Paralelização de Metaheurísticas de Otimização

Trabalho Final - Programação Distribuída e Paralela

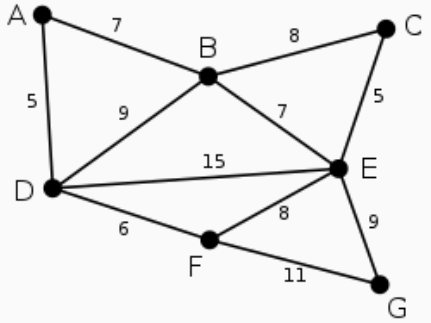
Fernando Garcia Bock

Motivação

- Problemas de otimização são encontradas em variadas áreas: logística, roteamento, alocação de recursos e tarefas.
- Método usual envolve utilização de metaheurísticas, ordens de magnitude mais rápidas que soluções exatas, porém sem garantias.
- O crescente volume de dados envolvidos nos problemas requer métodos mais eficientes para realizar sua avaliação.
- Possível solução? Paralelização de heurísticas.

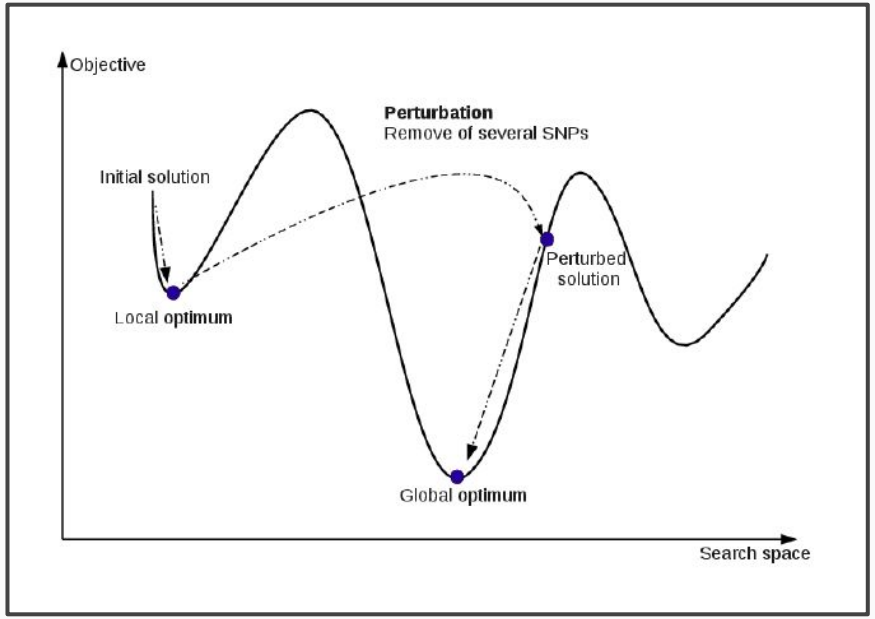
Problema de Teste

- **SOP - Problema de Ordenamento Sequencial**
- Entradas: matriz de grafo ponderado com dimensão N e uma lista de restrições.
- Restrições indicam que um nodo X não pode ser percorrido antes de um nodo Y.
- Dado estas entradas, encontrar o caminho de menor peso do nodo inicial ao nodo final.
- Presume-se que o nodo de índice 0 e o nodo de índice N são o inicial e final.



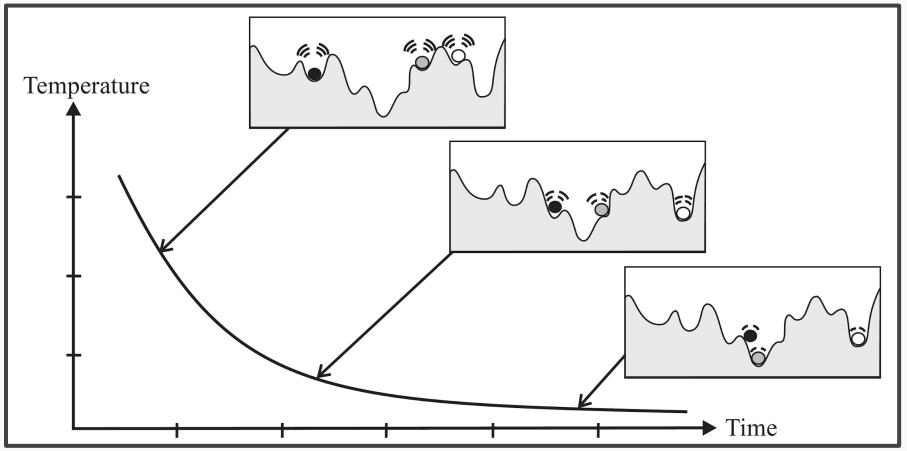
Metaheurísticas

- **Iterated Local Search**
- Encontra uma solução inicial.
- Dada solução inicial, realiza sucessivas “perturbações” (modificações) e então buscas locais sobre a solução perturbada.
- Caso nova solução seja melhor, guarda ela.
- Processo termina quando chega na condição parada - usualmente número de iterações.



Metaheurísticas

- **Simulated Annealing**
- Realiza uma busca local sobre o grafo.
- Diferente do ILS, aceitação depende de uma função probabilística que recebe “temperatura” como parâmetro (a utilizada é baseada na distribuição de Boltzmann).
- Maior chance de aceitar um novo estado de valor pior quando a “temperatura” é maior.
- Isso permite escapar de mínimos locais.



Implementação

- **Linguagem:** C++
- **Ferramentas:** OpenMP (paralelização), Bash (testagem)
- **Iterated Local Search:**
  - Vizinhança de busca local definida como swap de quaisquer nodos do caminho.
  - Paralelização do loop principal que chama as buscas locais, utilizando uma “área crítica” para controlar comparação e acesso à “melhor solução” durante a execução do loop.
- **Simulated Annealing:**
  - Vizinhança foi definida de forma mais simples: somente são considerados swaps entre alguns nodos próximos.
  - Paralelização do processo de comparação de vizinhos dentro da busca local, somente realiza comparação entre os diferentes swaps possíveis de forma paralela.

Resultados

- Conforme esperado, em alguns casos é possível aumentar número de iterações melhora os resultados obtidos.
- Exemplo:

ESC25.sop	
Best Value	Iterations
2745	10000
2376	15000
2223	20000

- Em nem todos os casos isto foi possível: possível problema de implementação?

Resultados

- Em todos os casos de teste para o ILS houve um resultado similar em termos de melhoria da performance, independente da entrada.
- Para o caso ilustrativo abaixo os tempos são dados em segundos.
  - Comparação caso de 3 threads e 15000 iterações vs. 1 thread e 5000 iterações.

ESC07.sop	Threads			
Iterations	**1**	**2**	**3**	**4**
5000	0.267	0.127	0.090	0.073
10000	0.488	0.253	0.177	0.153
15000	0.742	0.376	0.262	0.230
20000	0.976	0.501	0.342	0.304
Speedup Médio	-	1.987	2.856	3.324
Eficiência Média	-	0.116	0.053	0.035

Conclusões

- Paralelização pode ser uma boa alternativa para reduzir o tempo de execução.
- Como visto no slide anterior, é possível aumentar o número de iterações relativo ao número de threads paralelas, para chegar num resultado melhor em um mesmo tempo.
- Métodos mais inteligentes de paralelização poderiam melhorar resultados. Artigos consultados para realização mencionaram técnicas de paralelização mais complexas usando GPUs.
- Outras heurísticas podem se mostrar mais bem adaptadas - somente foram analisadas duas metaheurísticas, e não foram executados os testes completos para o Simulated Annealing, impedindo a comparação.

Obrigado!

- Relatório conterá mais informações!
- Perguntas?