Análise de uma Busca Local Iterada para o Problema do Roteamento de Veículos com Backhauls

Marcelo de Souza, Marcus R. P. Ritt
Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre - RS
{mdesouza, marcus.ritt}@inf.ufrgs.br

Resumo

O problema do roteamento de veículos com backhauls (PRVB) é uma variante do problema do caixeiro viajante (PCV) que considera a determinação de rotas para o suprimento de uma cadeia de clientes. Neste contexto, alguns clientes recebem bens originados no depósito, ao passo que outros enviam bens ao depósito. As rotas devem ser criadas para o suprimento das demandas com o menor custo possível, sem violar a capacidade máxima dos veículos. Este trabalho apresenta a análise do algoritmo de busca local iterada para o PRVB. O objetivo é identificar os componentes principais da heurística e determinar a influência de cada um deles no desempenho da busca. Neste sentido, foram analisados o mecanismo de perturbação, a estratégia da busca local, a geração e a exploração da vizinhança. Testes estatísticos determinaram a superioridade da implementação segundo as especificações da fonte. A estratégia de perturbação total e diferentes formas de exploração da vizinhança conseguem manter a qualidade dos resultados, diminuindo o tempo gasto na execução.

Problema do Roteamento de Veículos com Backhauls

Problema do Roteamento de Veículos (PRV)

- Um depósito central;
- Um conjunto de clientes que requisitam bens;
- Um conjunto de veículos idênticos com uma capacidade fixa;
- Cada rota inicia e termina no depósito;
- Cada cliente é visitado uma única vez;
- A quantia de bens de cada rota não pode exceder a capacidade do veículo;
- **Objetivo:** Definir uma rota para cada veículo minimizando a distância total percorrida.

Problema do Roteamento de Veículos com Backhauls (PRVB)

- Consumidores (linehauls) requisitam bens do depósito;
- Supridores (**backhauls**) enviam bens de volta ao depósito;
- Os consumidores devem ser visitados antes dos supridores;
- Cada rota deve possuir ao menos um cliente linehaul.

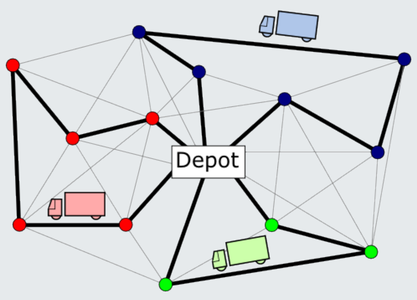

Fig 1. Funcionamento do PRV

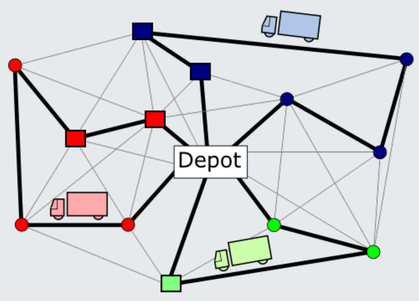

Fig 2. Funcionamento do PRVB

Heurística Analisada

Consiste em uma Busca Local Iterada proposta por [1].

- Função *Oscillated Local Search* (OLS) com estratégia de Melhor Melhora;
- Permite soluções inviáveis mediante uma penalidade na função objetivo.

Função objetivo

$$cost(S) = distance(S) + \sum_{r=1}^{m} [lh\_excess\_load(r) + bh\_excess\_load(r)]$$

Vizinhança

- Intra-route e inter-route relocation (shift);
- Intra-route e inter-route exchange (swap);
- Intra-route e intra-route 2-opt;
- Intra-route e inter-route crossover;

Perturbação
Realoca os clientes iterativamente, tanto os clientes como as posições são selecionadas aleatoriamente. O tamanho da perturbação (% dos clientes a serem realocados) é definido por um parâmetro.

Algorithm OLS
Input S_0: solução inicial
Output A melhor solução viável encontrada.

```
1  while ¬stop_criterion do
2      while neighbors_with_lower_cost(S_act, α) ≠ ∅ do
3          S_act ← neighbor_with_lowest_cost(S_act, α)
4      if is_feasible(S_act) then
5          if cost(S_act) < cost_best_feas then
6              cost_best_feas ← cost(S_act)
7              S_best_feas ← S_act
8              α ← α_0
9          else
10             stop_criterion ← true
11     else
12         α ← β · α
13 return S_best_feas
```

Componentes Estudados

Perturbação

- Nula (0%); Normal (40%); Semi-total (95%).

Estratégia de Busca Local

- Melhor Melhora (MM) e Primeira Melhora (PM).

Vizinhanças consideradas

- Somente relocation; Relocation e exchange; Completa (relocation, exchange, 2-opt, crossover).

Exploração da vizinhança

- Completa; Sequencial; Semi-sequencial.

Experimentos e Testes Estatísticos

Os experimentos foram executados sobre o benchmark GJB [2]. Foram considerados os grupos de instâncias A a L (total de 52 instâncias). Os experimentos foram conduzidos em um computador com processador Intel Core i7-3770 de 3.40GHz. Os resultados para um total de 10 replicações para cada experimento podem ser observados nas tabelas abaixo. Para um máximo de 1000 iterações, todos os ótimos conhecidos foram encontrados. No entanto, o máximo de iterações foi fixado em 400 por questões de tempo. O parâmetro α foi fixado em 0.01, a perturbação em 40% e o valor de β em 5.

Implementação da especificação de [1]

Instância	Tempo	Iterações	Desvio
A	0.301	55.5	0.0004
B	1.201	55.9	0.0001
C	9.406	131.1	0.0020
D	1.989	35.1	0.0000
E	3.602	58.6	0.0002
F	37.570	221.9	0.0033
G	24.871	175.1	0.0015
H	153.094	354.3	0.0137
I	671.593	400.0	0.0238
J	809.252	400.0	0.0067
K	930.640	400.0	0.0152
L	1181.100	400.0	0.0277
Média	318.718	224.0	0.0079

Na execução do algoritmo de acordo com as especificações apresentadas em [1], a qualidade das soluções foi bastante similar aos resultados produzidos no artigo original. O tempo gasto pela implementação foi consideravelmente maior. Este comportamento era esperado, uma vez que o objetivo consistiu em replicar a qualidade das soluções encontradas para futura análise dos componentes selecionados para estudo. Os resultados das alterações nos componentes podem ser observados na tabela a seguir.

Variações implementadas para análise dos componentes

Implementação	Tempo	Iterações	Desvio
Sem perturbação	33.717	379.1	0.1268
Máxima perturbação	261.526	234.5	0.0236
Primeira melhora	276.662	249.9	0.0150
Exploração sequencial	119.503	234.2	0.0327
Exploração semi-sequencial	112.203	230.6	0.0092
Vizinhança shift	11.961	400.0	1.2438
Vizinhança shift + swap	162.651	380.3	0.2179

Para confirmar algumas hipóteses, alguns testes estatísticos foram conduzidos sobre a qualidade das soluções encontradas. Pelo teste de Friedman, constatou-se que não existe diferença significativa entre as distintas formas de exploração da vizinhança. Para os demais casos (com diferença), foram realizados testes de Wilcoxon de postos com sinais, os quais são sumarizados abaixo.

Hipótese alternativa (H_1)	p-value	Resultado
Pert. normal melhor que sem pert.	2.2×10^{-16}	Pert. normal é superior em qualidade.
Pert. normal melhor que pert. semi-total	0.1006	Estratégias são iguais.
Pert. semi-total melhor que pert. normal	0.8994	Estratégias são iguais.
Pert. semi-total melhor que sem pert.	2.2×10^{-16}	Pert. semi-total é superior em qualidade.
MM melhor que PM.	3.21×10^{-5}	MM é superior em qualidade.
Ger. completa melhor que shift.	2.2×10^{-16}	Ger. Completa é superior em qualidade.
Ger. completa melhor que shift+swap.	2.2×10^{-16}	Ger. Completa é superior em qualidade.
Ger. shift+swap melhor que shift.	2.2×10^{-16}	Ger. shift+swap é superior em qualidade.

Para os casos em que não há diferença significativa em relação à qualidade, testes foram conduzidos com relação ao tempo gasto por cada implementação.

Hipótese alternativa (H_1)	p-value	Resultado
Pert. semi-total melhor que pert. normal	2.2×10^{-8}	Pert. semi-total é melhor no tempo.
Expl. sequencial melhor que completa	1.581×10^{-7}	Expl. sequencial é melhor no tempo.
Expl. semi-sequencial melhor que completa	1.623×10^{-7}	Expl. semi-sequencial é melhor no tempo.

Conclusões

A qualidade da solução inicial gerada não produz influência direta sobre a qualidade final da busca. Com relação às instâncias, percebeu-se que quanto maior o número de clientes, maior a complexidade da busca. Além disso, a complexidade aumenta com o crescimento do número de clientes do tipo backhaul, bem como quando a capacidade dos veículos diminui (ou a demanda dos clientes aumenta). As modificações implementadas permitiram identificar a contribuição de cada componente no desempenho da busca. Os testes estatísticos realizados permitiram confirmar algumas hipóteses e serviram de base para as conclusões.

Para a **perturbação**

- Perturbação semi-total mantém a qualidade dos resultados e melhora o tempo;
- A perturbação não se mostra fundamental para a heurística. A aplicação de um algoritmo GRASP sugere bons resultados.

Para a **geração da vizinhança**

- Somente a geração completa garante a qualidade das soluções encontradas;
- Evolução: + shift → + swap → completa.

Para a **exploração da vizinhança**

- Apesar de não apresentar diferença significa na qualidade, diminui o tempo.

Referências

[1] D. P. Cuervo, P. Goos, K. Sörensen, and A. Arráiz. An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 237:454–464, 2014.

[2] M. Goetschalckx and C. Jacobs-Blecha. The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42:39–51, 1989.

XLVI Simpósio Brasileiro de Pesquisa Operacional - 2014, Salvador / BA - Brasil