Implementação em C++

1.1 computador utilizado

O computador utilizado para desenvolvimento e testes é um desta implementação é um notebook Samsung core i5, com 8 gigabytes de memória RAM, 64 bits. Sistema operacional linux Mint 18.0, IDE codeblocks 16.01 com compilador c++ 14 e com gerador de gráficos GNUPLOT 5.0.

2.2 Aplicação

A aplicação desenvolvida possui 2 arquivos: “Trabalho.cpp” e “Trabalho.h” e implementa o pseudocódigo mostrado na introdução, ou seja, a forma clássica do Simulated Annealing (SA) para o problema proposto no estudo de caso.

O primeiro tem a chamada de execução para os métodos que estão contidos no segundo. No arquivo Trabalho.cpp possui o método principal com os parâmetros para o grafo, bem como os parâmetros para o método simulated\_Annealing().

No arquivo Trabalho.h tem os seguintes métodos:

* Método construtor – inicializando o objeto Grafo com um tamanho, lista de adjacências e uma matriz a ser preenchida com as distâncias;
* addAresta - método para adicionar aresta;
* peencherMatriz - método para preencher a matriz do construtor;
* mostarSolucao - método que imprime na tela uma solução qualquer a partir de uma lista de solução;
* dijkstra - método que calcula a distância entre duas cidades;
* gravarArquivo - gravarArquivo2() e gravarArquivo3() - métodos que geram arquivos do tipo texto com os dados de cada execução da aplicação a serem usados para plotar os gráficos;
* num Aleatorio - gera números aleatórios dentro de um intervalo;
* numAleatorio\_01 - gera números aleatórios entre 0 e 1;
* custoSolucao – mostra o custo em quilômetros a partir de uma solução;
* solucaoAleatoria – gera uma solução aleatoriamente;
* vizinho1, vizinho2 e vizinho3 – geram soluções vizinhas a partir de uma outra solução;
* gerarVizinho – gera um vizinho aleatório utilizado os métodos vizinho1 e vizinho2;
* simulated\_Annealing – utiliza a maioria dos métodos anteriores para gerar uma solução otimizada baseada nos custos das soluções geradas.

A implementação não possui interface gráfica e utiliza o console (terminal) para sua execução e mostra dos resultados. Há também a geração de arquivos textos com resumo das informações importantes.

Figura 1: Exemplo de resumo contido em arquivo de texto.

**Simulação 1**

**Solução aleatória:**

0 -> 14 -> 12 -> 2 -> 6 -> 10 -> 13 -> 9 -> 15 -> 4 -> 11 -> 1 -> 7 -> 8 -> 16 -> 3 -> 5 -> 0

**Custo: 21286**

Temperatura Final: 9.96062e-30

Número máximo de iterações na Temperatura: 100

Número máximo de iterações total: 7143

--------------------------------------------

**Solução final do Simulated Annealing**

0 -> 4 -> 3 -> 5 -> 6 -> 16 -> 15 -> 9 -> 10 -> 11 -> 8 -> 12 -> 13 -> 14 -> 7 -> 1 -> 2 -> 0

**Custo: 11746**

--------------------------------------------------------------------------------------------------------

**Simulação 1**

**Solução aleatória:**

0 -> 14 -> 12 -> 2 -> 6 -> 10 -> 13 -> 9 -> 15 -> 4 -> 11 -> 1 -> 7 -> 8 -> 16 -> 3 -> 5 -> 0

**Custo: 21286**

Temperatura Final: 9.96062e-30

Número máximo de iterações na Temperatura: 100

Número máximo de iterações total: 7143

--------------------------------------------

**Solução final do Simulated Annealing**

0 -> 4 -> 3 -> 5 -> 6 -> 16 -> 15 -> 9 -> 10 -> 11 -> 8 -> 12 -> 13 -> 14 -> 7 -> 1 -> 2 -> 0

**Custo: 11746**

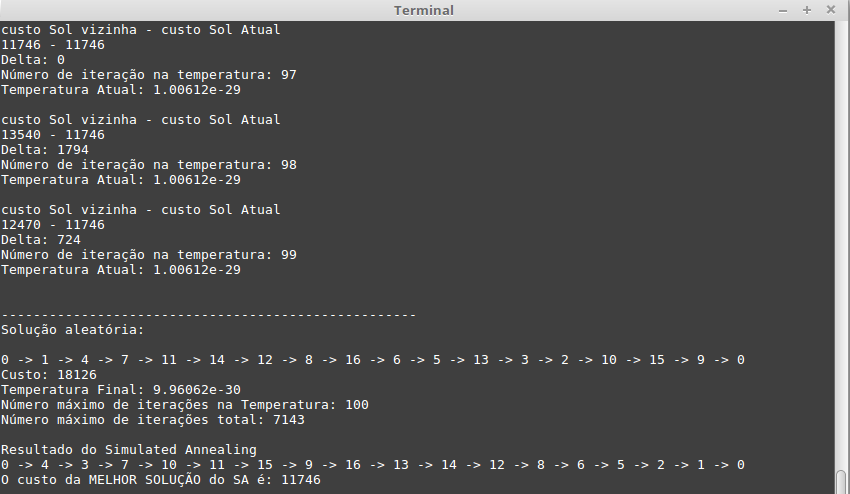
--------------------------------------------------------------------------------------------------------

Os parâmetros de entrada do método simulated\_Annealing são:

* alpha – fator de redução da temperatura;
* tempIni – temperatura inicial;
* itMaxTemp – máximo de iterações para cada temperatura;
* dado, dado2 e dado3 – são os nomes dos arquivos do tipo texto que serão gerados para plotar os gráficos.

As três primeiras variáveis acima são aquelas que podem ser configuradas com intuito de se obter um melhor resultado no algoritmo do simulated annealig.

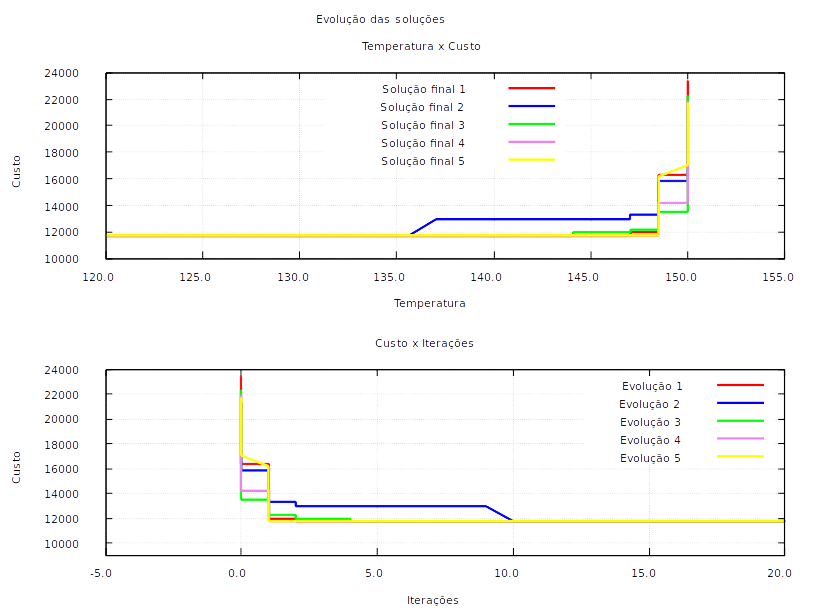
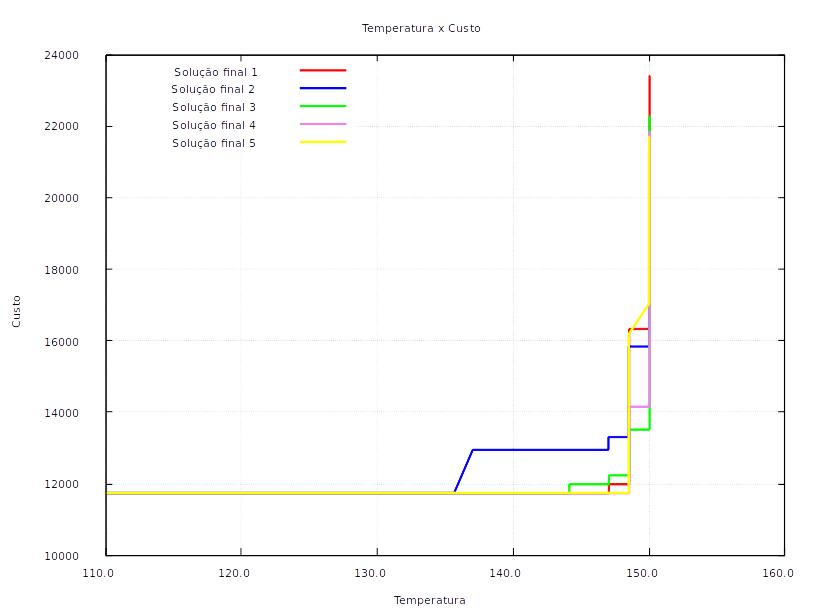
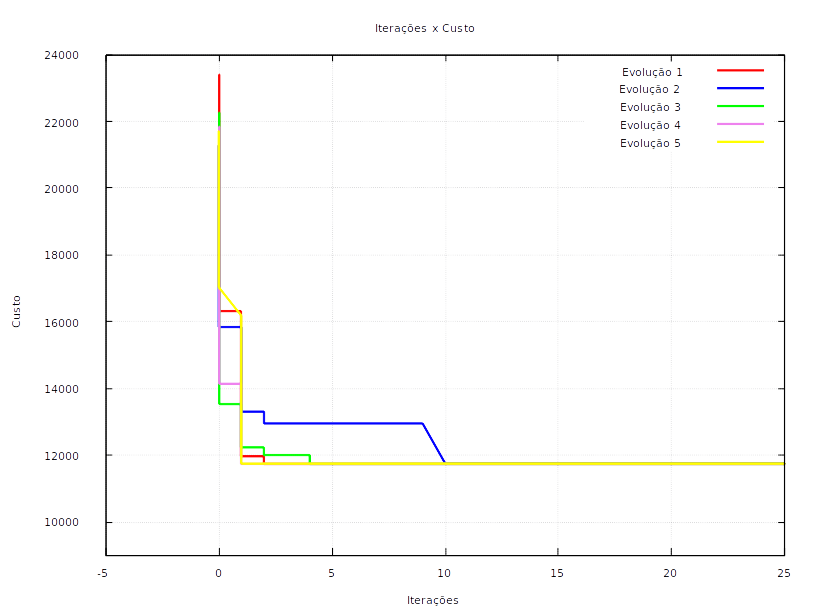
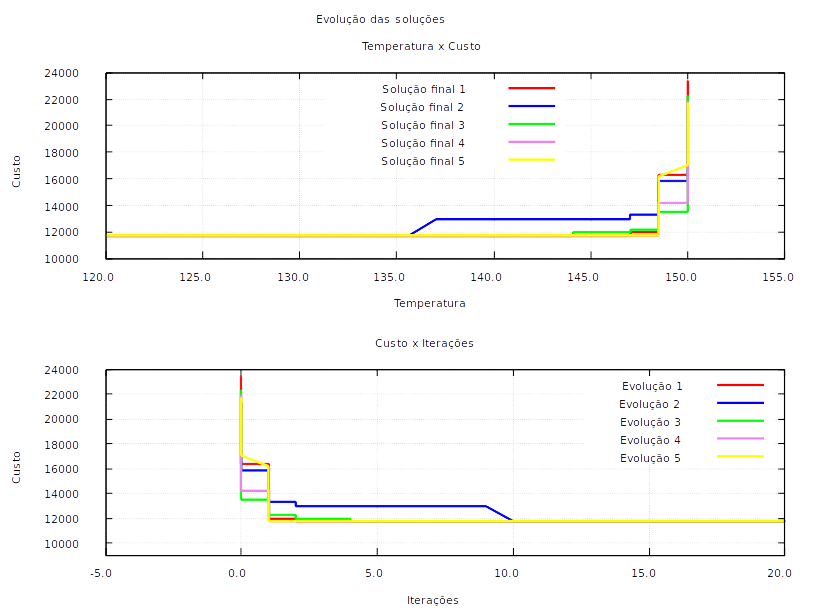
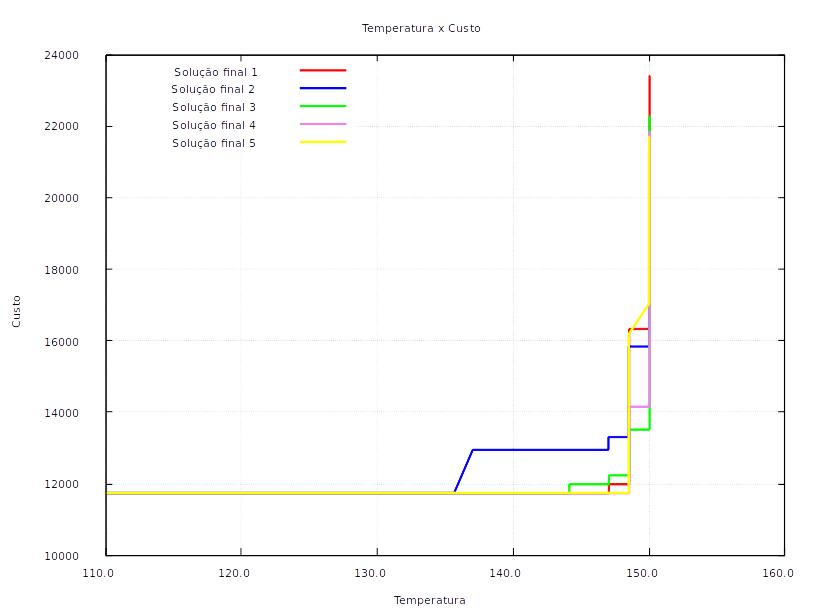
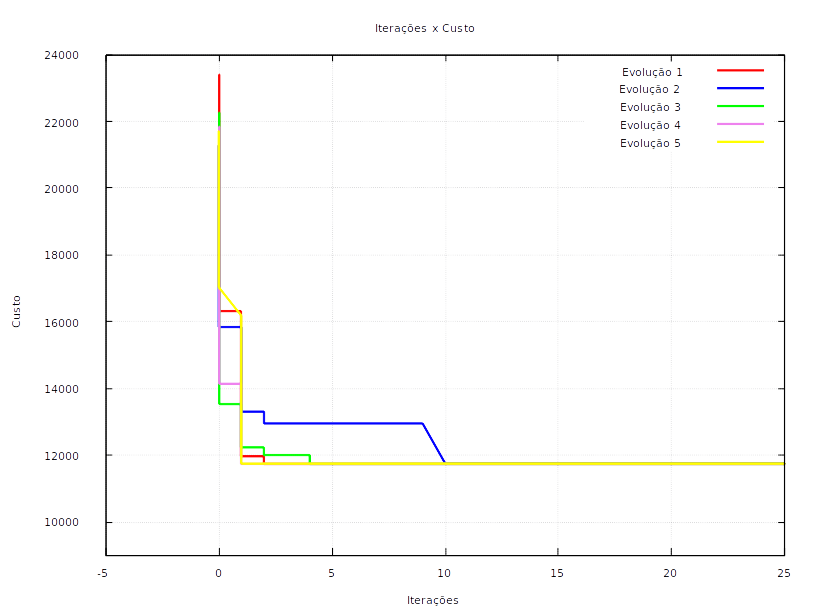
A execução do código produz resultados em tempo de execução mostrando a diferença entre os custos de uma solução vizinha gerada e o custo da solução atual (delta). E no final da execução mostra a solução aleatória – um conjunto de vértices ordenados configurando uma rota, mostra o custo desta solução – em quilômetros, a temperatura final – valor bem próximo de zero, o número máximo de iterações dentro de cada temperatura e o número total de iterações – total das temperaturas consideradas baseadas no fator de redução por números de iterações em cada temperatura.

Figura 2: Exemplo de Solução do SA

A finalização do código são gerados 03 (três) gráficos:

* Temperatura x Custo;
* Custo x Iterações;
* Gráfico com os dois anteriores com intervalos reduzidos.

Figura 3: Exemplo dos gráficos gerados



Para que os gráficos sejam gerados o código em c++ produz 03 (três) conjuntos de dados em arquivos de texto que são interpretados por 03 (três) scripts para o “gnuplot”, instalado no sistema operacional.

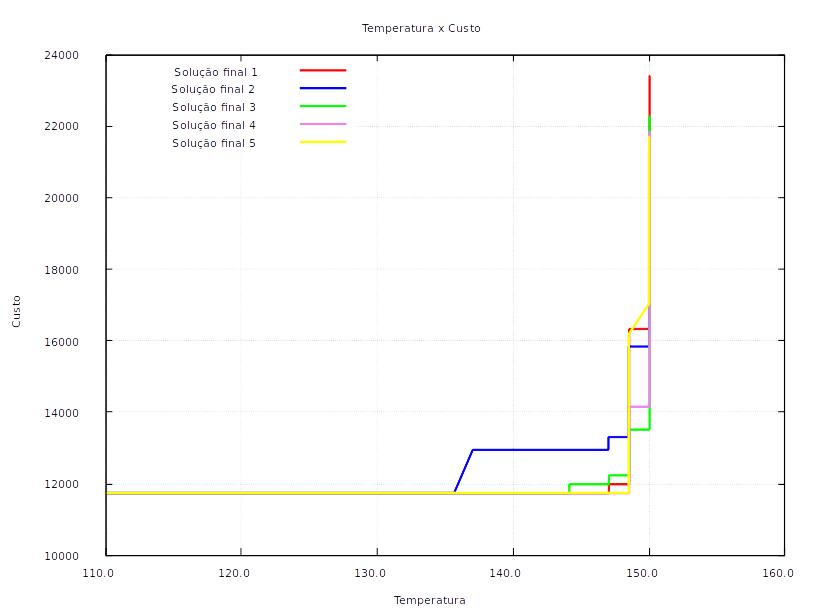
2.3 Dados e Análise

Para os testes foram configurados os seguintes dados:

* Alpha – 0.99;
* Temperatura inicial – 150;
* Número de iterações em cada temperatura – 100

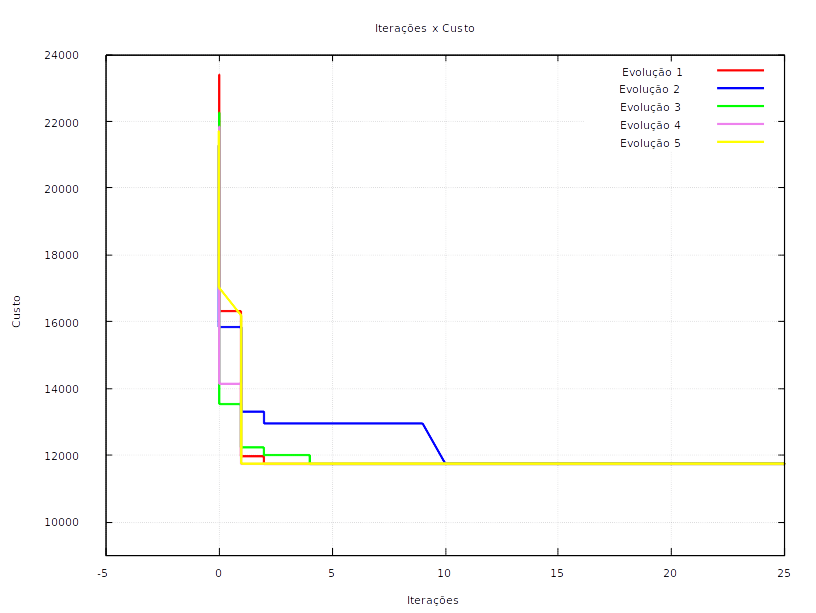
Baseado nesses valores o código foi executado 05 (cinco) vezes e foram gerados os dados em forma de gráficos e arquivos com o resumo das soluções obtidas.

Observemos o gráfico gerado pelas cinco execuções considerando a temperatura pelo custo:

  
Figura 4: Temperatura x Custo

O início das execuções sempre começa na temperatura 150, e no caso da busca da 5ª solução (amarelo) parte aproximadamente de 21964 km. Através do decréscimo da temperatura chega no mínimo de 11746 km. Aqui percebemos que toda e qualquer mudança de decréscimo no custo sempre ocorrerá nas primeiras dezenas das unidades de temperatura, a partir dai não há mais melhora de custos. Lembrando que a temperatura começa em 150 e termina zero (ou número muito próximo de zero) que não está representado no gráfico.

Outro gráfico gerado:

  
Figura 5: Iterações x Custo

Esse gráfico gerado nas mesmas três execuções verifica-se que na iteração 0 temos a maior quantidade de melhora de resultados. E ainda assim todas as melhoras se estendem até a iteração 10. Essas iterações são contadas a partir de 100 vezes a cada temperatura. No caso das configurações atuais temos 7143 iterações., contudo as melhorias só ocorrem nas iterações iniciais.

Uma análise bastante simples podemos concluir que na implementação do algoritmo clássico para o problema proposto as melhoras acontecem nas temperaturas iniciais ou nas iterações iniciais, portanto, começar com temperaturas altas ou números de iterações também grandes não vão trazer melhoras a mais ao problema proposto.

Na implementação em MATLAB foi incorporado o conceito de reaquecimento, onde em tempo de execução, sob determinadas condições, pode haver um aumento da temperatura na busca de melhores resultados.

Resumo das cinco simulações mostradas nos gráficos:

simulação 1

**Solução aleatória:**

0 -> 8 -> 4 -> 10 -> 2 -> 6 -> 9 -> 16 -> 15 -> 5 -> 11 -> 13 -> 7 -> 14 -> 1 -> 3 -> 12 -> 0

**Custo: 23420**

Temperatura Final: 9.96062e-30

Número máximo de iterações na Temperatura: 100

Número máximo de iterações total: 7143

**Solução final do Simulated Annealing**

0 -> 3 -> 4 -> 7 -> 8 -> 13 -> 14 -> 12 -> 16 -> 9 -> 15 -> 10 -> 11 -> 6 -> 5 -> 1 -> 2 -> 0

**Custo: 11746**

----------------------------------------------------------------------------------------------------

simulação 2

**Solução aleatória:**

0 -> 14 -> 12 -> 2 -> 6 -> 10 -> 13 -> 9 -> 15 -> 4 -> 11 -> 1 -> 7 -> 8 -> 16 -> 3 -> 5 -> 0

**Custo: 21286**

Temperatura Final: 9.96062e-30

Número máximo de iterações na Temperatura: 100

Número máximo de iterações total: 7143

**Solução final do Simulated Annealing**

0 -> 4 -> 3 -> 5 -> 6 -> 16 -> 15 -> 9 -> 10 -> 11 -> 8 -> 12 -> 13 -> 14 -> 7 -> 1 -> 2 -> 0

**Custo: 11746**

-----------------------------------------------------------------------------------------------------

simulação 3

**Solução aleatória:**

0 -> 3 -> 8 -> 9 -> 13 -> 10 -> 16 -> 1 -> 6 -> 14 -> 2 -> 4 -> 11 -> 5 -> 12 -> 15 -> 7 -> 0

**Custo: 22268**

Temperatura Final: 9.96062e-30

Número máximo de iterações na Temperatura: 100

Número máximo de iterações total: 7143

**Solução final do Simulated Annealing**

0 -> 3 -> 4 -> 7 -> 14 -> 13 -> 12 -> 8 -> 11 -> 10 -> 16 -> 15 -> 9 -> 6 -> 5 -> 1 -> 2 -> 0

**Custo: 11746**

-----------------------------------------------------------------------------------------------------

simulação 4

**Solução aleatória:**

0 -> 9 -> 5 -> 13 -> 6 -> 10 -> 14 -> 12 -> 7 -> 3 -> 8 -> 16 -> 15 -> 2 -> 4 -> 11 -> 1 -> 0

**Custo: 22714**

Temperatura Final: 9.96062e-30

Número máximo de iterações na Temperatura: 100

Número máximo de iterações total: 7143

**Solução final do Simulated Annealing**

0 -> 8 -> 12 -> 13 -> 14 -> 11 -> 10 -> 16 -> 9 -> 15 -> 7 -> 4 -> 3 -> 1 -> 2 -> 5 -> 6 -> 0

**Custo: 11746**

-----------------------------------------------------------------------------------------------------

simulação 5

**Solução aleatória:**

0 -> 4 -> 12 -> 7 -> 1 -> 16 -> 5 -> 6 -> 8 -> 2 -> 10 -> 3 -> 11 -> 14 -> 15 -> 13 -> 9 -> 0

**Custo: 21964**

Temperatura Final: 9.96062e-30

Número máximo de iterações na Temperatura: 100

Número máximo de iterações total: 7143

**Solução final do Simulated Annealing**

0 -> 2 -> 1 -> 6 -> 5 -> 14 -> 13 -> 12 -> 10 -> 11 -> 16 -> 15 -> 9 -> 8 -> 7 -> 4 -> 3 -> 0

**Custo: 11746**