Paralelismo no algoritmo heurístico para resolução de problemas de localização de concentradores com maximização do lucro e conexão direta

Henrique Heiderscheidt
henrique.heid@gmail.com
Instituto Federal Catarinense - Campus Blumenau
Blumenau, Santa Catarina, BR

Abstract

This work addresses the hub location problem with multiple allocation, focusing on profit-oriented configurations and allowing incomplete hub networks. A parallelized heuristic framework combining Iterated Local Search (ILS) and a Random Variable Neighborhood Descent (RVND) approach is proposed. The RVND method explores distinct neighborhood structures in a controlled yet flexible manner, while the ILS iterates between intensification and diversification phases to refine solutions and escape local optima. Parallelization leverages multiple threads to explore various regions of the solution space concurrently, significantly reducing computation times. Computational experiments were conducted using the AP dataset and instances with up to 200 nodes. Results indicate that the parallel ILS heuristic consistently produces solutions close to the optimal values obtained by the CPLEX solver, while requiring substantially less computational effort. The method also outperforms a non-parallelized heuristic (Smart ILS), confirming that parallelization can effectively address increasing problem complexity in hub location optimization.

Palavras-chave

Localização de Concentradores, Algoritmos Heurísticos, Paralelismo

ACM Reference Format:
Henrique Heiderscheidt. 2024. Paralelismo no algoritmo heurístico para resolução de problemas de localização de concentradores com maximização do lucro e conexão direta. In . -, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introdução

Os concentradores são instalações especiais que geram economias na conexão entre pares de origem e destino em uma rede, consolidando e distribuindo fluxos para reduzir os custos de transporte. O problema clássico de localização de concentradores envolve definir quais nós serão concentradores, alocar os nós de demanda e rotear os fluxos, sendo um problema NP-difícil [2] devido à complexidade dessas decisões combinadas. O objetivo é encontrar a configuração da rede que maximize o lucro gerado.

Este trabalho aborda o problema de localização de concentradores sob a ótica do lucro, utilizando a estratégia de alocação múltipla. Considera-se que a rede de concentradores é incompleta, que conexões diretas entre nós não concentradores são permitidas e que o problema é não capacitado, assumindo capacidade suficiente dos concentradores e arcos para atender ao fluxo de demanda. Essa configuração pode ser observada na Figura 1

Neste projeto, foi desenvolvida uma versão modificada do método de busca local *Variable Neighborhood Descent* (VND), que utiliza um modelo *multithread* para explorar simultaneamente diversas configurações de vizinhança. Combinado com o método *ILS* [3], onde o algoritmo realiza buscas iterativas por soluções de melhor qualidade. O *ILS* alterna entre etapas de intensificação, que aplicam busca local para refinar a solução corrente, e etapas de diversificação, que introduzem perturbações na solução para escapar de ótimos locais, garantindo uma exploração mais ampla do espaço de soluções. Essa abordagem permite intensificar a busca por melhores soluções, retornando o valor ótimo encontrado entre as múltiplas vizinhanças avaliadas. A versão proposta combina eficiência computacional com a capacidade de encontrar soluções próximas ao ótimo global, mesmo em instâncias de alta complexidade, destacando-se em relação às abordagens convencionais.

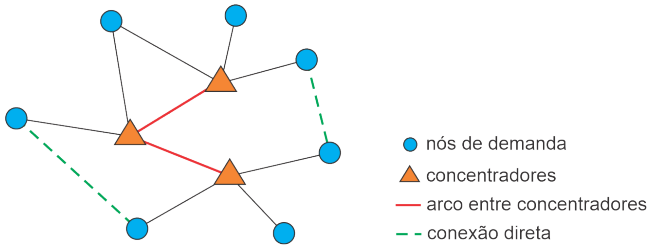

Figura 1: Exemplo de rede de concentradores

2 Metodologia

Os experimentos foram realizados em um computador com processador Intel Core i7-8700, 16 GB de RAM e sistema operacional Ubuntu 22.04 LTS em modo terminal. O algoritmo heurístico foi implementado em *C++* utilizando a biblioteca *OpenMP* para paralelismo e testado para resolver as formulações propostas. Para comparação, utilizou-se o *solver* CPLEX da IBM na resolução exata das instâncias. Os códigos e testes estão no repositório do *github* [1]

O conjunto de dados empregado foi o AP, introduzido por [1], baseado no sistema postal da Austrália. Esse conjunto inclui instâncias com até 200 nós, fornecendo valores de demanda relacionados à movimentação de correspondências e encomendas. O custo de transporte foi calculado com base na distância euclidiana entre os nós, considerando dois tipos de custos fixos para instalação de concentradores: *tight* (T) com valores mais próximos e *loose* (L) com valores mais espaçados. O conjunto AP é amplamente utilizado para avaliar a escalabilidade de algoritmos e testar modelos aplicados a redes logísticas complexas.

[1] https://github.com/Heidoco/HLP

O experimento realizado para verificar a validade da heurística proposta seguiu três etapas principais. Inicialmente, as instâncias foram resolvidas utilizando o *solver* CPLEX da IBM para obtenção dos valores ótimos ou de referência. Em seguida, foi aplicada a heurística *Smart ILS*[3], desenvolvida para explorar soluções de alta qualidade de forma eficiente, mas sem utilizar paralelização. Por fim, foi executado o *ILS* com *RVND paralelo*, uma variação do *Randomized Variable Neighborhood Descent* adaptada para processamento paralelo, visando melhorar o desempenho computacional e a qualidade das soluções em redes maiores. Esses métodos foram comparados em termos de tempo de execução e qualidade das soluções obtidas. Para cada instância, os algoritmos heurísticos foram executados 10 vezes; a média dos resultados obtidos foi utilizada para análises e comparações.

3 Algoritmo Heurístico Paralelo

O algoritmo heurístico utiliza como procedimento de busca local o método *Random Variable Neighborhood Descent* (RVND)[4][5]. O RVND explora o espaço de soluções por meio da troca sistemática de estruturas de vizinhança em ordem aleatória, buscando iterativamente soluções que aprimorem a solução corrente. Esse procedimento é realizado até que não sejam mais identificadas melhorias, garantindo uma exploração eficiente do espaço de soluções. As seguintes vizinhanças foram consideradas:

- $\mathcal{N}_1(s)$: soluções geradas pela instalação de um novo concentrador, com tamanho igual ao número de concentradores não instalados.
- $\mathcal{N}_2(s)$: soluções obtidas pela remoção de um concentrador instalado, excluindo também os arcos conectados a ele, com tamanho igual ao número de concentradores instalados.
- $\mathcal{N}_3(s)$: soluções criadas pela instalação de um arco entre concentradores já instalados, com tamanho igual ao número de arcos não instalados.
- $\mathcal{N}_4(s)$: soluções geradas pela remoção de um arco entre concentradores, com tamanho igual ao número de arcos instalados.
- $\mathcal{N}_5(s)$: soluções obtidas pela instalação de um novo concentrador conectado a todos os demais concentradores, com tamanho igual ao número de concentradores não instalados.
- $\mathcal{N}_6(s)$: soluções criadas pela troca de um concentrador instalado por outro não instalado, excluindo os arcos conectados ao concentrador removido, com tamanho proporcional ao produto do número de concentradores instalados pelo número de concentradores não instalados.

O Algoritmo 1 apresenta o procedimento de busca local RVND. Esse método funciona explorando inicialmente a primeira vizinhança de uma ordem gerada aleatoriamente. Caso nenhuma melhoria na solução atual seja identificada, o algoritmo avança para a próxima vizinhança, repetindo esse processo até que todas as vizinhanças tenham sido analisadas. Sempre que uma solução melhor é encontrada, a busca reinicia na primeira vizinhança da lista, assegurando que, ao término da execução, a solução final seja a melhor possível dentro do conjunto de vizinhanças exploradas.

O Algoritmo 2 implementa o *Iterated Local Search* (ILS) com uma abordagem paralela para acelerar a busca por soluções. Durante cada iteração, múltiplas *threads* trabalham simultaneamente, aplicando diferentes perturbações baseadas no identificador da *thread* variando de 1 até 3. As perturbações são aplicações aleatórias da vizinhança 6 até atingir o nível de perturbação desejado. Após a perturbação, cada *thread* executa uma busca local independente utilizando o método *Variable Neighborhood Descent* (VND) para refinar sua solução. Ao término, as soluções encontradas pelas *threads* são comparadas, e a melhor é selecionada como solução global. Essa paralelização permite que diferentes áreas do espaço de soluções sejam exploradas simultaneamente, aumentando a eficiência e reduzindo o tempo total de execução do algoritmo.

Algorithm 1 *Random Variable Neighborhood Descent* - RVND

Entrada: Solução s, Conjunto de vizinhanças $\mathcal{N}$
Saída: Solução refinada s

$\mathcal{NR} \leftarrow \mathcal{N}$ em ordem aleatória
$i \leftarrow 1$
while $i \leq |\mathcal{N}|$ **do**
    Encontre o melhor vizinho $s' \in \mathcal{NR}_i(s)$
    if $f(s') > f(s)$ **then**
        $s \leftarrow s'$
        $i \leftarrow 1$
    else
        $i \leftarrow i + 1$
    end if
end while
return s

Algorithm 2 ILS Paralelo

Entrada: Número máximo de iterações sem melhora $iter\_max$
Saída: Melhor solução encontrada

$num\_threads \leftarrow$ número de *threads* disponíveis
$iter \leftarrow 0$
while $iter < iter\_max$ **do** *thread*
    for *thread* t **do**
        Aplicar $Perturbacao(t \bmod 4 + 1)$
        Executar $RVND()$
        Salvar soluções de t
    end for
    Selecionar a melhor solução entre as *threads*
    if solução atual > melhor solução **then**
        Atualizar melhor solução
        $iter \leftarrow 0$
    else
        $iter \leftarrow iter + 1$
    end if
end while
return melhor solução

4 Resultados Computacionais

Nesta seção, são apresentados os resultados obtidos nos experimentos realizados utilizando o conjunto de dados AP. Foram conduzidos dois conjuntos de testes, com o objetivo de avaliar o desempenho do

algoritmo paralelo ILS. O foco foi verificar a qualidade das soluções encontradas e sua eficiência em termos de tempo computacional. Neste estudo, o fator de desconto por transporte através de concentrador foi fixado em 0,75 e foram analisados três valores de receita: 20, 30 e 50 unidades. O conjunto de dados fornece as demandas e os custos de transporte entre pares de nós na rede, bem como os custos fixos de instalação de concentradores, categorizados como *loose* (L) e *tight* (T). As instâncias testadas contêm 40, 50, 75, 100, 125, 150 e 200 nós. Para cada instância, foram realizadas 10 execuções do algoritmo heurístico paralelo, avaliando sua capacidade de explorar o espaço de soluções, a qualidade das soluções e o tempo de execução.

4.1 Comparação entre CPLEX e ILS paralelo

A Tabela 1 apresenta uma comparação entre os resultados obtidos pelo ILS paralelo e o CPLEX para as instâncias do conjunto de dados AP com até 75 nós. As colunas incluem o valor ótimo obtido pelo CPLEX, o tempo de execução necessário para alcançar esse valor, o valor médio encontrado pelo ILS paralelo, a quantidade de execuções em que o algoritmo heurístico atingiu ou se aproximou suficientemente do valor ótimo (dentro de uma tolerância de 0,1 unidade), e o tempo médio de execução do ILS. Observa-se que, para a maioria das instâncias, o ILS paralelo conseguiu encontrar o valor ótimo em todas as execuções, demonstrando alta precisão e eficiência. No entanto, para algumas instâncias, como 50L (50), 75L (30) e 75L (50), a quantidade de ótimos atingidos foi menor, indicando que o desempenho do ILS pode ser sensível à complexidade da instância. Adicionalmente, o tempo médio do ILS paralelo foi significativamente inferior ao do CPLEX em todas as instâncias, evidenciando a vantagem computacional do método heurístico.

Tabela 1: Resultados do ILS paralelo e do CPLEX com as instâncias AP contendo até 75 nós

Instância	Valor Ótimo CPLEX	Tempo CPLEX (s)	Valor Médio Paralelo	Quantidade de ótimos	Tempo Médio Paralelo (s)
40L (20)	15.860,10	118,12	15.860,10	10	0,01
40L (30)	44.888,70	1.058,85	44.888,70	10	0,06
40L (50)	120.130,00	3.423,90	120.130,00	10	0,06
40T (20)	6.690,35	68,35	6.690,35	10	0,01
40T (30)	33.383,60	193,92	33.383,60	10	0,02
40T (50)	106.719,00	197,61	106.719,00	10	0,05
50L (20)	15.003,60	408,74	15.003,60	10	0,01
50L (30)	43.682,50	7.041,48	43.682,50	10	0,10
50L (50)	119.992,00	26.402,76	119.340,59	0	0,18
50T (20)	8.338,91	187,39	8.338,91	10	0,01
50T (30)	31.787,10	599,80	31.787,10	10	0,05
50T (50)	105.523,00	649,68	105.501,98	9	0,05
75L (20)	14.424,70	8.089,10	14.424,70	10	0,03
75L (30)	44.670,00	31.517,34	43.985,90	0	0,30
75L (50)	120.653,00	26.796,84	119.524,02	0	0,89
75T (20)	1.144,13	1.951,32	1.144,13	0	0,03
75T (30)	25.999,60	3.731,06	25.999,60	10	0,07
75T (50)	99.186,30	2.456,42	99.186,30	10	0,13

O gráfico da figura 2 retrata a proximidade entre os valores médios obtidos pelo ILS paralelo e os valores ótimos fornecidos pelo CPLEX para as instâncias analisadas. A linha de identidade, traçada em azul, representa a correspondência ideal entre os dois métodos, enquanto a dispersão dos pontos reflete a precisão relativa do ILS em relação aos valores ótimos. O gráfico destaca que, para a maioria das instâncias, o ILS apresenta resultados consistentes e próximos ao ótimo, com desvios pontuais em instâncias de maior complexidade.

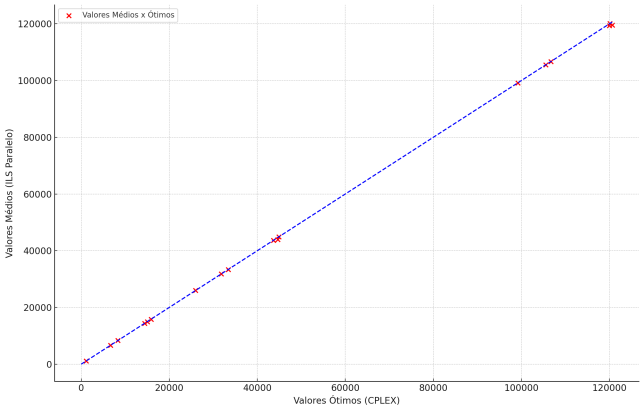

Figura 2: Dispersão entre os valores CPLEX e ILS paralelo

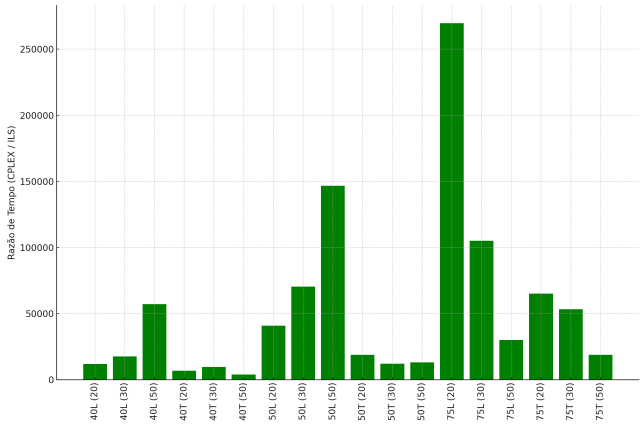

Figura 3: Razão entre tempo CPLEX e ILS

A Figura 3 apresenta a razão entre os tempos de execução do CPLEX e do ILS paralelo para as instâncias analisadas. Cada barra do gráfico representa o número de vezes que o ILS paralelo foi mais rápido do que o CPLEX para resolver uma determinada instância. Observa-se que, para a maioria das instâncias, o ILS paralelo apresenta uma vantagem computacional significativa, com tempos até centenas de milhares de vezes inferiores aos do CPLEX. Esse desempenho ressalta a eficiência do método heurístico proposto, especialmente em cenários de maior complexidade computacional, onde a execução do CPLEX é substancialmente mais demorada.

4.2 Comparação entre Smart ILS e ILS paralelo

A Tabela 2 apresenta os resultados obtidos pelo ILS paralelo e pelo Smart ILS para as instâncias do conjunto de dados AP contendo entre 100 e 200 nós. A tabela compara os valores encontrados pelos dois métodos heurísticos, destacando o valor da melhor solução encontrada e os tempos de execução médios para cada abordagem. Observa-se que, em todas as instâncias, ambos os métodos obtiveram valores iguais ou muito próximos, evidenciando a qualidade das soluções geradas. Contudo, o tempo de execução do ILS paralelo

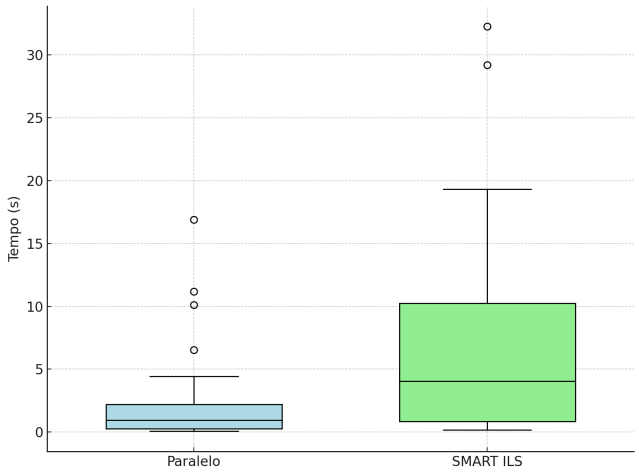

Figura 4: Comparacao de tempo entre Smart ILS e Paralelo

foi menor na maioria dos casos, demonstrando sua eficiência em relação ao Smart ILS.

Tabela 2: Resultados obtidos pelo ILS paralelo e Smart ILS com as instâncias AP contendo de 100 a 200 nós

Instância (Receita)	Valor Paralelo	Valor Smart ILS	Tempo Smart ILS	Tempo Paralelo (s)
100L (20)	13943.89	13943.89	0.57	0.07
100L (30)	43044.59	43022.68	4.57	0.78
100L (50)	118102.64	118102.64	0.30	1.37
100T (20)	2116.57	2116.57	0.39	0.06
100T (30)	25271.32	25271.32	0.31	0.16
100T (50)	98113.77	98113.77	0.16	0.32
125L (20)	15135.19	15135.19	2.15	0.54
125L (30)	45080.31	45081.60	6.03	1.46
125L (50)	120104.21	120104.21	9.40	1.94
125T (20)	7169.77	7169.77	1.00	0.13
125T (30)	29586.11	29586.11	0.88	0.14
125T (50)	93251.38	93251.38	0.81	0.13
150L (20)	14540.77	14540.77	3.53	0.61
150L (30)	44168.96	44168.96	12.75	3.01
150L (50)	119329.25	119320.76	14.56	4.43
150T (20)	10936.81	10936.81	6.48	0.62
150T (30)	38294.30	38294.30	1.18	1.11
150T (50)	111271.76	111271.76	1.09	1.95
200L (20)	13660.35	13660.35	5.16	0.80
200L (30)	43253.98	43253.98	29.21	11.19
200L (50)	118304.70	118304.70	17.93	16.88
200T (20)	7392.64	7392.64	7.15	1.65
200T (30)	35317.26	35317.26	19.31	6.54
200T (50)	109676.37	109676.37	32.27	10.10

Na Figura 4, observa-se que o ILS paralelo apresenta tempos de execução concentrados em valores baixos, com uma média de 2,75 segundos e uma mediana ainda menor, refletindo sua eficiência e consistência em manter tempos reduzidos. Por outro lado, o Smart ILS apresenta maior variabilidade nos tempos, com uma média de 7,38 segundos e uma mediana inferior à média, indicando a influência de valores extremos. Essa diferença evidencia que o paralelismo contribui diretamente para a eficiência do ILS paralelo, permitindo que ele reduza significativamente os tempos de execução em comparação ao Smart ILS.

5 Conclusão

A implementação do ILS paralelo apresentou resultados satisfatórios em termos de qualidade, alcançando valores próximos aos ótimos obtidos pelo CPLEX. Ao mesmo tempo, a abordagem paralela superou o Smart ILS em termos de eficiência computacional. A capacidade de executar a busca local de forma simultânea em múltiplas regiões do espaço de soluções permitiu reduzir o número de iterações necessárias para atingir bons resultados, resultando em um tempo total de execução menor.

A redução do número de iterações, associada à exploração paralela de vizinhanças, não apenas acelerou o processo de otimização, como também manteve a qualidade das soluções. Assim, o emprego do paralelismo emerge como uma estratégia que impulsiona o desempenho de heurísticas complexas, viabilizando a análise de instâncias maiores sem comprometer a aderência aos valores ótimos conhecidos, o que torna a abordagem adequada para aplicações práticas em ambientes logísticos e operacionais.

Referências

[1] Andreas T. Ernst and Mohan Krishnamoorthy. 1996. Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science* 4 (1996), 139–154. Issue 3.

[2] Fabricio Alves Oliveira, Elisangela Martins de Sá, and Sérgio Ricardo de Souza. 2022. Benders decomposition applied to profit maximizing hub location problem with incomplete hub network. *Computers & Operations Research* 142 (2022), 105715.

[3] J. A. Reinsma, P. H. V. Penna, and M. J. F. Souza. 2018. Um algoritmo simples e eficiente para resolução do problema do caixeiro viajante generalizado (in Portuguese). In *Anais do 50º Simpósio Brasileiro de Pesquisa Operacional*. Galoá, Rio de Janeiro, Brazil. Available at https://proceedings.science/sbpo/papers/um-algoritmo-simples-e-eficiente-para-resolucao-do-problema-do-caixeiro-viajante-generalizado.

[4] M.J.F. Souza, I.M. Coelho, S. Ribas, H.G. Santos, and L.H.C. Merschmann. 2010. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research* 207, 2 (2010), 1041–1051. https://doi.org/10.1016/j.ejor.2010.05.031

[5] A. Subramanian, L.M.A. Drummond, C. Bentes, L.S. Ochi, and R. Farias. 2010. A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers & Operations Research* 37, 11 (2010), 1899–1911. https://doi.org/10.1016/j.cor.2009.10.011 Metaheuristics for Logistics and Vehicle Routing.