Problema de Mínima Latência

Trabalho de Inteligência Artificial

Guilherme Dantas

PUC-Rio

Maio de 2020

Sumário

Introdução

Foram desenvolvidas duas heurísticas:

- *Iterated Local Search* + *Simulated Annealing* + *VNS* [1]
- Algoritmo Genético + *VNS*

A linguagem de programação escolhida foi C++.

[1] Variable Neighbourhood Search

Características da Busca Local

A implementação apresenta algumas otimizações:

- k-vizinhança (*Gamma Set*)
- Cálculo do custo de movimentos de vizinhança
- *Lower Bound* e *Upper Bound* no *Local Search*

k-vizinhança ou *Gamma Set*

É o conjunto dos k nós mais próximos de um dado nó.
A busca local opera somente entre os nós e seus k-vizinhos.
Complexidade na ordem de $O(kn)$, ao invés de $O(n^2)$.

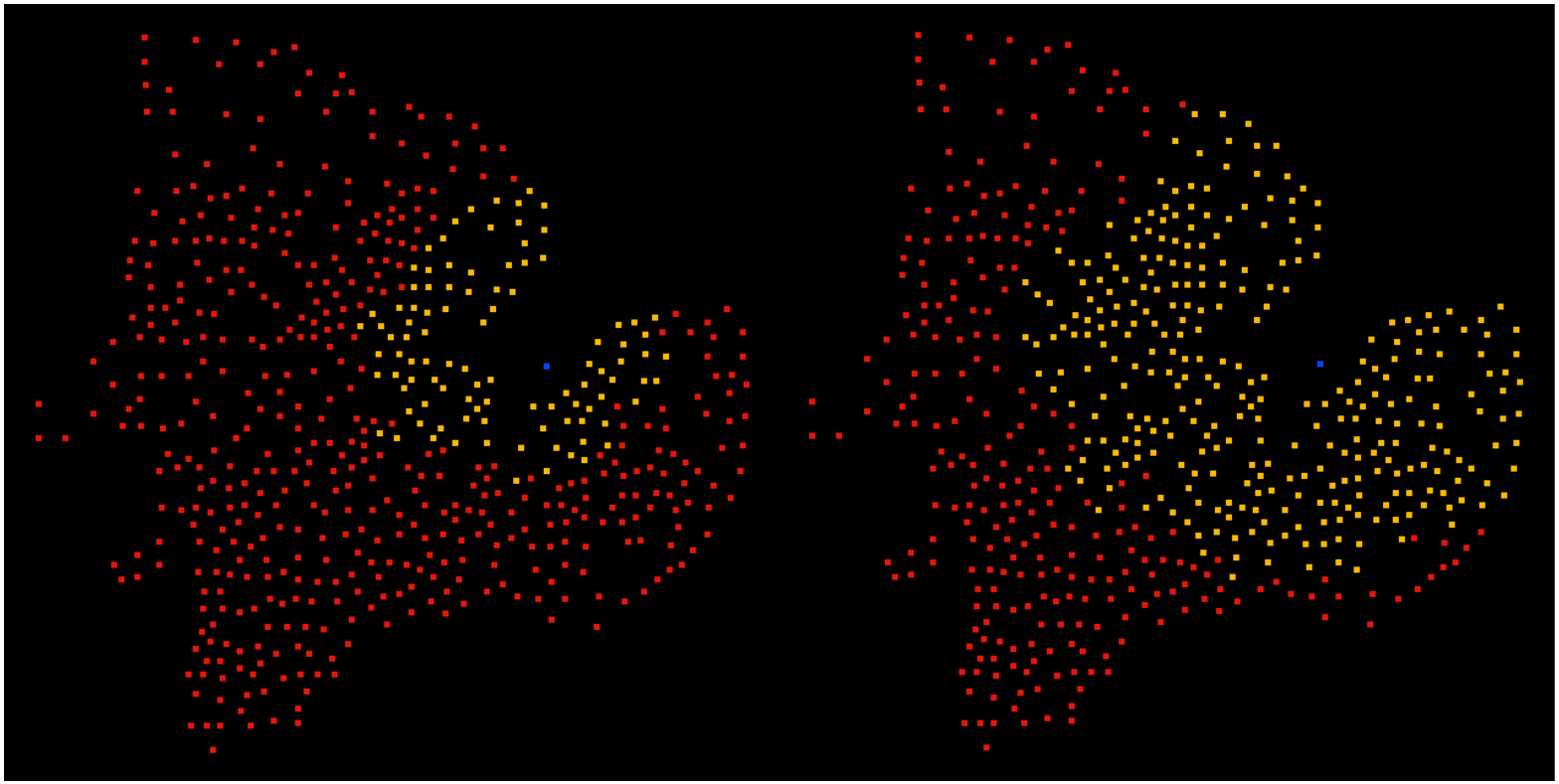

Figura: $k = 100$ .vs. $k = 300$

Cálculo do custo de movimento de vizinhança

Antes de aplicar um movimento, é calculado o seu custo para se certificar que é vantajoso (isto é, o custo da solução diminui).

Conceito

É chamado de "custo"de um movimento a diferença entre o custo da solução depois e antes de aplicado um movimento.

Para as vizinhanças definidas, o cálculo do custo tem complexidade $O(1)$, e, no caso especial do 2-opt, $O(k)$, aonde k é a diferença entre o primeiro e o último nó.

Lower Bound e Upper Bound

Aplicado um movimento $m(p, q)$, qualquer movimento que não envolve nós entre p e q possui o mesmo custo.
Assim, é possível rejeitar movimentos que são sabidamente desvantajosos.

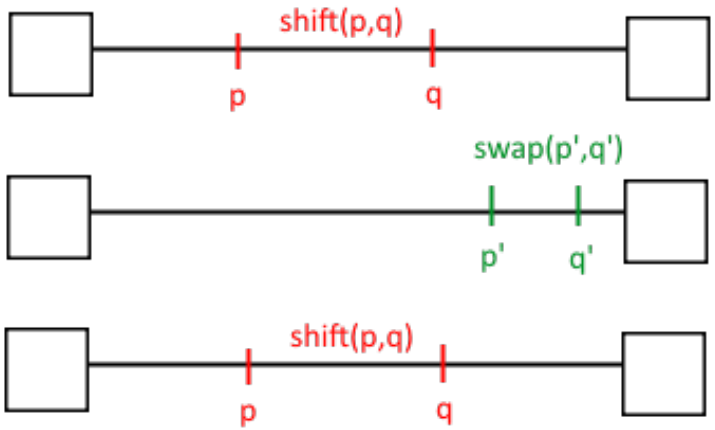

Figura: Exemplo de aplicação do Upper Bound

Vizinhanças

O *VNS* foi utilizado tanto para a busca local quanto para a perturbação em soluções, usado em ambas as heurísticas desenvolvidas.
As seguintes vizinhanças são iteradas em ordem de prioridade.

- Shift(p,q)
- 2-Opt(p,q)
- Swap(p,q)
- Shift2(p,q,r)

Local Search

```
t <- 0
repeat:
    improved_once <- false
    for p in Clients(S, rng):
        for q in GammaSet(p, rng):
            feasible, delta <- Test(S, N(t), p, q)
            if not feasible or
               (must_improve and delta >= 0):
                continue
            S <- Apply(S, N(t), p, q)
            improved_once <- true
            t <- 0
    if not improved_once:
        t <- t + 1
while t < |N|
```

Perturbação

```
t <- 0
s <- perturbation_size
repeat:
    for p in Clients(S):
        for q in GammaSet(p):
            feasible, size <- Test(S, N(t), p, q)
            if (not feasible) or
               (size > s):
                continue
            s <- s - size
            S <- Apply(S, N(t), p, q)
            t <- (t + 1) % N
while s > 0
```

Iterated Local Search

```
S0 <- InitialSolution;
S <- LocalSearch(S0, rng);
repeat:
  S' <- Perturbation(S, history, rng);
  S'' <- LocalSearch(S', rng);
  S <- AcceptanceCriterion(S, S'', history);
until stopping criterion is not satisfied anymore
return S;
```

A perturbação segue uma exponencial decrescente da forma $p = p_0 \cdot e^{-i/I}$, aonde p_0 e I são parâmetros do algoritmo.

Algoritmo genético

```
mating_pool <- binary_tournament(P, pool_size, rng)
for wife, husband in mating_pool:
    if wife == husband:
        continue
    offspring <- crossover_cx(wife, husband, rng)
    if should_apply_mutation(mutation_chance, rng):
        p <- unif(pmin, pmax, rng)
        offspring <- perturb(offspring, p, rng)
        offspring <- local_minima(offspring, rng)
    P <- P U {offspring}
if |P| > maxsize:
    P <- P - clones(P)
    repeat:
        P <- P - worst(P)
    while |P| > minsize
```

Resultados - ILS

Instância	Gap (%)	Tempo (s)
brazil58	**0,00%**	< 1
dantzig42	**0,00%**	< 1
gr120	-0,60%	27
gr48	**0,00%**	< 1
pa561	-3,88%	1325
MÉDIA	-0,90%	270

Tabela: Resultados médios do algoritmo **ILS**

Resultados - GA

Instância	Gap (%)	Tempo (s)
brazil58	**0,00%**	< 1
dantzig42	**0,00%**	< 1
gr120	-0,66%	35
gr48	**0,00%**	< 1
pa561	-1,48%	10900
MÉDIA	-0,43%	2187

Tabela: Resultados médios do algoritmo **genético**

Conclusões

Mesmo usando o mesmo algoritmo de busca local e de perturbação...

- ILS obteve bons resultados mais rapidamente
- GA obteve resultados melhores para a maior instância

Para instâncias pequenas ($n < 120$), a solução ótima foi obtida por ambas as heurísticas, e em períodos de tempo igualmente curtos.