## plot\_graph

Affiche un graphe étant données un ensemble de noeuds et d'arêtes.

Exemple:

```
graph_nodes, graph_edges = read_stsp("bayg29.tsp")
plot_graph(graph_nodes, graph_edges)
savefig("bayg29.pdf")
```

Fonction de commodité qui lit un fichier stsp et trace le graphe.

```
#the repository can be found on the github link:
   https://github.com/houskkam/mth6412b-starter-code
begin
import Pkg
Pkg.add("Plots")
include("projet\\phase1\\node.jl")
include("projet\\phase1\\edge.jl")
include("projet\\phase1\\graph.jl")
include("projet\\phase1\\graph.jl")
end
```

```
Resolving package versions...
No Changes to `C:\Users\ahoj7\.julia\environments\v1.9\Project.toml`
No Changes to `C:\Users\ahoj7\.julia\environments\v1.9\Manifest.toml`
```

```
▶ Dict(5 ⇒ [750.0, 2030.0], 16 ⇒ [1280.0, 1200.0], 20 ⇒ [590.0, 1390.0], 12 ⇒ [1170.]

1 #A file name (fn) is defined by obtaining the current working directory with pwd() and combining it with a relative path to a file named "bayg29.tsp."

2 begin

3 fn = pwd() * "\\instances\\stsp\\bayg29.tsp"

4 #reads the data from the specified file (fn) and stores it in various variables such as header, almost_edges, and almost_nodes.

5 header = read_header(fn)

6 almost_edges = read_edges(header, fn)

7 almost_nodes = read_nodes(header, fn)

8 end

9
```

```
# Constructing my_nodes of type Node from the given file
# starting with an empty array for nodes and filling it up with a for loop
begin
# my_nodes = Vector{Node{Float64}}()
for almost_node in almost_nodes
    new_node = Node(string(almost_node[2][1]), almost_node[2][2])
# adding the new nodes
push!(my_nodes, new_node)
end
end
```

```
# Constructing my_edges of type Edge from the given file
# starting with an empty array for edges and filling it up with a for loop
begin

# my_edges = Vector{Edge{Float64, Node{Float64}}}()
for almost_edge in almost_edges
    new_edge = Edge(my_nodes[almost_edge[1]], my_nodes[almost_edge[2]],
    almost_edge[3])
# adding the new edges
    push!(my_edges, new_edge)
end
end
```