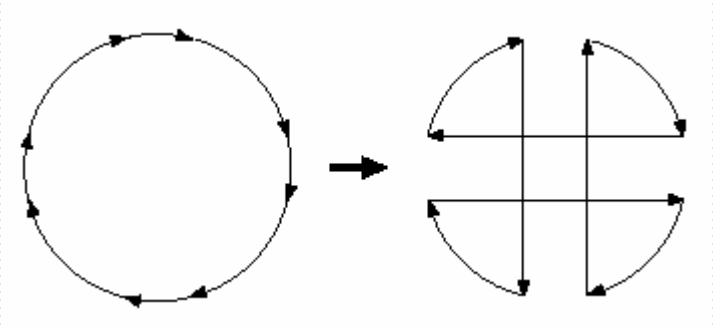
Perturbação

- A ILS aplica perturbações para sair dos ótimos locais.
- Grau de perturbação: número de vezes que os componentes da solução forem modificados.
- Pode-se obter melhores soluções se as características do problema são considerados.
- Perturbação Alta – ILS comporta-se como random restart e há pouca possibilidade de se encontrar boas soluções.
- Perturbação Fraca – a busca local cai frequentemente em um ótimo local já visitado e a diversificação torna-se muito limitada.
- Em problemas simples como TSP, pode se obter resultados satisfatórios usando perturbação fixa (movimento double-bridge)

Algoritmo – Melhorando a performance

Perturbação

- Em problemas simples como TSP, pode se obter resultados satisfatórios usando perturbação fixa (movimento double-bridge)



- Já em problemas complexos pode haver uma grande perda de desempenho por causa da dificuldade de configurar a perturbação.
- Perturbações fracas geralmente levam a execuções mais rápidas, porém pode-se cair num mesmo ótimo local.

Algoritmo – Melhorando a performance

Perturbações adaptativas

- Não existe um melhor valor para o grau de perturbação

- Assim, se torna melhor que seja adaptado durante a execução

  Pode ser feito de duas formas:

  1. Quardando o histórico da busca
  2. Mudando deterministicamente o grau durante a busca, utilizando oscilações estratégicas.

Algoritmo – Melhorando a performance

Velocidade

- Empiricamente ILS é mais rápido em busca locais que random restart

instance	$\#LS_{RR}$	$\#LS_{1\text{-DB}}$	$\#LS_{k\text{-DB}}$
kroA100	17507	56186	34451
d198	7715	36849	16454
lin318	4271	25540	9430
pcb442	4394	40509	12880
rat783	1340	21937	4631

Algoritmo – Melhorando a performance

Critério de Aceitação

- A perturbação junto com a busca local definem as posíveis transações entre a solução atual s* e uma solução vizinha s*';
- O critério de aceitação determina quando s*' é aceito ou não;
- Pode ser usado para controlar o balanço entre intensificação e diversificação da busca.

Algoritmo – Melhorando a performance

Critérios de aceitação

- Better: forte intensificação e só aceita as melhores soluções.

$$\texttt{Better}(s^*, s^{*\prime}, history) = \begin{cases} s^{*\prime} & \text{if } C(s^{*\prime}) < C(s^*) \\ s^* & \text{otherwise} \end{cases}$$

Algoritmo – Melhorando a performance

Critérios de aceitação

- RW: sem considerar o custo, aplica a perturbação ao ótimo ,local visitado mais recentemente; favorece a diversificação sobre a intensificação

$$\mathrm{RW}(s^*, s^{*\prime}, history) = s^{*\prime}$$

Algoritmo – Melhorando a performance

Critérios de aceitação

- Restart: quando a intensificação parece ineficiente o algoritmo deve ser reinicializado. Por exemplo, quando não se obtem melhoras em um determinado número de iterações.

$$\texttt{Restart}(s^*, s^{*\prime}, history) = \begin{cases} s^{*\prime} & \text{if } C(s^{*\prime}) < C(s^*) \\ s & \text{if } C(s^{*\prime}) \geq C(s^*) \text{ and } i - i_{last} > i_r \\ s^* & \text{otherwise.} \end{cases}$$

Algoritmo – Melhorando a performance

Critérios de aceitação

- Exemplo TSP

  Foi comparado o uso na ILS de Better e RW contra Random Restart

  - ILS conseguiu soluções melhores utilizando a mesma busca local
  - Para TSP as soluções boas estam clusterizadas (agrupadas)
  - Uma boa estratégia é incorporar a intensificação

Algoritmo – Melhorando a performance

Busca Local

- Deve ser otimizada o máximo possível
- Nem sempre a melhor leva a uma melhora na ILS.
- Se for dado um tempo de computação fixo: é melhor um algoritmo que seja mais rápido porém menos eficiente
- A escolha deve considerar se existe um tempo para executar a busca.
- Não faz sentido usar uma busca local que desfaz a perturbação, por isso a importância da otimização global
- Para o TSP, por exemplo, a busca que se comporta melhor é A Lin-Kernighan

Algoritmo – Melhorando a performance

Otimização Global

- Ao otimizar um dos componentes os outros se mantêm fixos
- A otimização de um componente depende da escolha feita para os outros componentes
- Neste artigo a geração da solução inicial foi ignorada
- A perturbação depende da busca local
- O critério de aceitação depende da busca local e da perturbação
- Otimização iterativa: otimização sucessiva de cada componente até que não se obtenha mais melhoras

Algoritmo – Melhorando a performance

Características do espaço de busca

- Se as melhores estão agrupadas em S*, no TSP, a intensificação será útil, aumentando assim, a possibilidade de se encontrar o ótimo local.
- Se o agrupamento é incompleto, como nos exemplos QAP, grafo biparticionado e MAX-SAT, será útil uma fase de intensificação.
- O balanço entre intensificação e diversificação é importante e desafiante.

Aplicações

ILS para TSP

LSMC, proposto por Martin, Otto e Felten

- Perturbação: movimento de Double-bridge
- Critério de Aceitação: SA

ILS usando Lin-Kernighan como busca local, proposto por Johnson.

As principais diferenças com a implementação LSMC são:

- Movimentos Double-bridge são aleatórios ao invés de parcial;
- Os custos estão melhorando (só os melhores tours são aceitos)

Aplicações

ILS para TSP

LK por Aplegate, Bixby, Chvatal e Cook

- Tour inicial;
- Implementação das escolhas da heurística LK;
- Tipos de perturbação.

Aplicações

ILS para problemas de planejamento:

1. Single Machine Total Weighted Tardiness Problem (SMTWTP)

- Encontrar o melhor movimento que é composto de um conjunto de movimentos de troca independentes.
- Cada movimento troca o trabalho nas posições i e j, j diferente de i.
- Dois movimentos de troca são independentes se não fazem sobreposição.
- A perturbação consiste de uma série de movimentos de troca aleatória.
- Critério de aceitação é introduzido o *backtrack step*

Heurísticas usadas:

- Dynasearch;
- Busca local baseada na primeira melhora da descida;
- Busca local baseada na melhor melhora da descida.

Aplicações

2. Single and parallel machine scheduling

- Planejamento de uma máquina o de máquinas paralelas.
- Método de busca local baseado em duas vizinhanças.
- Vizinhança primária corresponde a fase de busca local.
- Vizinhança secundária corresponde a fase de perturbação.

3. Programação de fluxo de compra

- ILS é um algoritmo baseado em uma busca local de primeira melhora direta usando a inserção de vizinhança, onde um trabalho na posição i é movido e inserido na posição j i.
- Perturbações com apenas uma pequena swap e movimentos de troca são suficientes para obter ótimos resultados.

Aplicações

4. Programação de trabalho de compra

- ILS obteve uma performance melhor que random restart para qualquer heurística de busca local escolhida.
- GLS - baseado em árvores de vizinhança, cada nó corresponde a uma solução.

Aplicações

ILS para outros problemas

1. Graph bipartitioning

- Busca local Lin-Kernighan
- Critério de aceitação *Better.*
- O SA foi o melhor método para este problema.

2. MAX-SAT

- Battiti e Protasi apresentaram uma aplicação de busca reativa no problema
- O algoritmo tem duas fases: Busca local e diversificação (perturbação)

3. Prize-collecting Steiner tree problem

Estratégias de busca local utilizadas: melhora iterativa, multi-start com perturbações, path-relinking, busca em vizinhança variável, e um algoritmo baseado na integração de todos esses.

Sumário

- A escolha do algoritmo de busca pode não ser muito bom se ele não for otimizado

- Os componentes da ILS também devem ser otimizados e configurados adeqüadamente para que se obtenha um bom desempenho

ILS é uma metaheurística versátil que pode ser adaptada a diferentes problemas.

Comparação com metaheurísticas

Metaheurísticas baseadas em vizinhança

- As metaheuristicas diferem em sua estratégia de movimento.

- Qualquer metaheurística baseada em vizinhança pode ser colocada como BuscaLocal da ILS é que freqüentemente obtem-se melhores soluções do que com algoritmos de descida iterativa.

Comparação com metaheurísticas

Metaheurísticas baseadas em multi-inicialização

Construtivas

- Ex.: Colônia de Formigas e GRASP as quais fazem contrução da solução
- ILS não constrói
- Uma fase de perturbação na ILS pode ser substituída por uma fase de construção na metaheurística construtiva.

Baseadas em perturbação

- Baseados em população: EA e scatter search.
- Única solução: ILS – gera soluções por perturbações
- Algumas extensões de ILS baseadas em população têm sido propostas, pai é perturbado para dar um filho.

Comparação com metaheurísticas

A maior diferença entre iLS e VNS é:

- ILS tem o objetivo de construir um caminho no conjunto das soluções ótimas locais,
- enquanto VNS é derivado da idéia de trocar sistematicamente as vizinhanças durante a busca.

- Os limites entre as diferentes metaheurísticas não estam bem definidos.

Conclusões

ILS é

- simples
- fácil de implementar
- robusto
- altamente eficiente.

Idéia essencial:
Focar a busca nopequeno subespaço definido pelas soluções que são localmente ótimas.

- O sucesso da ILS depende principalmente da escolha da busca local, das perturbações e do critério de aceitação.

Conclusões

Sugestões para aplicação da ILS:

- problemas onde a maior parte das metaheurísticas falham;
- problemas com multi objetivos;
- problemas dinâmicos ou de tempo-real onde os dados variam durante a resolução do problema.

As idéias e resultados apresentados deixam muitas perguntas sem respostas.
Espera-se significantes melhoras com o uso de memória, estratégias explícitas de intensificação e diversificação.

Perguntas