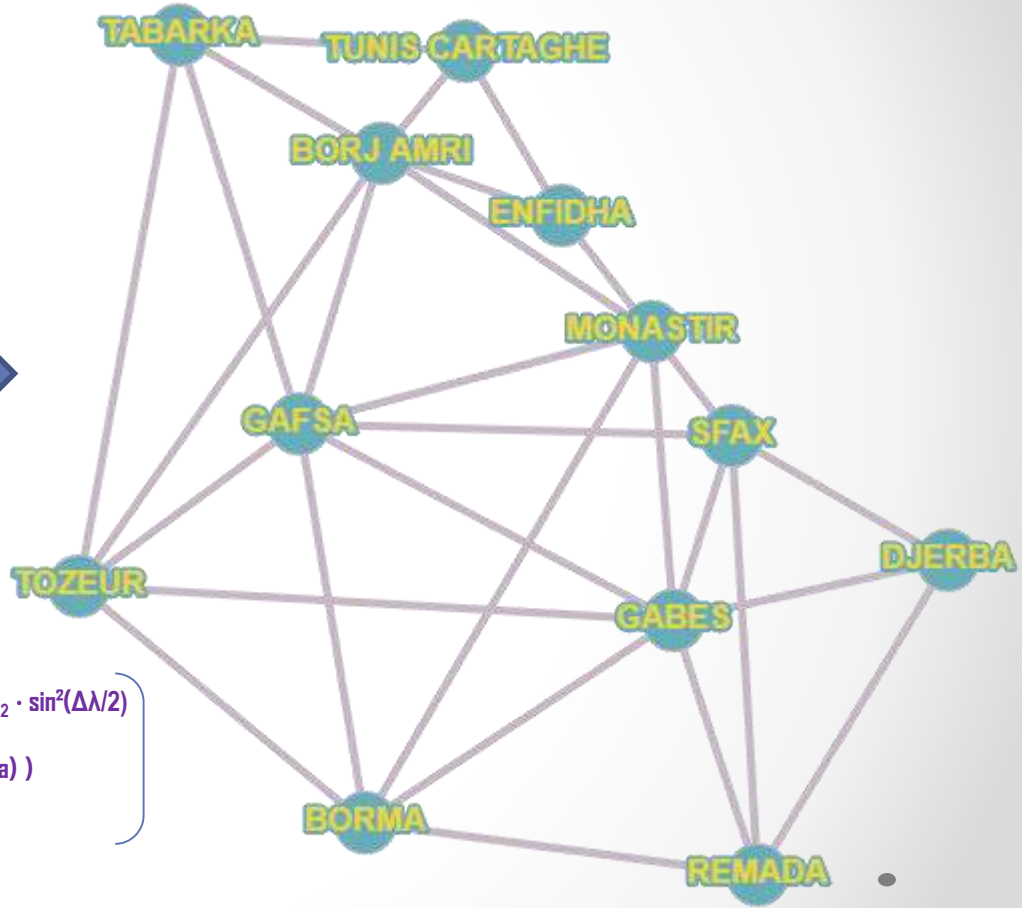
Optimisation des trajectoires dans un réseau de transport aérien

Candidat : 44175
Filière : MP Option : sciences industrielles

sommaire

Modélisation du réseau par un graphe



$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$\text{distance} = R \cdot c$$

Algorithme de Dijkstra

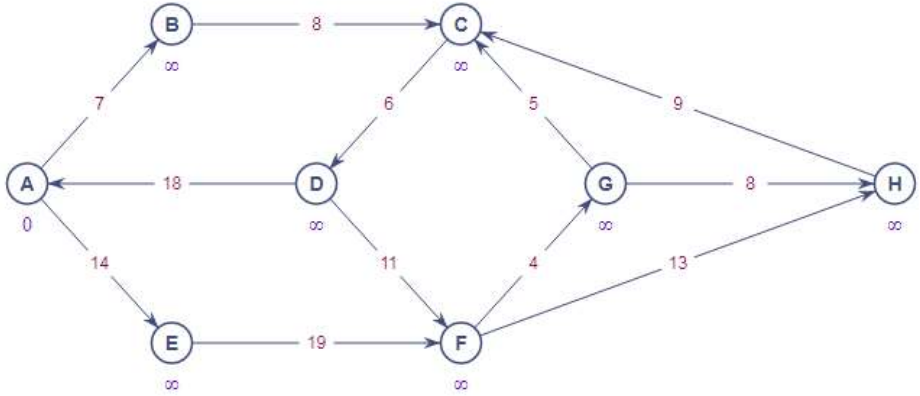
- Entrées : G=(S,A) un graphe avec une pondération positive **poids** des arcs, **deb** un sommet de **S**
- **P:== ø**
- **d[a]:== ∞**
- **d[deb]:==0**

- Tant qu'il existe un sommet hors de **P**
- Choisir un sommet **a** hors de **P** de plus petite distance **d [a]**
- Mettre **a** dans **P**
- Pour chaque sommet **b** hors de **P** voisin de **a**
- Fin Pour
- Fin Tant Que

Etape 0

On affecte le poids 0 au sommet origine (**A**) et on attribue provisoirement un poids ∞ aux autres sommets.
Le sommet **A** de poids 0 est sélectionné .

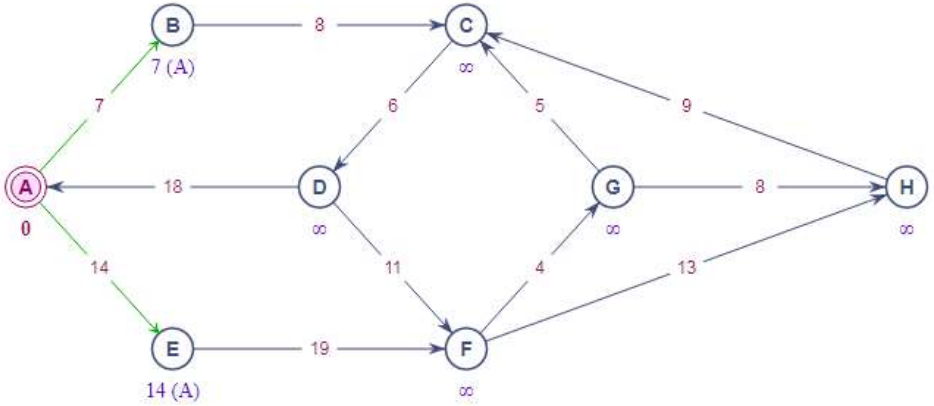
× 10



A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)

1ère étape :

Le sommet **A** de poids 0 ayant été sélectionné, on marque provisoirement les sommets B (0+7<∞) et E (0+14<∞) adjacents à **A**.
On sélectionne le sommet **B** de poids minimal. Le prédécesseur de **B** est **A**.

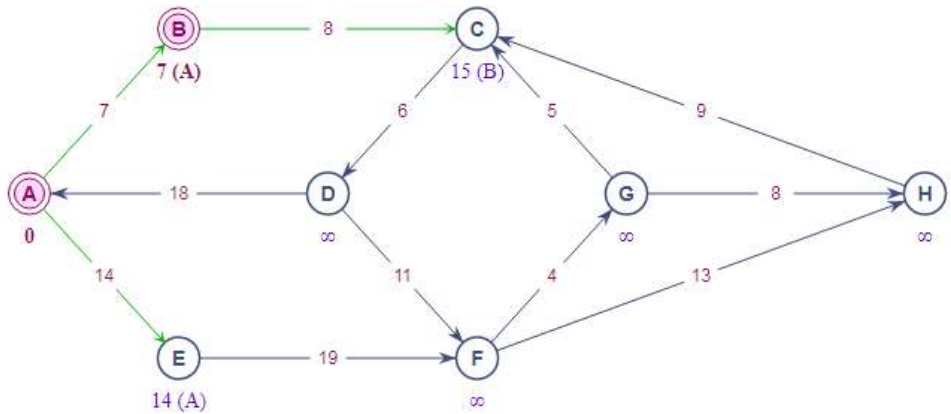


× 10

A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)

2ème étape:

B de poids 7 ayant été sélectionné, on marque provisoirement le sommet **C** (7+8<∞) adjacent à **B**. On sélectionne le sommet **E** de poids minimal. Le prédécesseur de **E** est **A**.

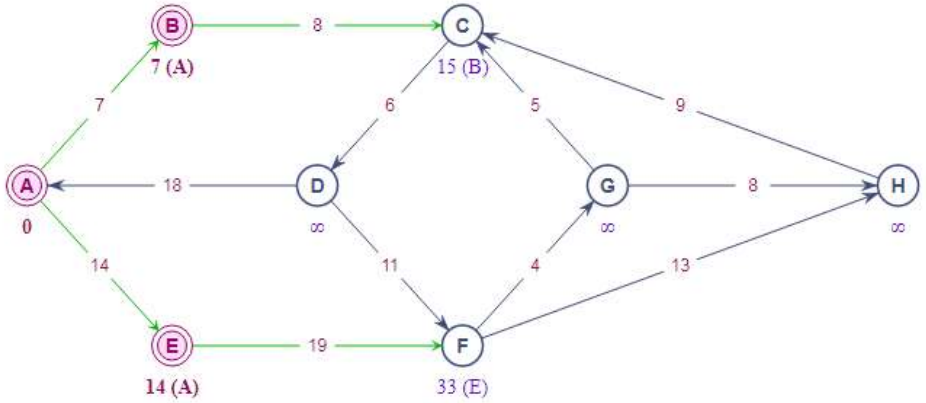


× 10

A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)
		15(B)	∞	14 (A)	∞	∞	∞	E (14)

3ème étape:

E de poids 14 ayant été sélectionné, on marque provisoirement le sommet F (14+19<∞) adjacent à **E**.
On sélectionne le sommet **C** de poids minimal. Le prédécesseur de **C** est **B**.

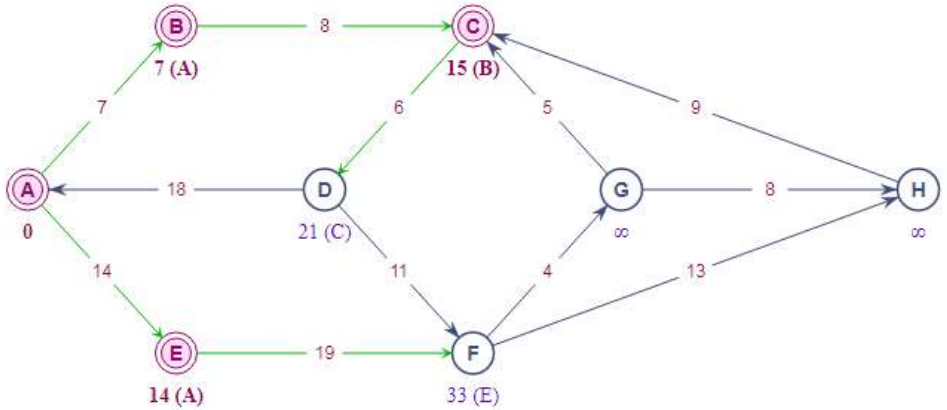


× 10

A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)
		15(B)	∞	14 (A)	∞	∞	∞	E (14)
		15 (B)	∞		33(E)	∞	∞	C (15)

4ème étape:

C de poids 15 ayant été sélectionné, on marque provisoirement le sommet D (15+6<∞) adjacent à **C**.
On sélectionne le sommet **D** de poids minimal. Le prédécesseur de **D**est **C**.

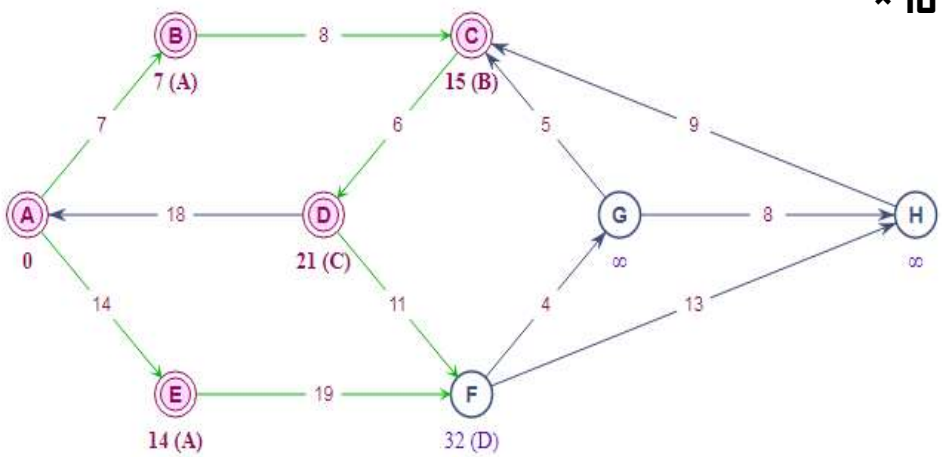


× 10

A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)
		15(B)	∞	14 (A)	∞	∞	∞	E (14)
		15 (B)	∞		33(E)	∞	∞	C (15)
			21 (C)		33 (E)	∞	∞	D(21)

6ème étape:

D de poids 21 ayant été sélectionné, on marque provisoirement le sommet F (21+11<33) adjacent à **D**.
On sélectionne le sommet **F** de poids minimal. Le prédécesseur de **F** est **D**.

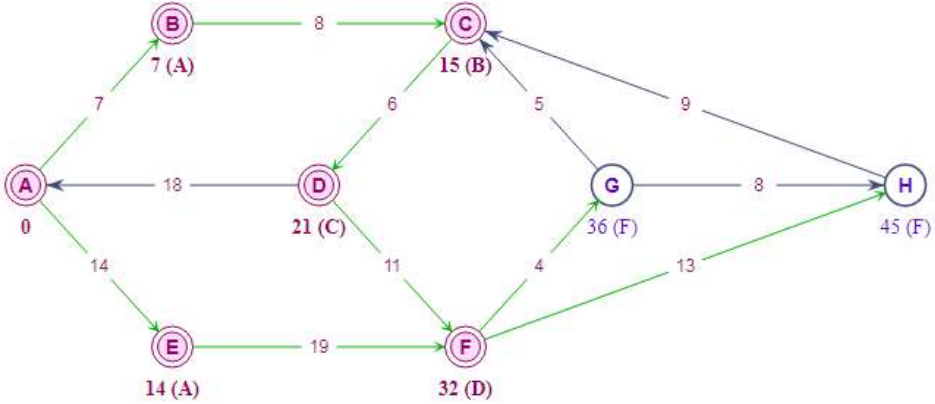


× 10

A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)
		15(B)	∞	14 (A)	∞	∞	∞	E (14)
		15 (B)	∞		33(E)	∞	∞	C (15)
			21 (C)		33 (E)	∞	∞	D(21)
					32 (D)	∞	∞	F(32)

7ème étape:

F de poids 32 ayant été sélectionné, on marque provisoirement les sommets **G** (32+4<∞) et E (32+13<∞) adjacent à **F**.
On sélectionne le sommet **G** de poids minimal. Le prédécesseur de **G** est **F**

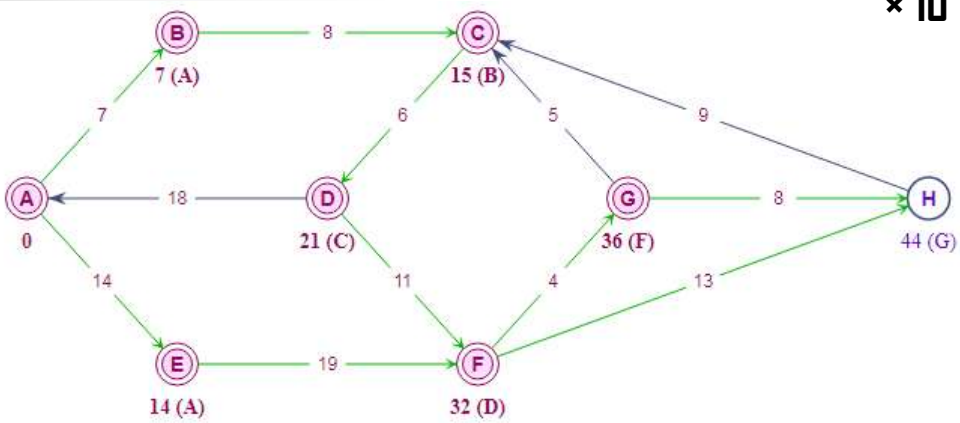


× 10

A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)
		15(B)	∞	14 (A)	∞	∞	∞	E (14)
		15 (B)	∞		33(E)	∞	∞	C (15)
			21 (C)		33 (E)	∞	∞	D(21)
					32 (D)	∞	∞	F(32)
						36 (F)	45(F)	G(36)

8ème étape:

G de poids 36 ayant été sélectionné, on marque provisoirement le sommet **H** (36+8<45).
On sélectionne le sommet **H** de poids minimal. Le prédécesseur de **H** est **G**.



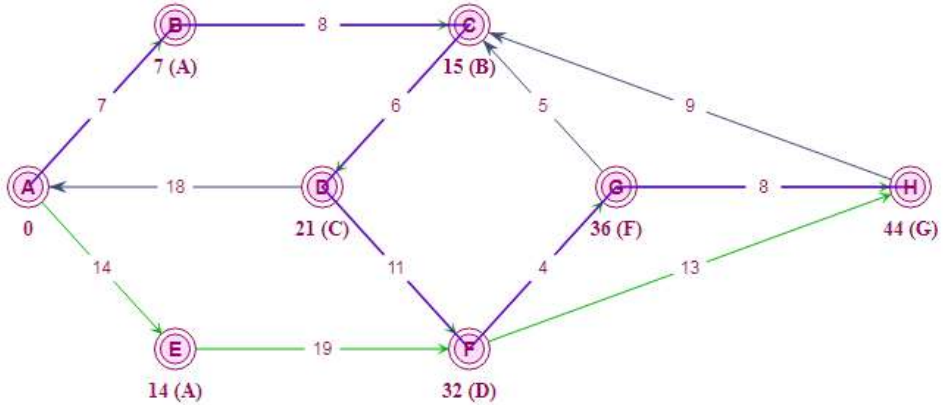
× 10

A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)
		15(B)	∞	14 (A)	∞	∞	∞	E (14)
		15 (B)	∞		33(E)	∞	∞	C (15)
			21 (C)		33 (E)	∞	∞	D(21)
					32 (D)	∞	∞	F(32)
						36 (F)	45(F)	G(36)
							44 (G)	H(44)

Fin

Sommet final H marqué : le chemin le plus court s'obtient en remontant la chaîne des prédécesseurs.

× 10



A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	A (0)
	7 (A)	∞	∞	14 (A)	∞	∞	∞	B (7)
		15(B)	∞	14 (A)	∞	∞	∞	E (14)
		15 (B)	∞		33(E)	∞	∞	C (15)
			21 (C)		33 (E)	∞	∞	D(21)
					32 (D)	∞	∞	F(32)
						36 (F)	45(F)	G(36)
							44 (G)	H(44)

Principe de : A*

BUT: recherche de plus court chemin entre deux points **S** et **G**

Cout au point n:

f(n) = g(n) + h(n)

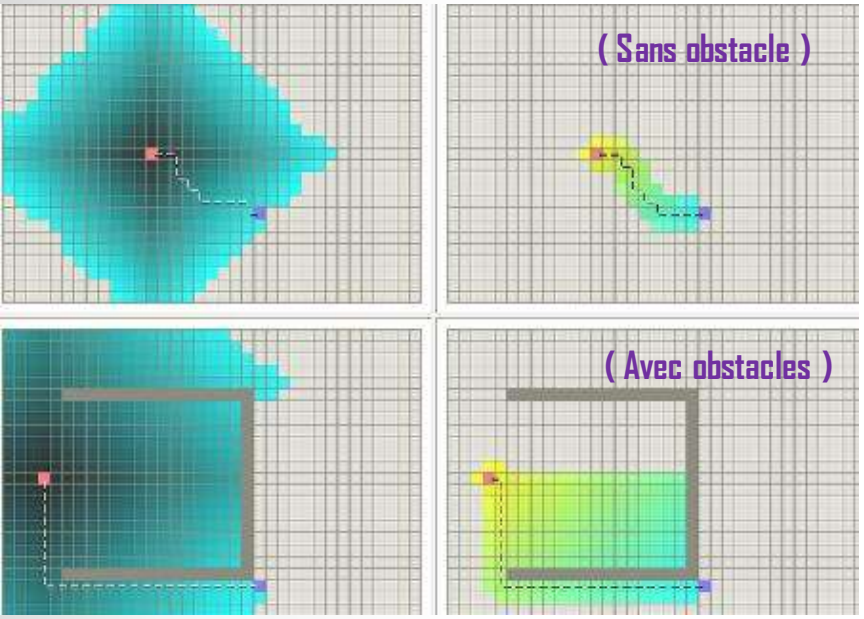
g(n) : distance entre le point de départ **S** et le point n

h(n) distance estimée entre le point n et le point **G**

f(n): cout total pour le point n

h : évaluation heuristique sur chaque nœud pour estimer le meilleur chemin y passant : exemple (distance à vol de oiseau pour un réseau de transport routier)

Dijkstra vs A*



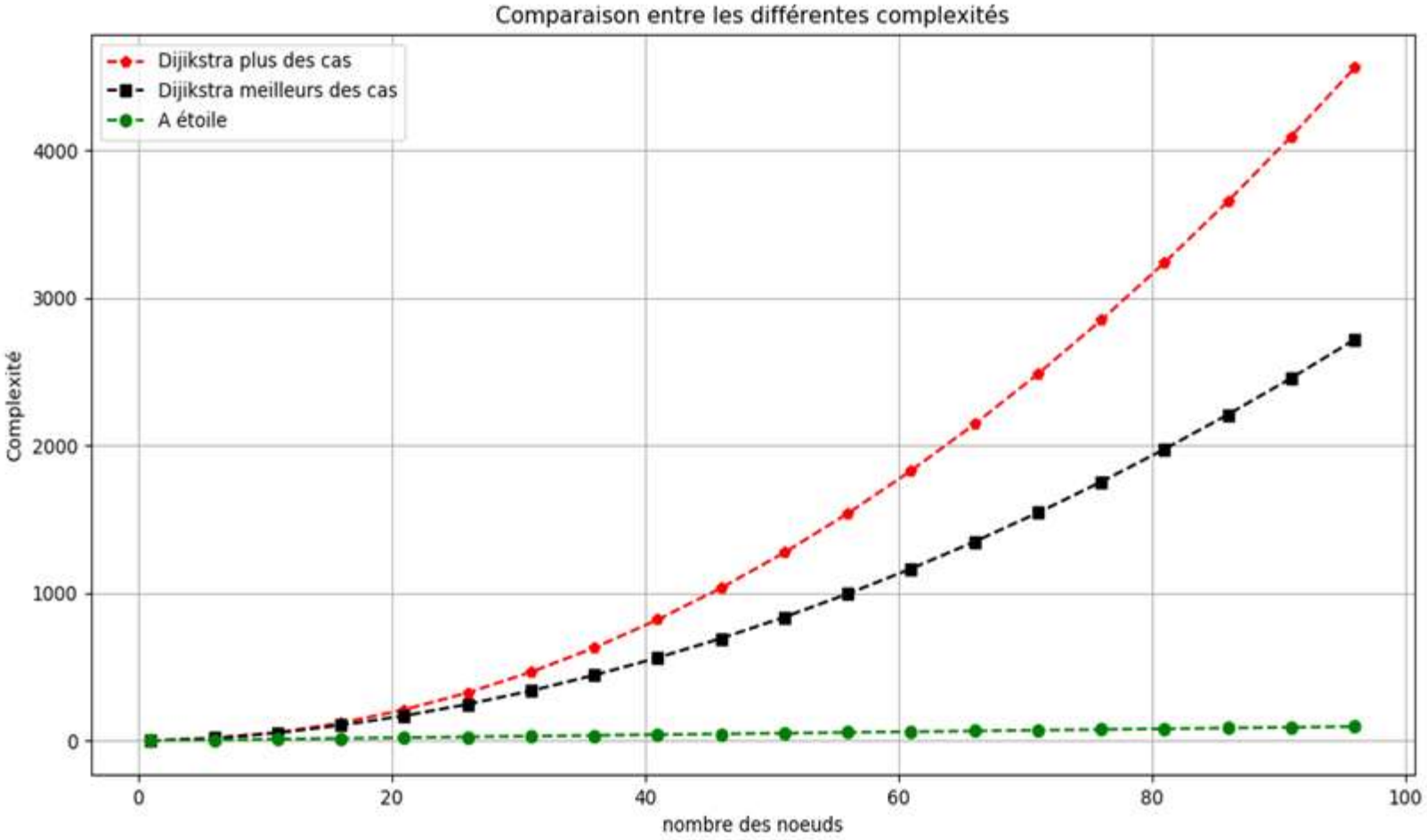
Dijkstra A*

Dijkstra :

- doit absorber tout l'espace de recherche et essayer toutes les solutions possibles
- trouve toujours la meilleure solution.

A* :

- Simple , rapide et donne un bon résultat
- utilise une heuristique d'estimation qui permet d'éliminer les chemins ayant un coût élevé ⇒ ce qui réduit le processus de recherche

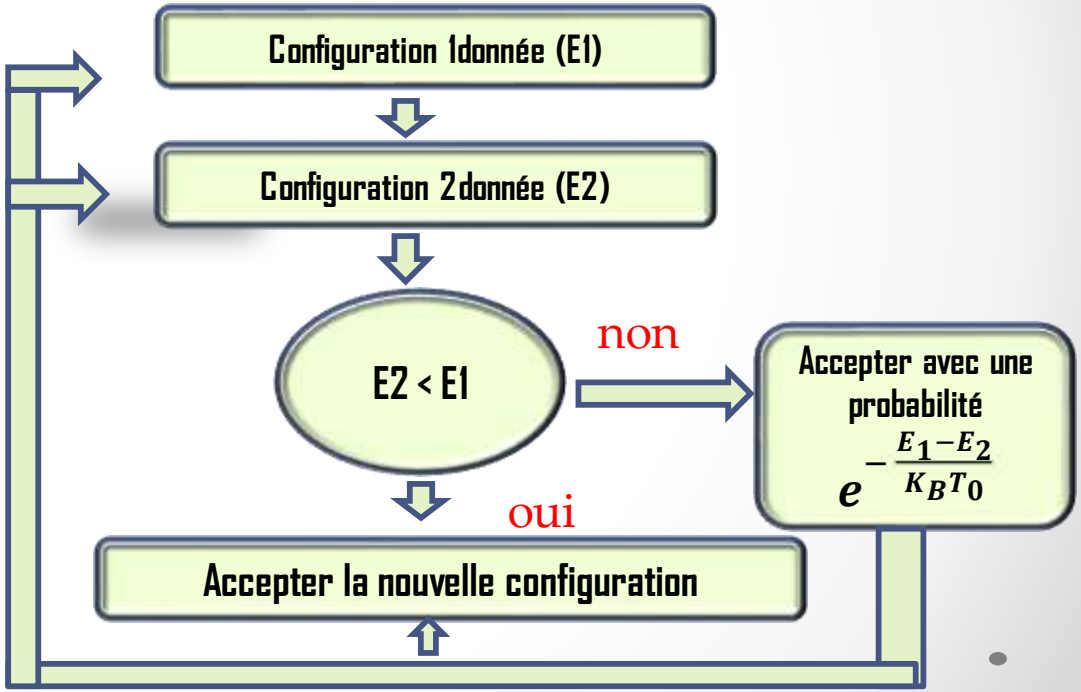

Comparaison entre les différentes complexités
Dijikstra plus des cas
Dijikstra meilleurs des cas
A étoile
Complexité
nombre des noeuds
0
1000
2000
3000
4000
0
20
40
60
80
100

Introduction au recuit simulé

Loi de distribution de Boltzman

$$P(z) = Ae^{\frac{-E_P}{K_B T_0}}$$

Metropolis a établi un algorithme qui simule l'évolution d'un système physique vers son équilibre thermodynamique à une température donnée



Algorithme de récuit simulé

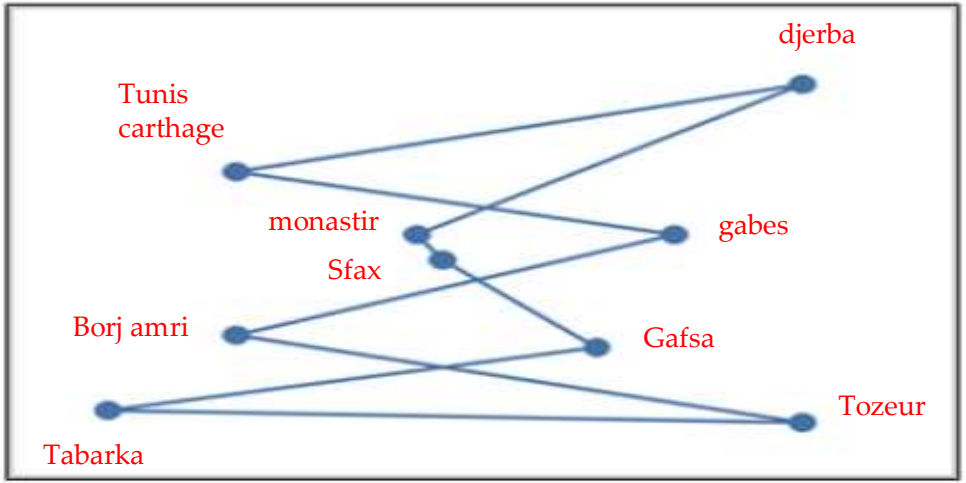
soit X_i un état d'énergie E_i, dans notre système
je fais varier X_i en lui effectuant une perturbation,
j'obtiens X_{i+1} d'énergie E_{i+1}
si $E_{i+1} \leq E_i$ alors X_{i+1} est le nouvel état de mon système, parce que son énergie diminue
sinon, je ne me bloque pas pour ne pas rester dans un minimum local éventuel, je décide que X_{i+1} devient mon nouvel état avec une probabilité égale à $e^{-\frac{E_{i+1}-E_i}{T_0}}$

on choisit généralement une loi de décroissance exponentionnelle du genre : $T=T_0 e^{-\frac{t}{\tau}}$

l'algorithme du recuit simulé appliqué à l'optimisation d'une fonction $F(X_1, X_2, \ldots, X_n)$, en utilisant l'algorithme de Metropolis pour le traitement des fluctuations

Définir la fonction f, la valeur initiale des variables xi et la valeur initiale T_0 de la température
Tant que $T_i < T_f$
    Atteindre l'équilibre du système à T_i et Calculer l'énergie moyenne du système pour T_i .
    Traiter l'écart d'énergie $E_i - E_{i-1}$ selon l'algorithme de Metropolis
    Sinon, on diminue la température selon la formule choisie.

Djerba	monastir	gabès	Borj amri	Tozeur	Tabarka	Gafsa	Sfax	Tunis	Djerba

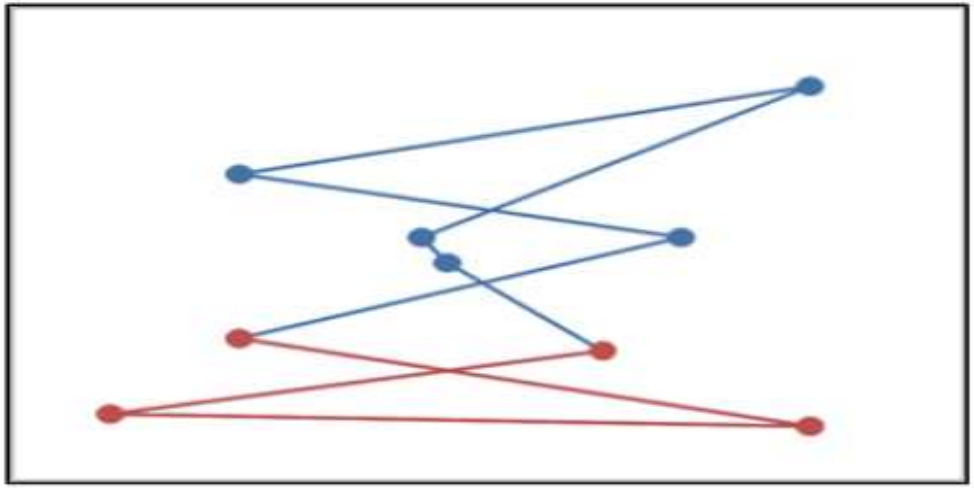


Etat initial aléatoire

- Coût initial

Utilisation d un générateur de nombres aléatoires selon la loi uniforme discrète de support N

Djerba	Tunis	Gabès	Borj Amri	Tozeur	Tabarka	Gafsa	Sfax	Monastir	Djerba

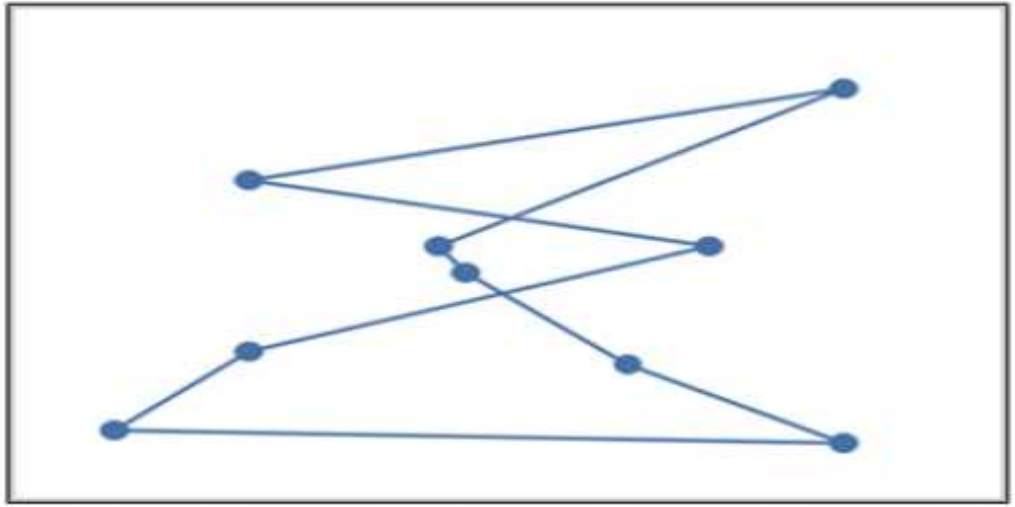


1ère étape

- On sélectionne une transformation.

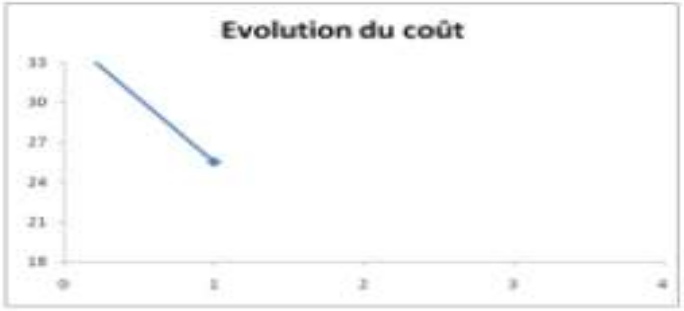
Perturbation aléatoire permettant de modifier le coût

djerba	monastir	gabes	Borj amri	tozeur	tabarka	gafsa	sfax	tunis	djerba

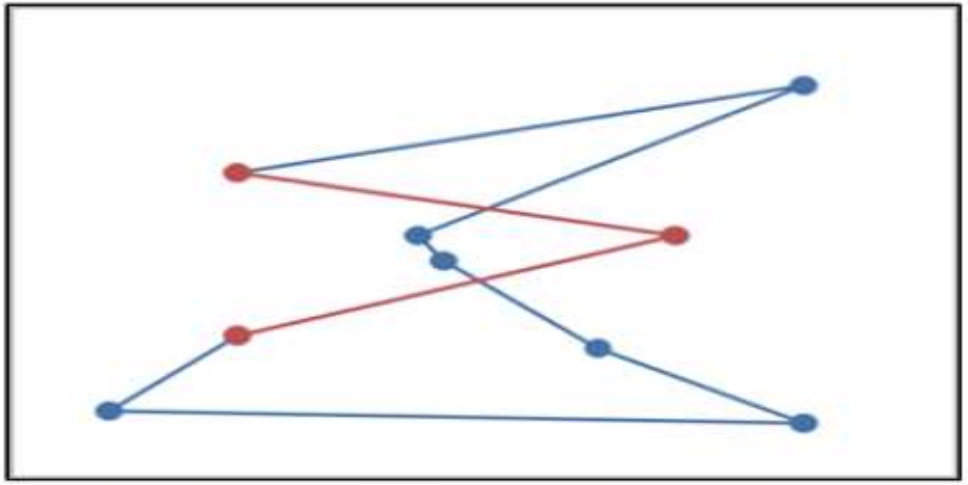


Nouvel état obtenu

- Coût à l'étape 1



Calcul du coût et comparaison avec le coût initial

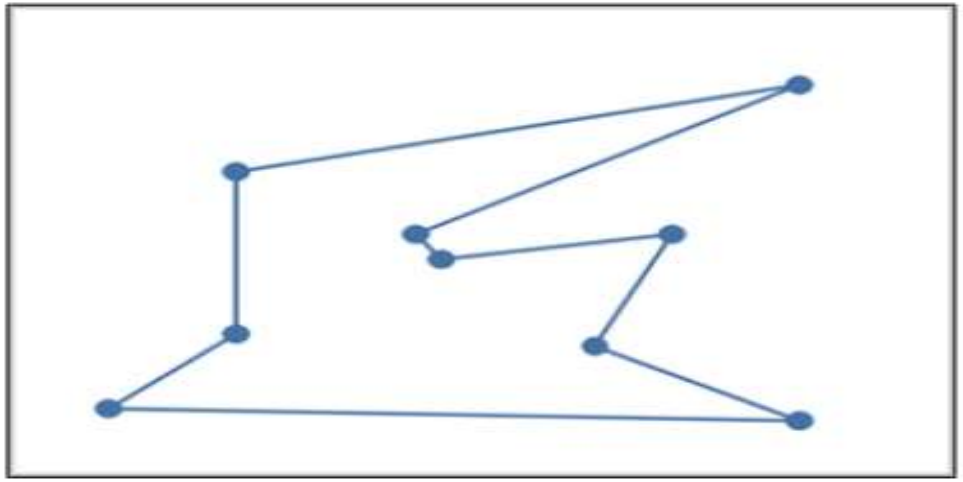


2ème étape

- On sélectionne une autre transformation.

Perturbation aléatoire
permettant de modifier le coût

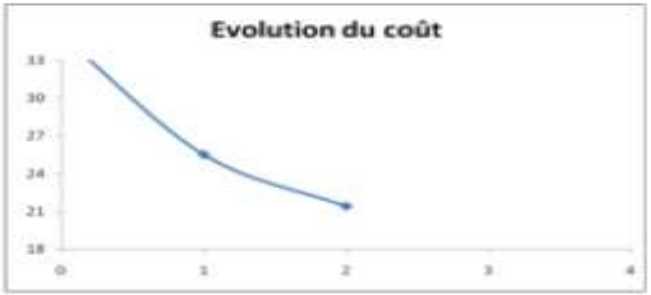
djerba	monastir	gabes	tunis	tozeur	tabarka	gafsa	sfax	Borj amri	djerba



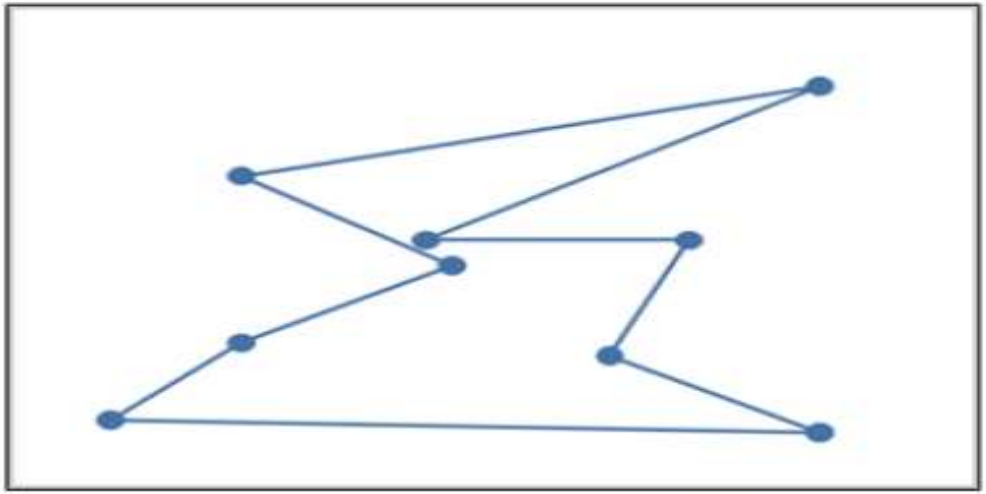
Nouvel état obtenue

- Coût à l'étape 2

Calcul du coût et comparaison avec le coût initial



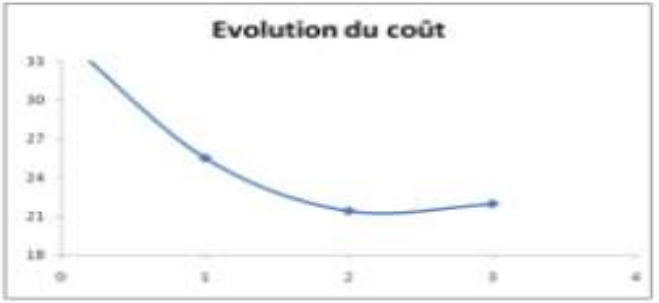
djerba	tunis	sfax	Borj amri	tabarka	Tozeur	gafsa	gabes	monastir	djerba



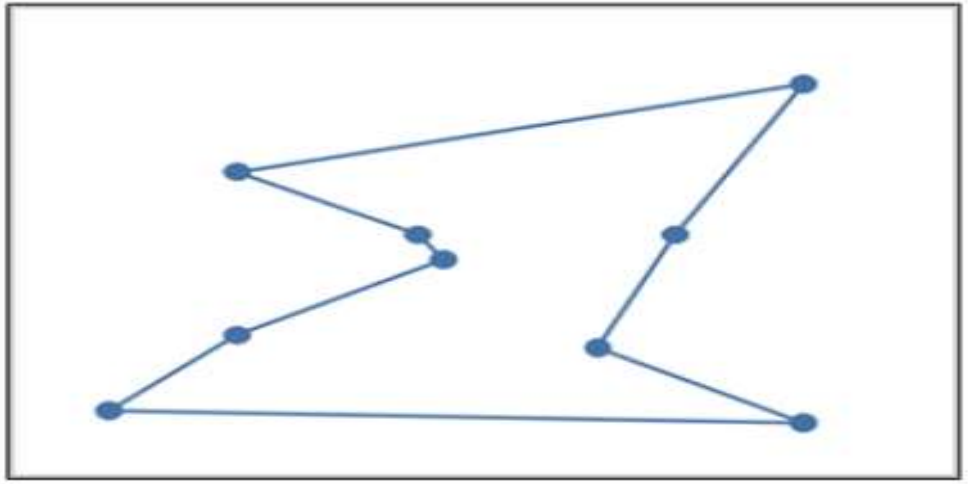
Nouvel état obtenue

- Coût à l'étape 3

Accepter avec une probabilité $e^{-\frac{E_1-E_2}{K_B T_0}}$

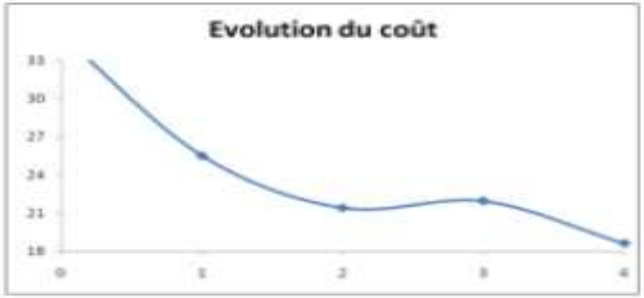


djerba	tunis	monastir	sfax	Borj amri	Tabarka	tozeur	gafsa	gabes	Djerba



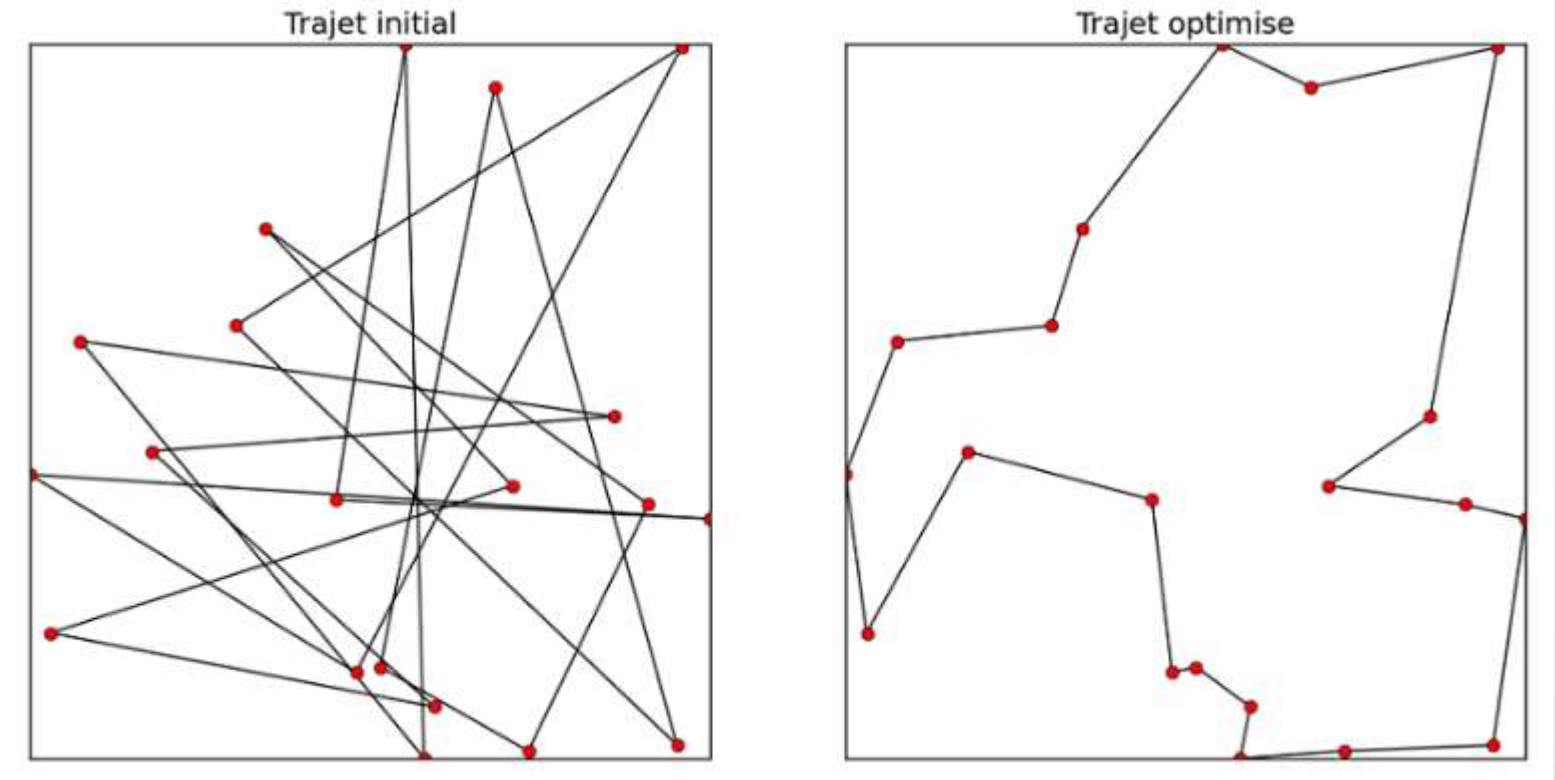
Nouvel état obtenue

- Coût à l'étape 4

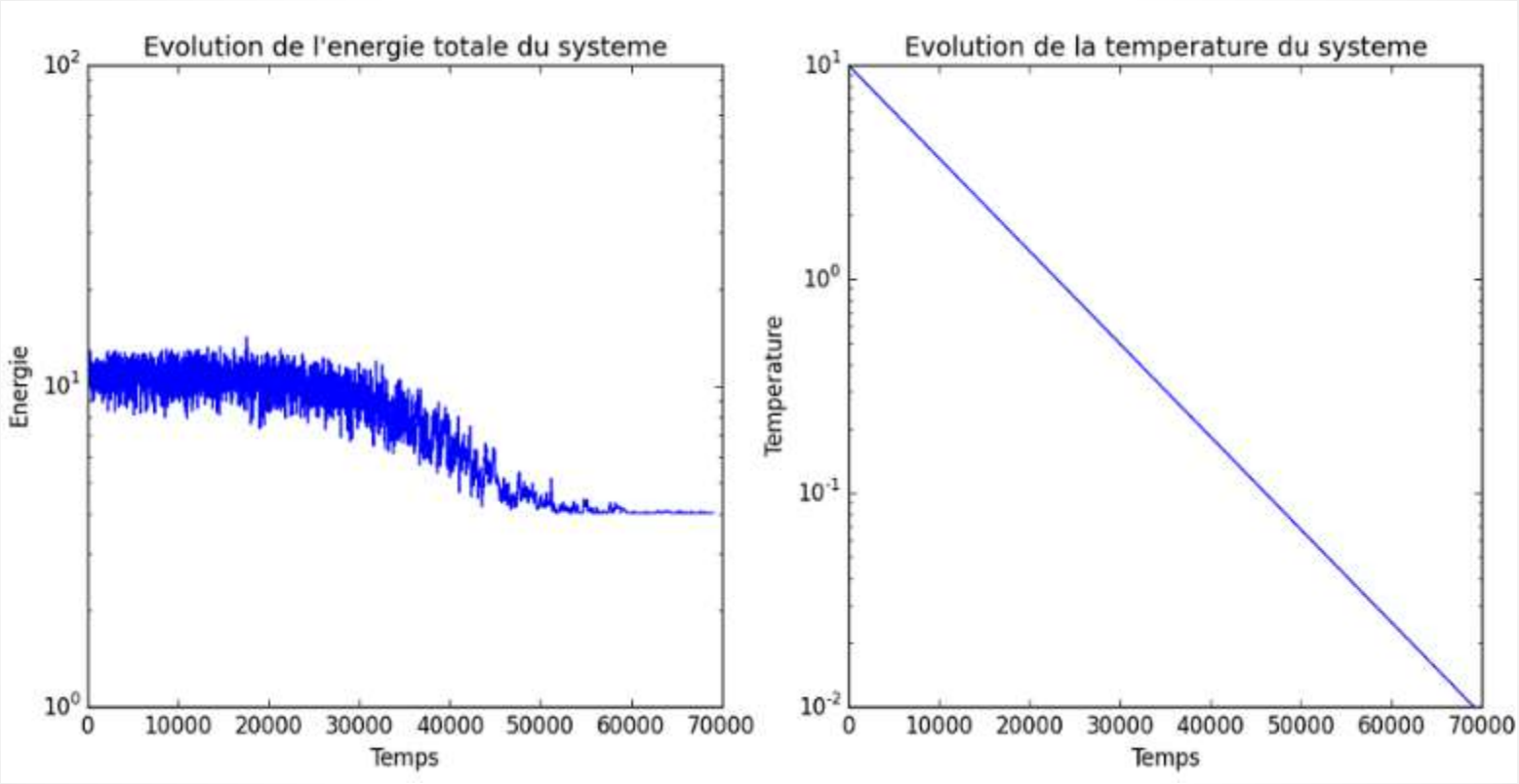


CONFIGURATION FINALE

J'ai essayé avec N, le nombre d' aéroports qui est égal à 20. C'est un nombre assez limité et le temps de calcul est court (quelques secondes sur mon pc). Le temps de calcul restera assez court pour N inférieur à la centaine. Après, cela devient un peu pénible, sur un processeur i5.

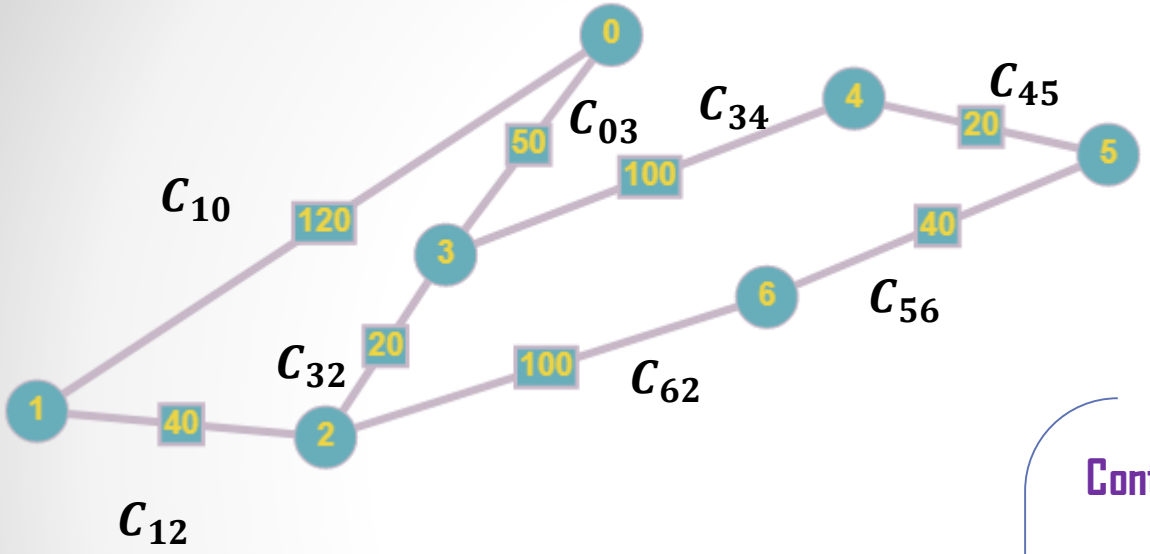


La figure suivante montre l'évolution de l'énergie du système (à gauche) et de sa température (à droite) en fonction des itérations de l'algorithme de recuit simulé



Les axes des ordonnées sont gradués en semi-log, ce qui explique la droite d'évolution de la température.

Modélisation d'un problème de recherche de plus court chemin



Optimisation de la fonction objective F sous des contrainte

$$\text{Min} : F(x_{01}, \ldots\ldots, x_{56}) = \sum c_{ij}\, x_{ij}$$

Cont 1 : $$\sum_{X\, j \in B} x_{ij} \leq a_i$$

Cont 2 : $$\sum_{X\, j \in A} x_{ij} \geq b_j$$

Cont 3 : $$x_{ij} \in \{1,0\} \quad \forall i,j$$

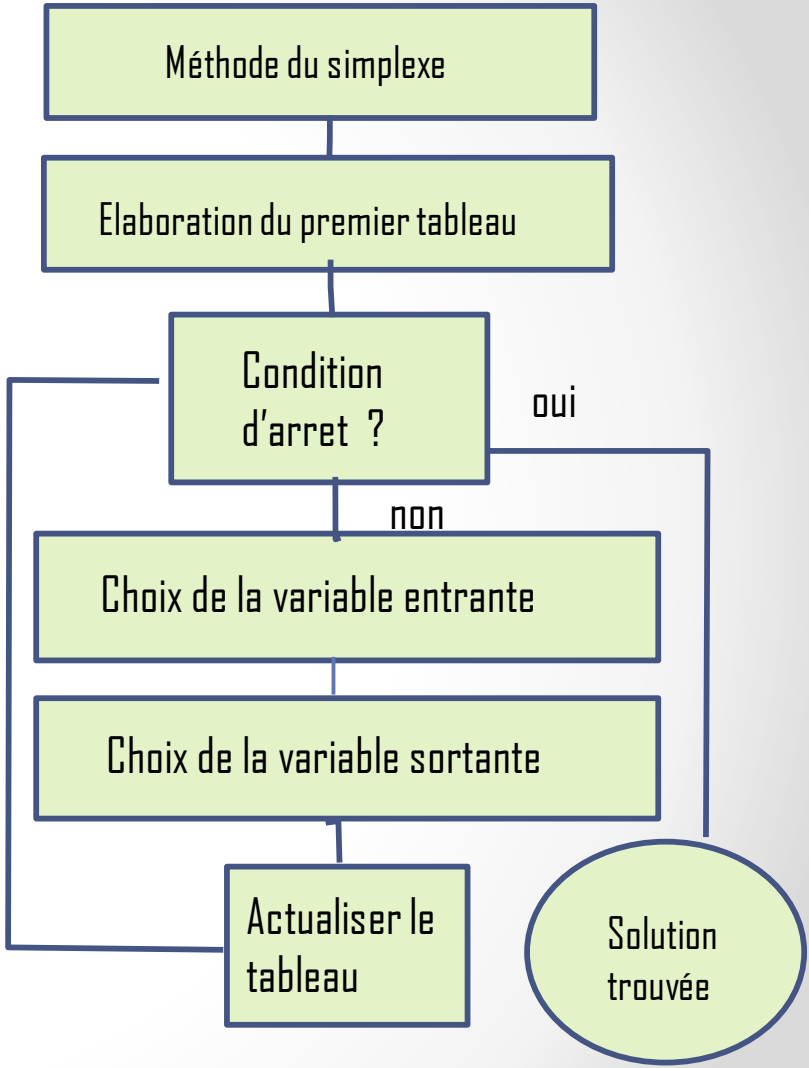
Algorithme de simplexe

La méthode du simplexe est une procédure itérative permettant d'effectuer une exploration dirigée de l'ensemble des solutions réalisables de base.

L'application de la méthode nécessite la connaissance d'une solution réalisable de base, au départ.

La méthode consiste à calculer à chaque itération un programme (une solution réalisable) «voisin» de celui qui vient d'être calculé et «au moins aussi bon» que celui-ci.

On peut aussi s'assurer, moyennant certaines précautions, que la même base ne puisse jamais apparaître dans deux itérations distinctes, ce qui suffit à assurer la convergence du procédé.



Bilan des trajets de la ville 0 XAB + XAC = 1

Bilan des trajets de la ville 1 XBD + XBE - XAB - XDB - XEB = 0

Bilan des trajets de la ville 2 XCD + XCF - XAC - XDC - XFC = 0

Bilan des trajets de la ville 3: XDB + XDC + XDE - XBD - XCD - XED = 0

Bilan des trajets de la ville 4 XEB + XED + XEG - XBE - XDE = 0

Bilan des trajets de la ville 5 XFC + XFG - XCF = 0

Bilan des trajets de la ville 6 - XEG - XFG = -1

$X_{ij} \geq 0$

X_{ij} sont booléens

Ajout des variables d'écart

1 X1 + 1 X2 + 1 X21 = 1

1 X1 -1 X3 -1 X4 + 1 X5 + 1 X6 + 1 X20 = 0

0 X1 + 1 X2 -1 X7 -1 X8 + 1 X9 + 1 X10 + 1 X19 = 0

0 X1 + 1 X4 -1 X5 + 1 X7 -1 X10 -1 X11 + 1 X12 + 1 X18 = 0

0 X1 + 1 X3 -1 X6 + 1 X11 -1 X12 -1 X13 + 1 X17 = 0

0 X1 + 1 X8 -1 X9 -1 X14 + 1 X16 = 0

0 X1 + 1 X13 + 1 X14 + 1 X15 = 1

1

Fonction objective :

$$Min \sum c_{ij}\, x_{ij}$$

MAXIMISER: Z = -120 X1 -40 X2 -30 X3 -50 X4 -50 X5 -30 X6 -20 X7 -100 X8 -100 X9 -20 X10 -100 X11 -100 X12 -20 X13 -40 X14 + 0 X15 + 0 X16 + 0 X17 + 0 X18 + 0 X19 + 0 X20 + 0 X21

~

MINIMISER: - Z = 120 X1 + 40 X2 + 30 X3 + 50 X4 + 50 X5 + 30 X6 + 20 X7 + 10 X8 + 100 X9 + 20 X10 + 100 X11 + 100 X12 + 20 X13 + 40 X14+ 0 X16 + 0 X17 + 0 X18 + 0 X19 + 0 X20 + 0 X21

Déterminer une solution de base admissible.

Toute solution de base de (PL=) pour laquelle toutes les variables sont non négatives, est appelée solution de base admissible. Cette solution de base admissible correspond à un point extrême.

Formuler le tableau simplexe correspondant

choix de la variable entrante (dans la base)

Maximum des $C_j - Z_j$ pour des problèmes de max.
Minimum des $C_j - Z_j$ pour des problèmes de min.

choix de la variable sortante

Dans un problème de min OU de max, la variable sortante sera le minimum des

$$\frac{b_i}{a_{ik}} \text{ tq } \quad a_{ik} > 0$$

REMARQUE :

Si dans la colonne correspondant à la variable entrante toutes les variables de base ont un coefficient négatif ou nul alors il n'existe pas de solution optimale finie (aucune contrainte ne limite l'augmentation de la valeur de la variable entrante et en conséquence la valeur de la fonction objectif n'est pas bornée)

Choix de pivot

Le coefficient du tableau simplexe situé à l'intersection de la colonne correspondant à la variable entrante et de la ligne correspondant à la variable sortante servira de pivot pour l'étape suivante.

Le pivotage

s'effectue de la manière suivante :
On commence par diviser la ligne du pivot par le chiffre du pivot
Nous devons calculer les nouvelles valeurs pour les cases restantes à partir du tableau précédent :

1

$$\text{élément initial} - \frac{\text{élement de la ligne pivot} * \text{élement de la colonne pivot}}{pivot}$$

6. Le critère d'arrêt

Nous arrêtons lorsque nous obtenons le critère d'optimalité. L'algorithme du simplexe s'arrête lorsque:

pour un problème de max $C_j - Z_j \leq 0$
pour un problème de min $C_j - Z_j \geq 0$

Coeff dans Z								
Base		X_1	X_2	E_1	E_2	E_3	E_4	b_i
COEF z	VAR BASE							
	E_1	a_{11}	a_{12}	1	0	0	0	b1
	E_2	a_{21}	a_{22}	0	1	0	0	b2
	E_3	a_{31}	a_{32}	0	0	1	0	b3
	E_4	a_{41}	a_{42}	0	0	0	1	b4
z_J								
$\mathbf{C_j - Z_j}$								

1

Conclusion

Pendant mon travail durant cette année :

Recherche d'un plus court chemin :

- Dijkstra : algorithme itératif : mauvaise complexité
- A* : algorithme basé sur une heuristique : amélioration de la complexité (intelligence artificielle)
- algorithme de récuit simulé : algorithme basé sur la physique statistique .

- algorithme de simplexe : programmation linéaire

➡ **Approche multiparadigme et multidisciplinaire**

MERCI POUR VOTRE ATTENTION