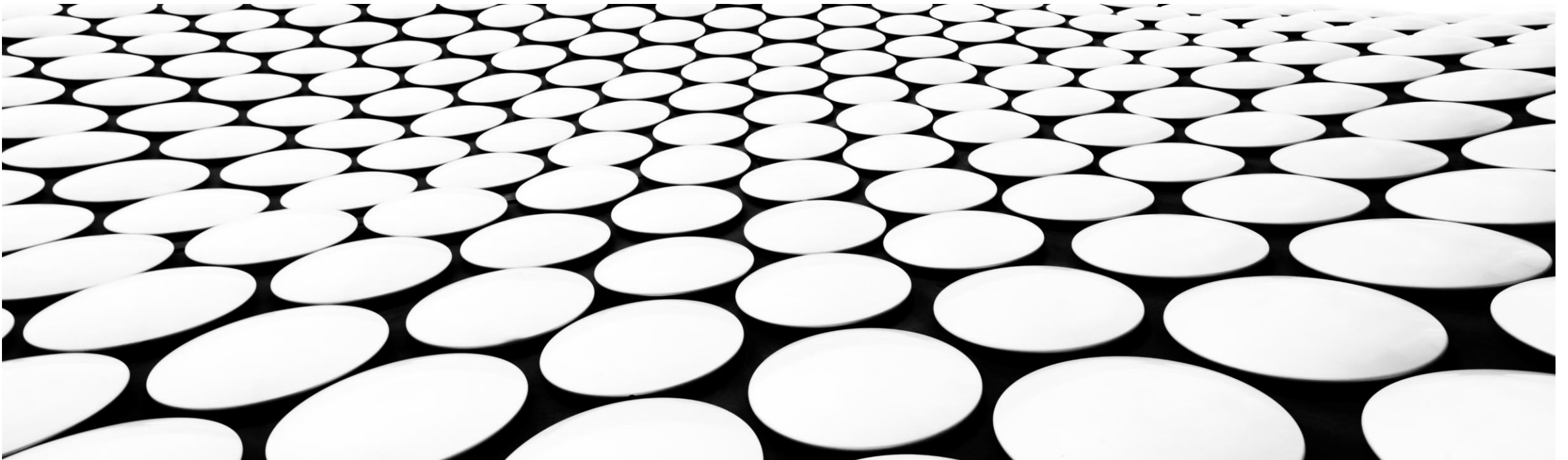


EXPLORATION DE L'ESPACE DE SOLUTIONS

SATISFAISABILITÉ ET OPTIMISATION





EXPLORATION DE L'ESPACE DE SOLUTIONS

- Partie 1 – Propagation de contraintes
- Partie 2 – Recuit Simulé
- Partie 3 – Algorithme génétique

Partie 1

Propagation de contraintes

EXEMPLE D'UN PSC

- Variables: A, B, C, D, E représentent les activités
- Domaines:
 - $D_A = \{1, 2, 3, 4\}$
 - $D_B = \{1, 2, 3, 4\}$
 - $D_C = \{1, 2, 3, 4\}$
 - $D_D = \{1, 2, 3, 4\}$
 - $D_E = \{1, 2, 3, 4\}$
- Contraintes rigides:

$(B \neq 3)$ and $(C \neq 2)$ and $(A \neq B)$ and $(B \neq C)$ and
 $(C < D)$ and $(A = D)$ and $(E < A)$ and $(E < B)$ and
 $(E < C)$ and $(E < D)$ and $(B \neq D)$
- Contraintes souples

B et C doivent être le plus petit possible

Espace de solutions (A, B, C, D, E)

$(1,1,1,1,1)$ $(1,1,3,4,1)$
 $(1,1,3,1,1)$ $(2,3,3,4,1)$
 $(2,1,3,4,4)$ $(2,1,3,4,1)$ $(4,2,3,4,1)$
 $(2,5,3,4,3)$ $(5,1,4,4,2)$
 $(5,1,3,4,1)$
 $(4,1,3,4,2)$

Trouver le(s) solutions

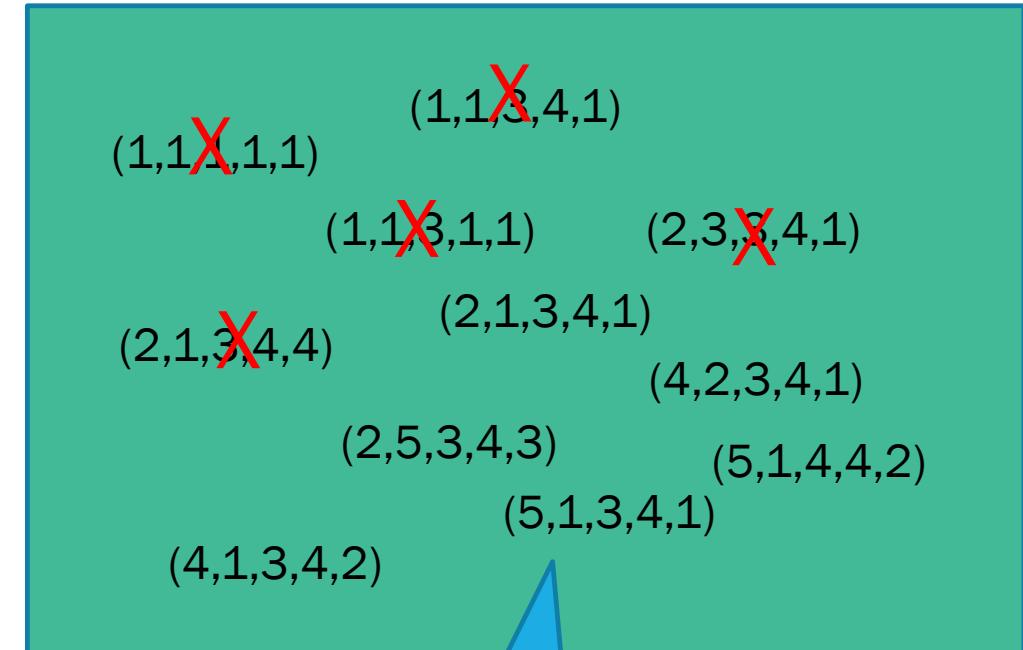
EXEMPLE D'UN PSC

- Variables: A, B, C, D, E représentent les activités
- Domaines:
 - $D_A = \{1, 2, 3, 4\}$
 - $D_B = \{1, 2, 3, 4\}$
 - $D_C = \{1, 2, 3, 4\}$
 - $D_D = \{1, 2, 3, 4\}$
 - $D_E = \{1, 2, 3, 4\}$
- Contraintes rigides:

$(B \neq 3)$ and $(C \neq 2)$ and $(A \neq B)$ and $(B \neq C)$ and
 $(C < D)$ and $(A = D)$ and $(E < A)$ and $(E < B)$ and
 $(E < C)$ and $(E < D)$ and $(B \neq D)$
- Contraintes souples

B et C doivent être le plus petit possible

Espace de solutions (A, B, C, D, E)



Pourrions-nous réduire les domaines pour ne pas explorer ces solutions?

Réduction de domaines par propagation de contraintes

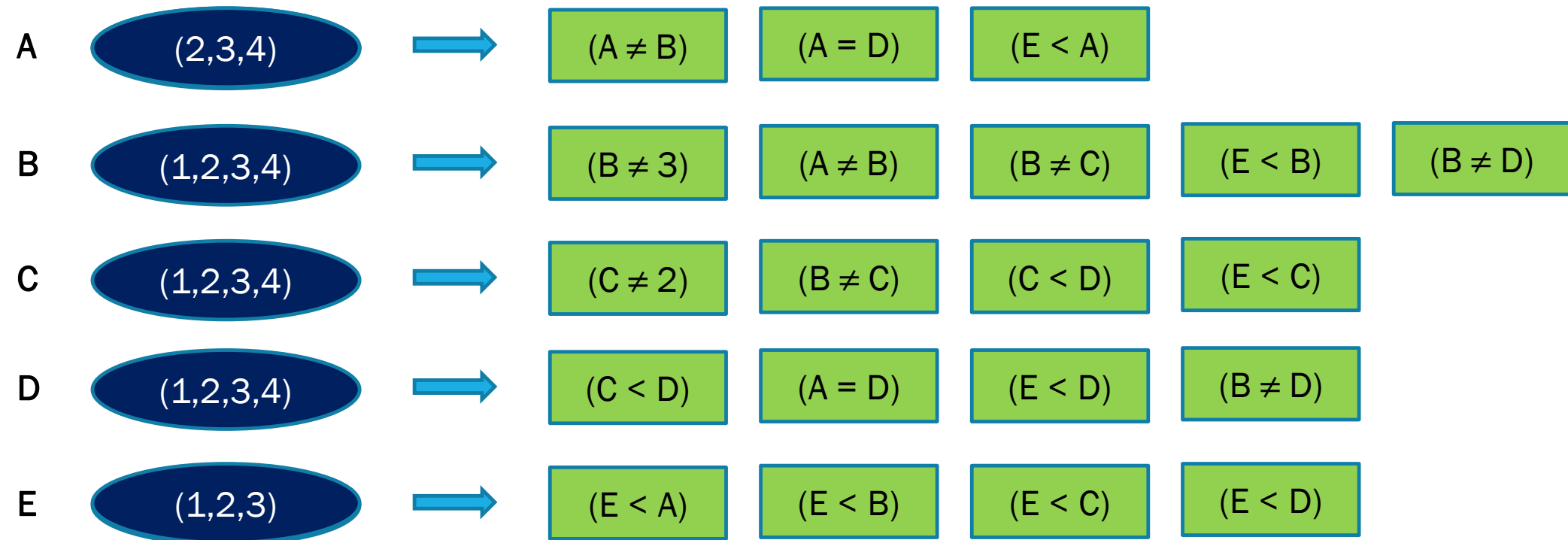
Types de contraintes

- Égalité
- Inégalité
- Ordonnancement

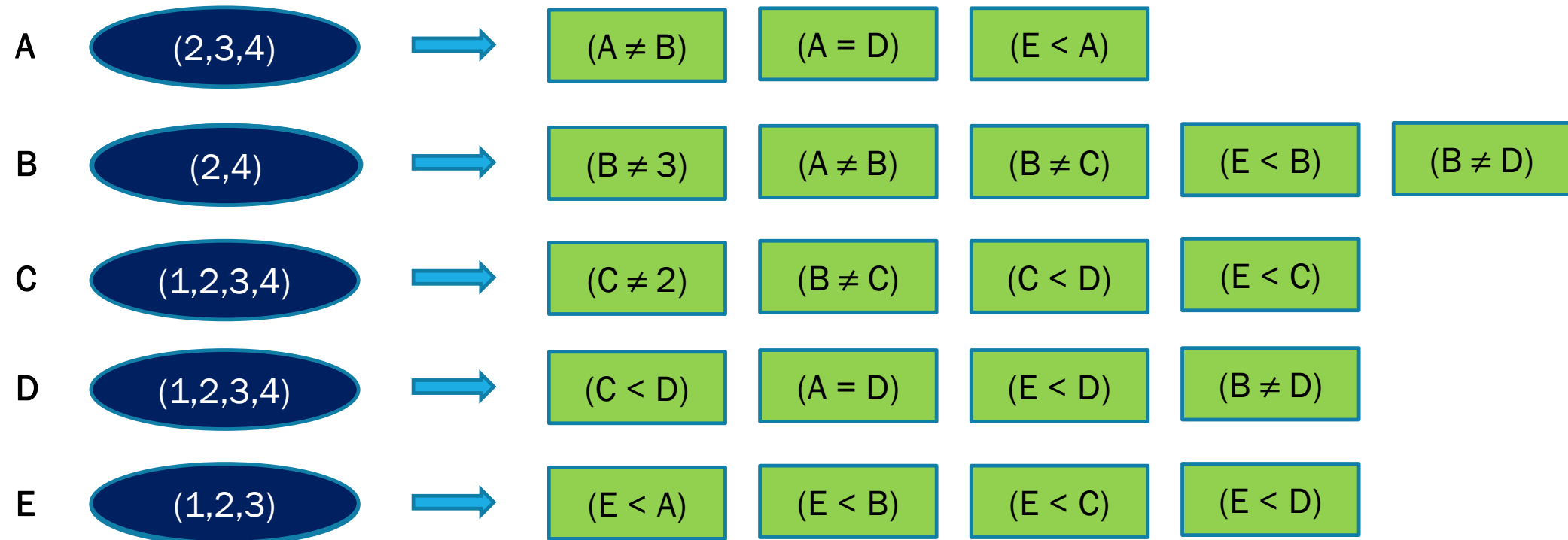
PROPAGATION DE CONTRAINTES (UN ALGORITHME POSSIBLE)

- SolutionPossible = Vrai
- Faire une liste de toutes les variables avec leurs contraintes
- Pour chaque variable, faire une liste de toutes les contraintes qui s'y appliquent
- Mettre toutes les variables dans la liste « à revoir »
- Répéter
 - Enlever la variable au haut de la liste « à revoir »
 - Pour chacune de ses contraintes
 - Supprimer des domaines impliqués toutes les valeurs IMPOSSIBLES selon la contrainte
 - Si un domaine pour une variable est modifié, ajouter la variable à la liste « à revoir » (sauf si elle est déjà présente dans la liste)
 - Si un domaine devient vide, mettre SolutionPossible à Faux
- Jusqu'à ce que la liste « à réviser » soit vide ou que SolutionPossible soit Faux

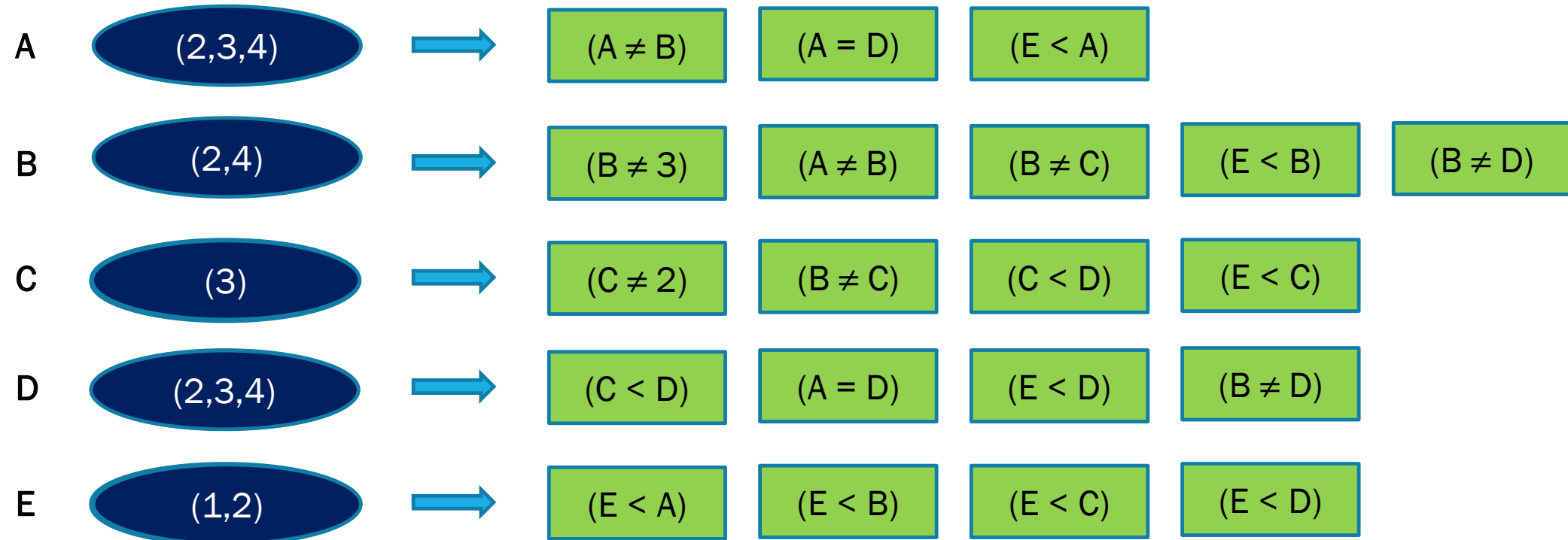
CONSTRAINT PROPAGATION



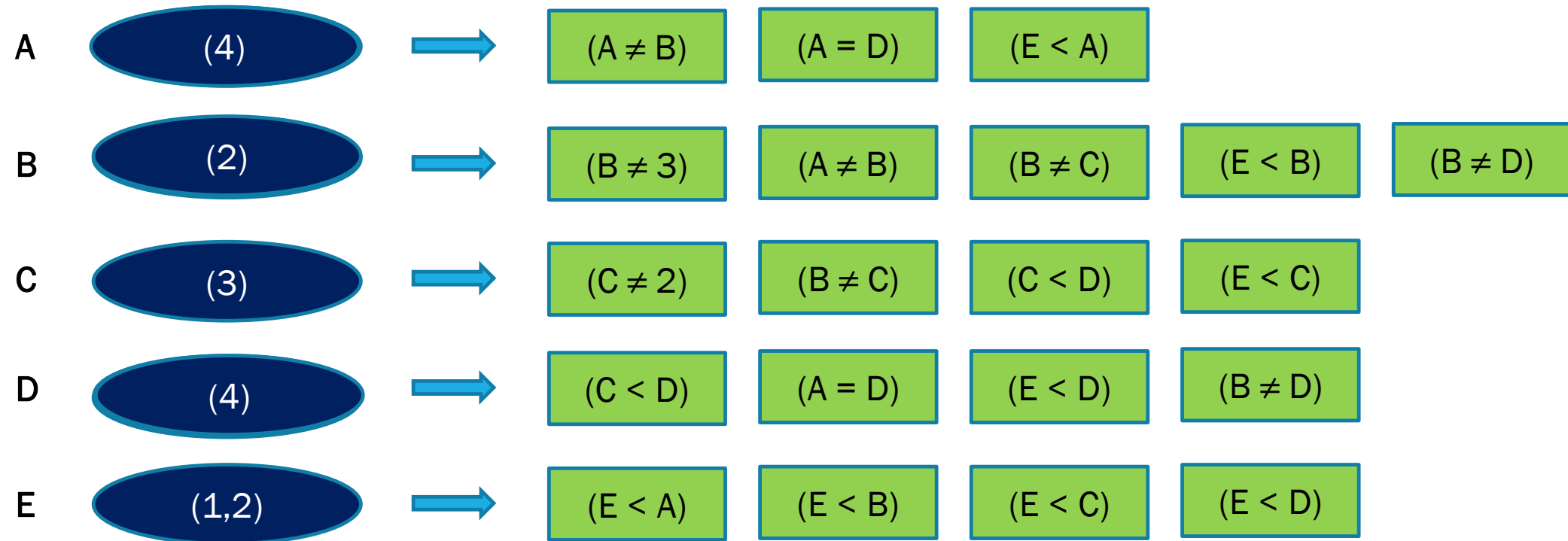
CONSTRAINT PROPAGATION



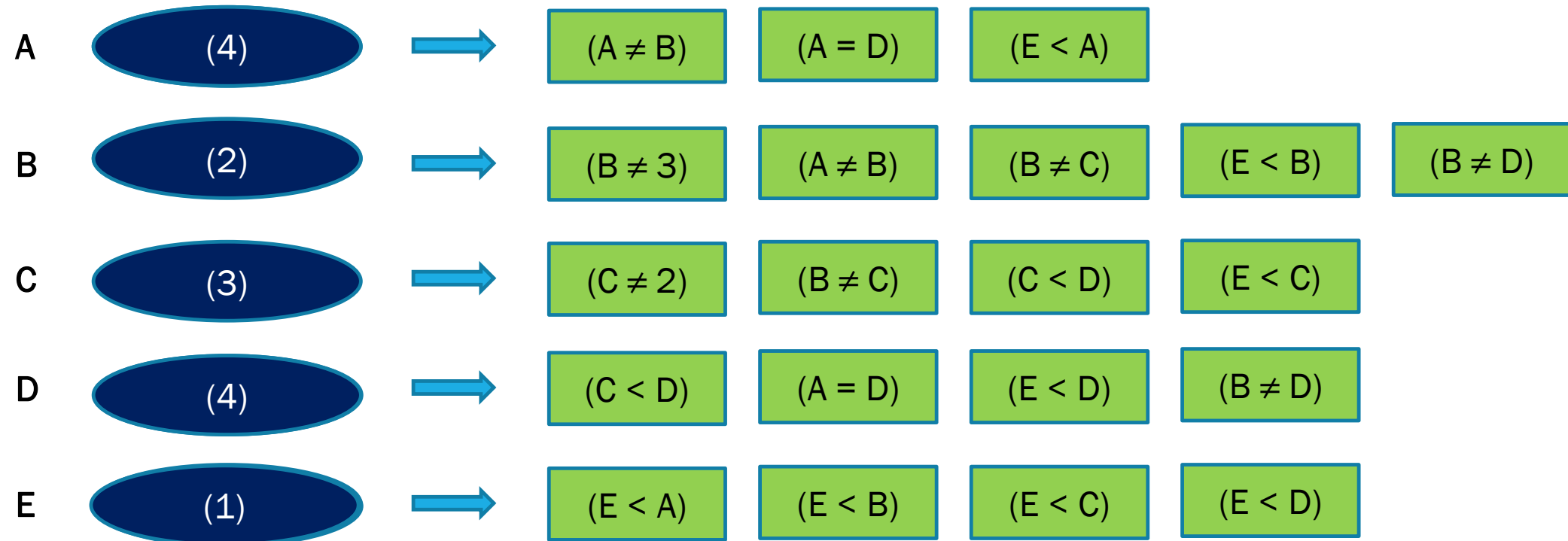
CONSTRAINT PROPAGATION



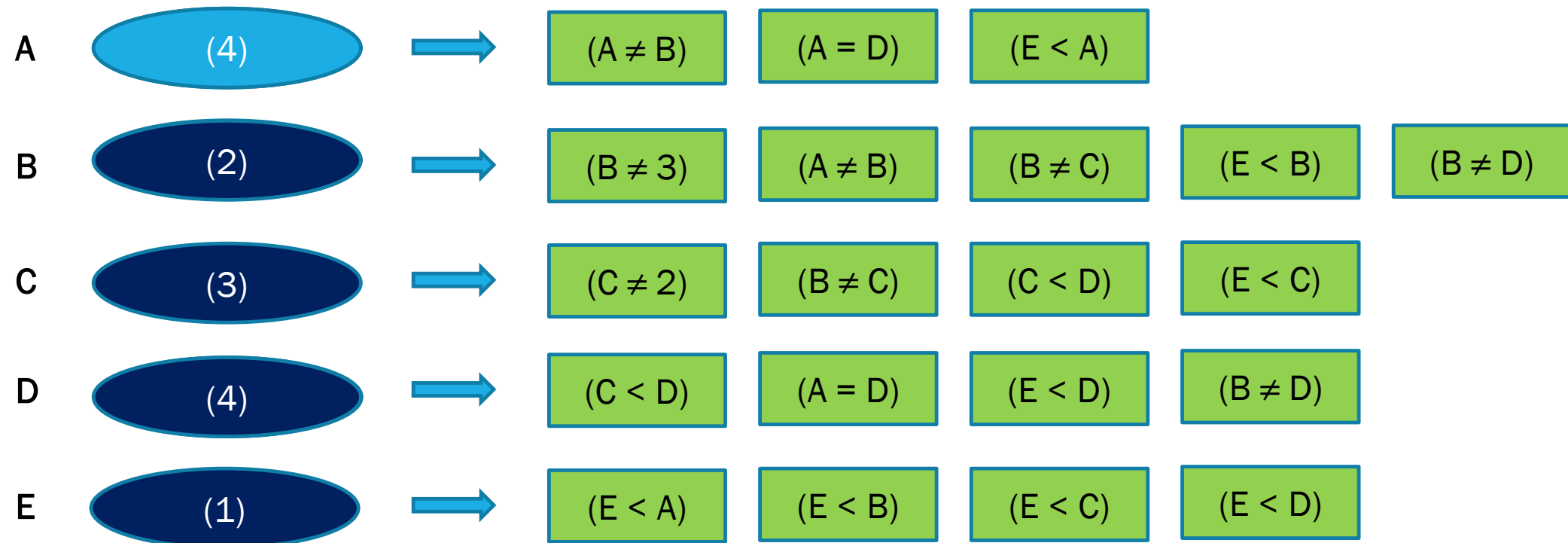
CONSTRAINT PROPAGATION



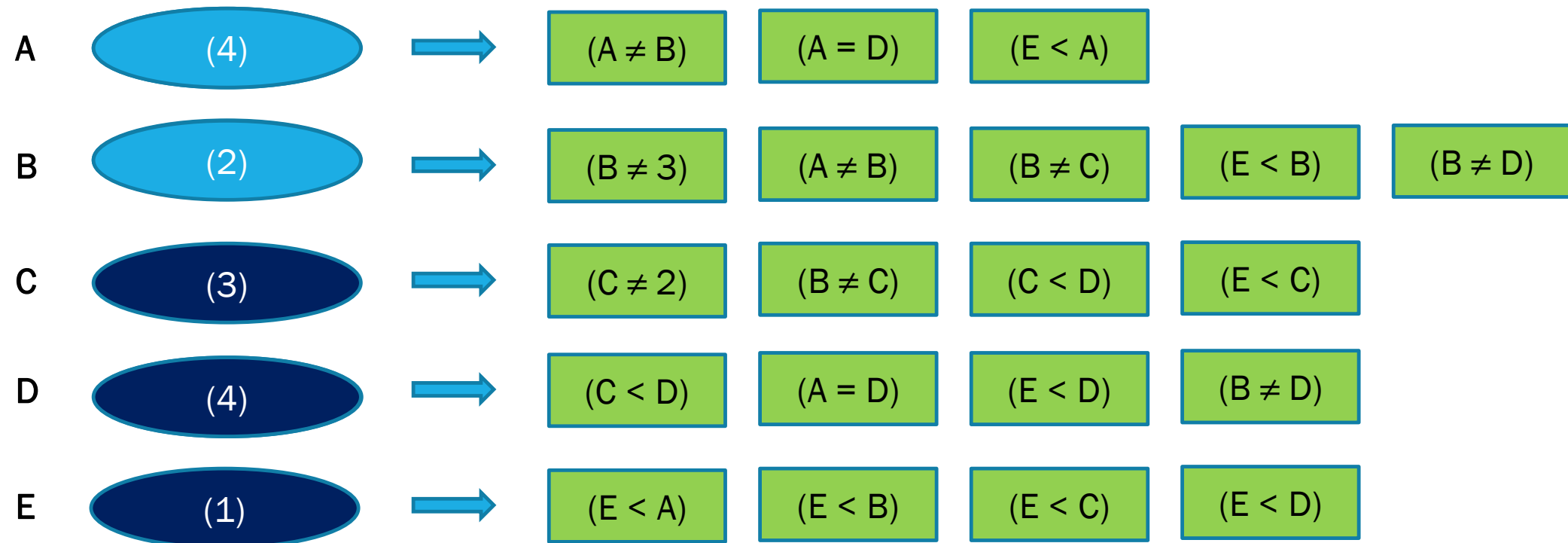
CONSTRAINT PROPAGATION



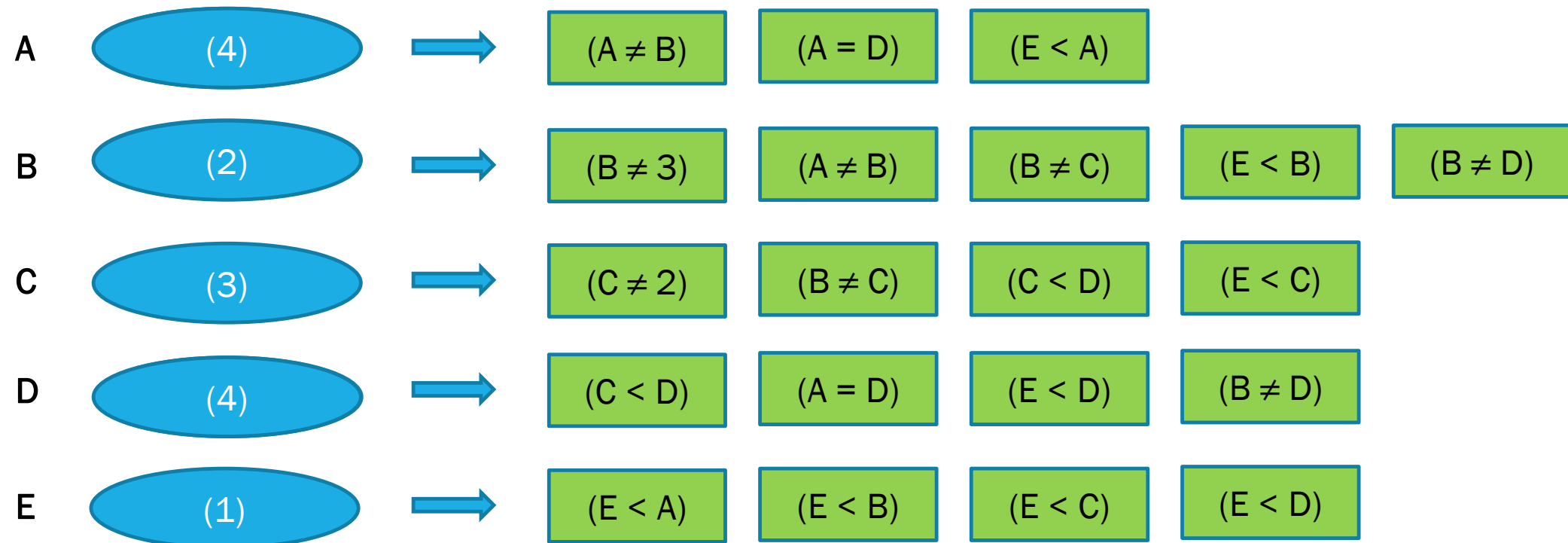
CONSTRAINT PROPAGATION



CONSTRAINT PROPAGATION



CONSTRAINT PROPAGATION



PROPAGATION DE CONTRAINTES (UN ALGORITHME POSSIBLE)

- SolutionPossible = Vrai
- Faire une liste de toutes les variables avec leurs contraintes
- Pour chaque variable, faire une liste de toutes les contraintes qui s'y appliquent
- Mettre toutes les variables dans la liste « à revoir »
- Répéter
 - Enlever la variable au haut de la liste « à revoir »
 - Pour chacune de ses contraintes
 - Supprimer des domaines impliqués toutes les valeurs IMPOSSIBLES selon la contrainte
 - Si un domaine pour une variable est modifié, ajouter la variable à la liste « à revoir » (sauf si elle est déjà présente dans la liste)
 - Si un domaine devient vide, mettre SolutionPossible à Faux
- Jusqu'à ce que la liste « à réviser » soit vide ou que SolutionPossible soit Faux

Résultat de la réduction de domaines par propagation de contraintes

- Réduction de chaque variable à une seule valeur
 - Solution unique
- Réduction d'une des variables à un domaine vide
 - Aucune solution
- Réduction des variables à un sous-ensemble de valeurs
 - Plusieurs solutions



EN RÉSUMÉ

- Algorithme de propagation de contraintes
- Exemple de propagation avec contraintes d'égalité, inégalité et ordonnancement
- Types de résultats (aucune, une seule, ou plusieurs solutions)

Partie 2

Recuit simulé

Espace de solution réduit

- Génération et test (*Generate and Test*)
 - Si l'espace de recherche est suffisamment petit, et que nous cherchons une solution optimale

Espace de recherche encore large

- Recherche gloutonne (Greedy search)
 - S'appuyer sur une stratégie basée sur la connaissance du problème
- Recherche aléatoire
 - Redémarrage aléatoire
 - Étape aléatoire
 - Modification aléatoire

Malheureusement, cela peut conduire à un maximum/minima local

En essayant encore et encore... une plus grande partie de l'espace de recherche est exploré et nous pourrions trouver les maxima/minima globaux.

MODIFICATION ALÉATOIRE

Algorithme générique:

- Débuter avec une solution initiale (gloutonne ou autre):
- Répéter pour N itérations:
 - Faire un changement local
 - Si la solution est meilleure (selon une fonction de coût):
 - Garder le changement
 - Sinon
 - Garder le changement en fonction d'une probabilité

Il existe des variations sur comment calculer/ajuster cette probabilité

On garde aussi en mémoire la meilleure solution jusqu'à date.

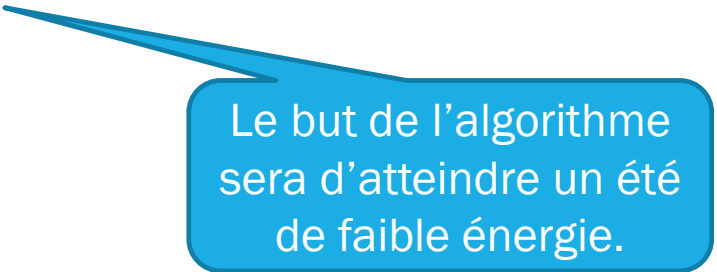
Les chercheurs, au fil des ans, ont pris leur inspiration de divers phénomènes du monde réel (physique ou social)

Simulated annealing (Recuit Simulé): un algorithme bien connu de modification aléatoire

Le recuit simulé s'inspire du processus de restructuration d'une configuration interne d'un solide qui est recuit (par exemple, processus de cristallisation).

Le solide est chauffé jusqu'au point de fusion, de sorte que ses particules sont réparties de manière aléatoire.

Le matériau est ensuite lentement refroidi, afin que les particules se réorganisent pour atteindre un état de faible énergie.



Le but de l'algorithme sera d'atteindre un état de faible énergie.

Algorithm 2: Simulated Annealing Optimizer

```
 $T \leftarrow T_{max}$   
 $\mathbf{x} \leftarrow$  generate the initial candidate solution  
 $E \leftarrow E(\mathbf{x})$  compute the energy of the initial solution  
while  $(T > T_{min})$  and  $(E > E_{th})$  do  
   $\mathbf{x}_{new} \leftarrow$  generate a new candidate solution  
   $E_{new} \leftarrow$  compute the energy of the new candidate  $\mathbf{x}_{new}$   
   $\Delta E \leftarrow E_{new} - E$   
  if Accept  $(\Delta E, T)$  then  
     $\mathbf{x} \leftarrow \mathbf{x}_{new}$   
     $E \leftarrow E_{new}$   
  end  
   $T \leftarrow \frac{T}{\alpha}$  cool the temperature  
end  
return  $\mathbf{x}$ 
```

E_{th} est l'énergie minimale désirée
(peut être 0)

Plutôt utiliser $r \cdot T$ avec r
(cooling factor) plus petit
que 1

Algorithm 1: Acceptance Function

Data: T , ΔE - the temperature and the energy variation between the new candidate solution and the current one.

Result: Boolean value that indicates if the new solution is accepted or rejected.

```
if ( $\Delta E < 0$ ) then
|   return True;
else
|    $r \leftarrow$  generate a random value in the range  $[0, 1)$ 
|   if ( $r < \exp(-\Delta E/T)$ ) then
|   |   return True
|   else
|   |   return False
|   end
end
```

Impact de la température sur la probabilité d'acceptation d'une solution

Exemple avec solution actuelle = 0,1 *fitness* (Énergie)

La nouvelle solution est 0,3 – solution moins bonne, nous déciderons si nous la gardons

En fonction de la température (T), le tableau montre la probabilité d'acceptation.

Le refroidissement (r) est appliqué à T .

```

 $r \leftarrow$  generate a random value in the range  $[0, 1)$ 
if ( $r < \exp(-\Delta E/T)$ ) then
  | return True
else
  | return False
  
```

$T, r=0,8$	$\exp(-0,2/T)$
1	0,81873075
0,8	0,77880078
0,64	0,73161563
0,512	0,67663385
0,4096	0,61368025
0,32768	0,54315988
0,262144	0,46629376
0,2097152	0,38532262
0,16777216	0,30358523
0,13421773	0,22534649

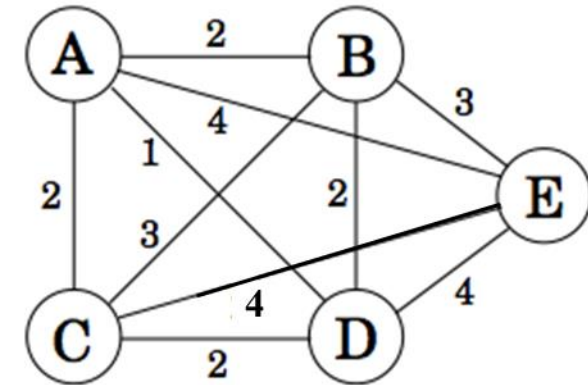
PROBLÈME DU TSP

First solution...

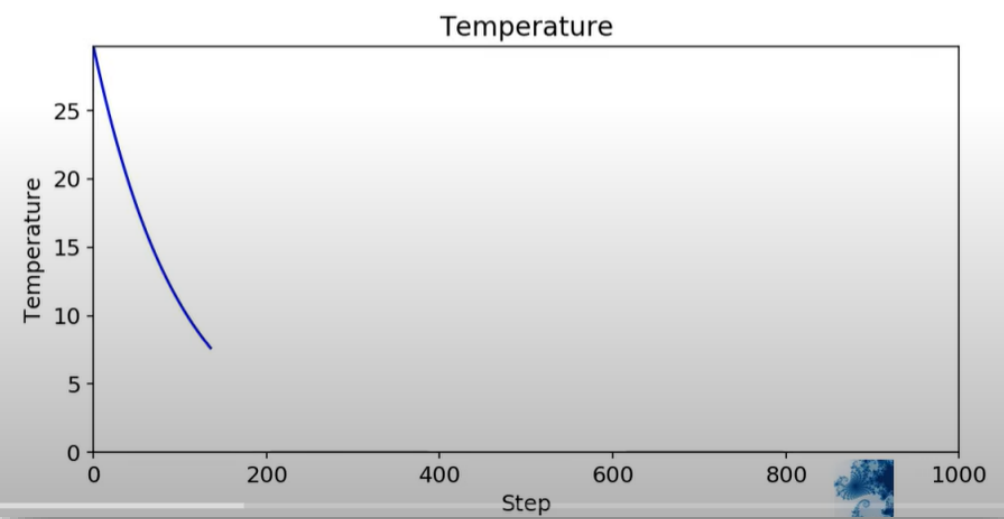
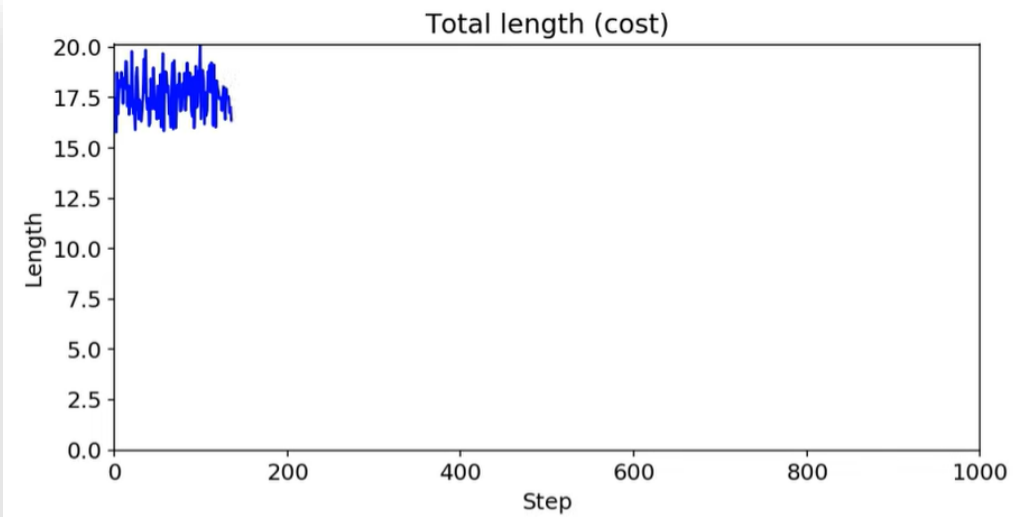
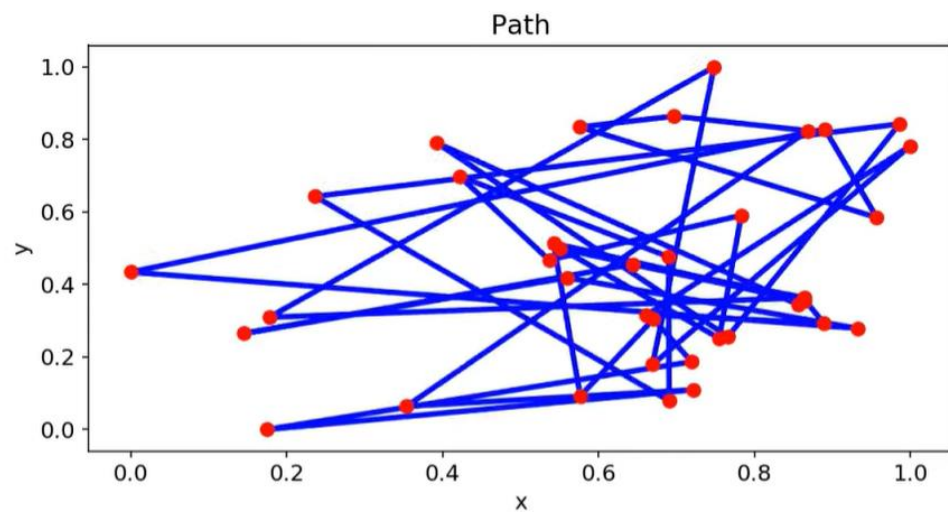
T1	T2	T3	T4	T5	retour	Coût
D	A	C	B	E	D	
	1	2	3	3	4	13

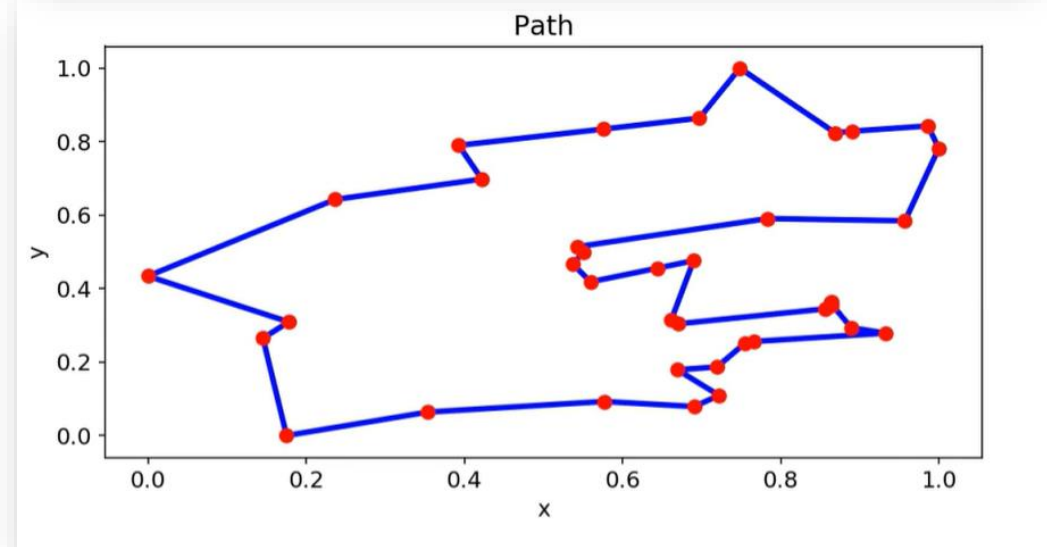
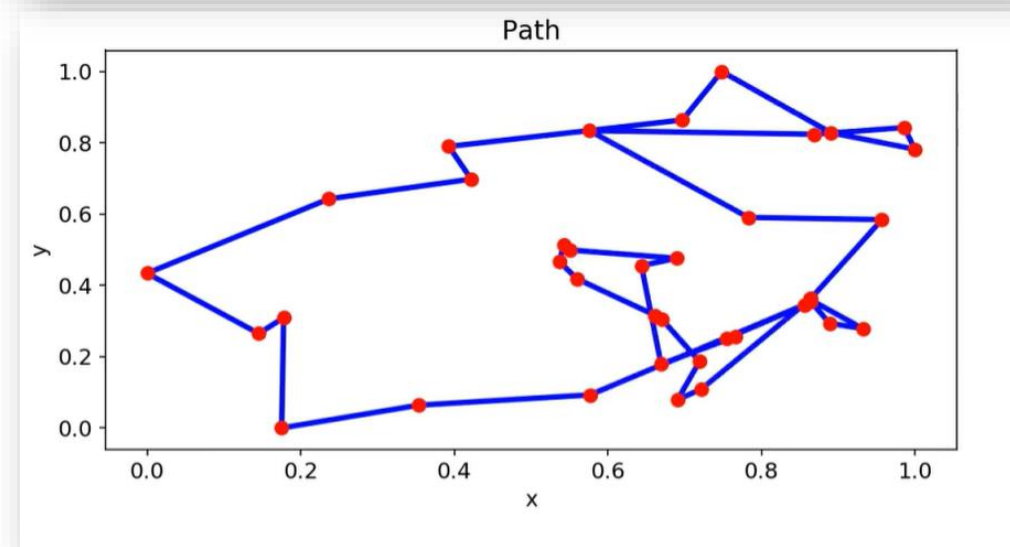
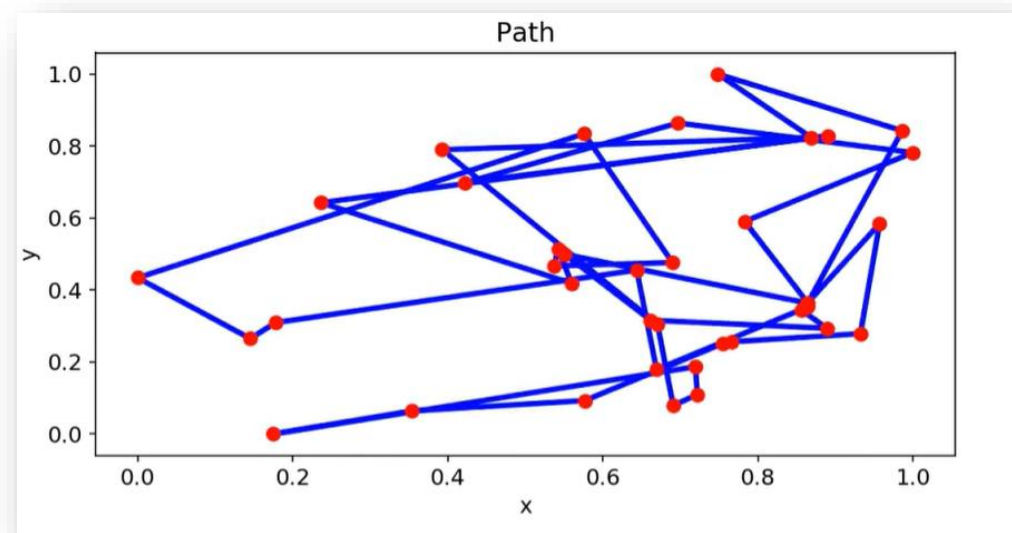
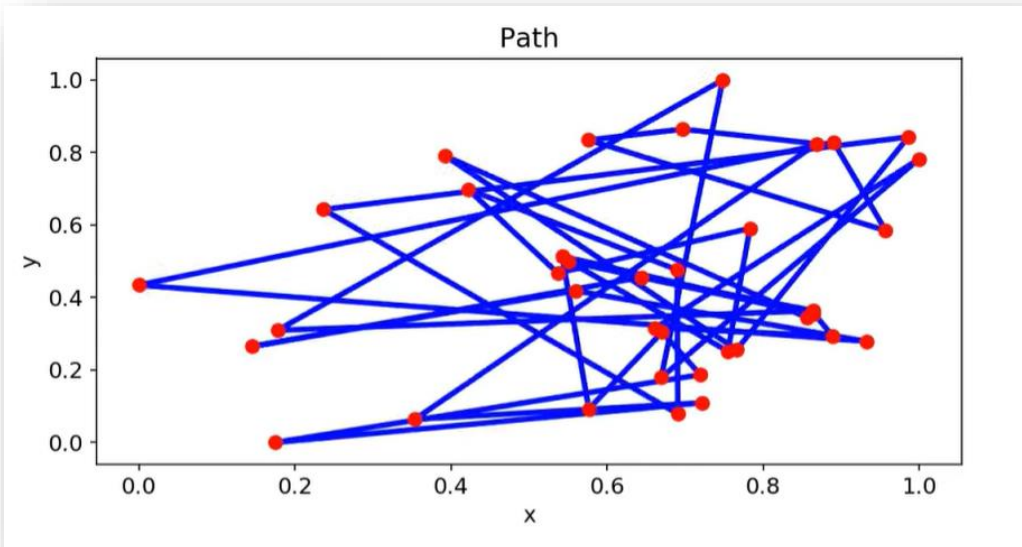
We exchange cities (modification)...

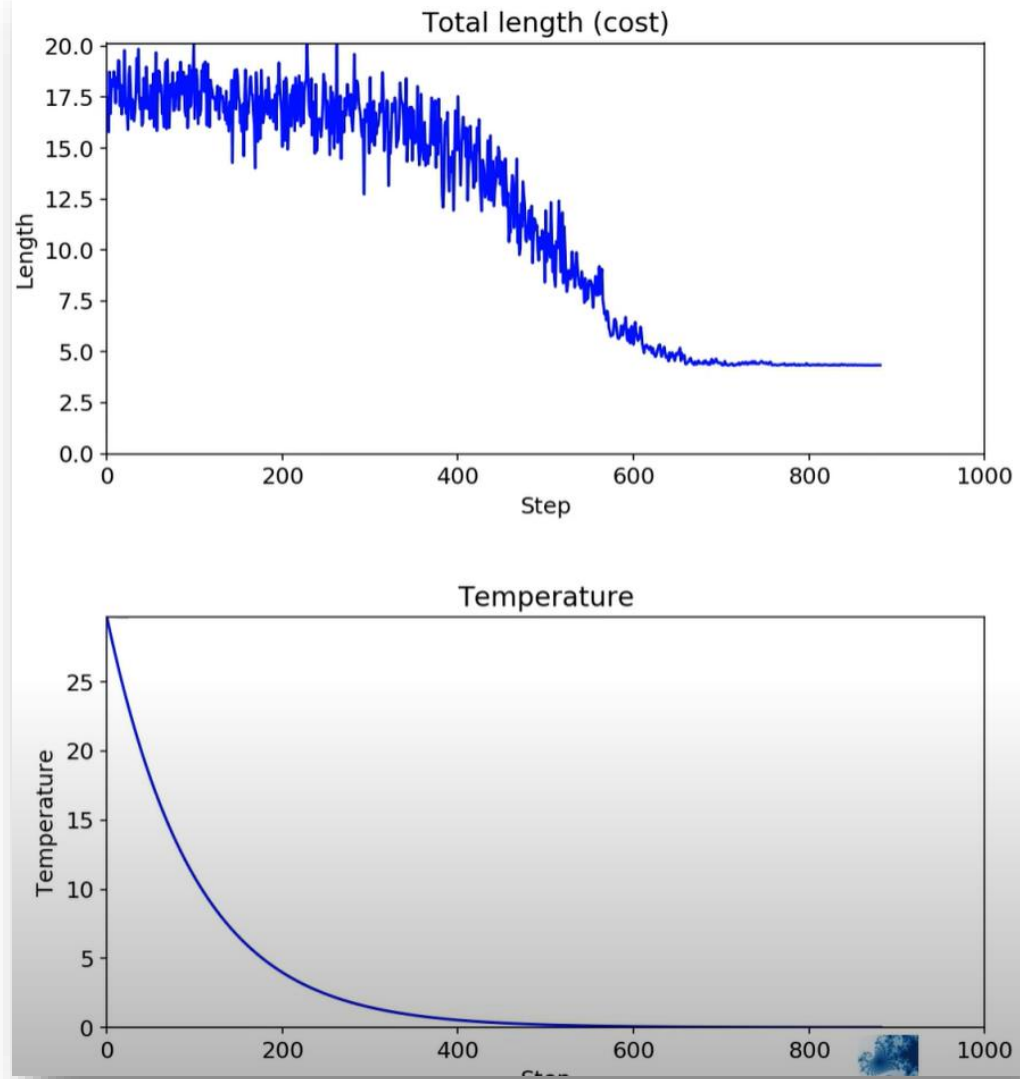
T1	T2	T3	T4	T5	retour	Coût
D	A	C	E	B	D	
	1	2	4	3	2	12



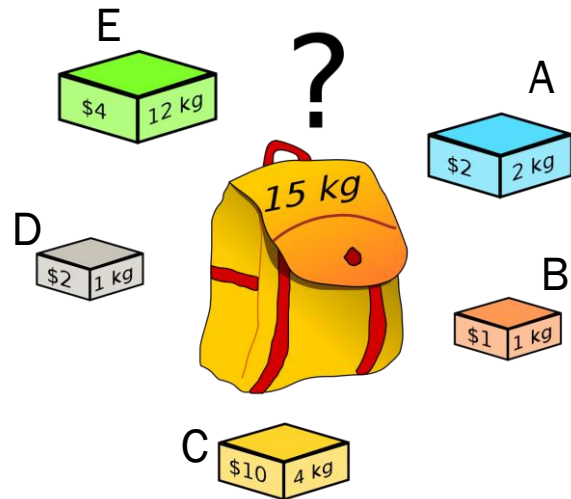
Trouver une solution
« voisine »







PROBLÈME DU SAC À DOS



Trouver une solution
« voisine »

(A,B,C,D,E)
(1,0,1,1,0) – Valeur (Énergie) = 14\$



(1,0,1,0,0) – Valeur (Énergie) = 12\$

Attention: ici, nous tentons de maximiser, il faudra ajuster le critère d'acceptation dans le recuit simulé

Algorithm 2: Simulated Annealing Optimizer

```
 $T \leftarrow T_{max}$ 
 $\mathbf{x} \leftarrow$  generate the initial candidate solution
 $E \leftarrow E(\mathbf{x})$  compute the energy of the initial solution
while  $(T > T_{min})$  and  $(E > E_{th})$  do
     $\mathbf{x}_{new} \leftarrow$  generate a new candidate solution
     $E_{new} \leftarrow$  compute the energy of the new candidate  $\mathbf{x}_{new}$ 
     $\Delta E \leftarrow E_{new} - E$ 
    if Accept  $(\Delta E, T)$  then
         $\mathbf{x} \leftarrow \mathbf{x}_{new}$ 
         $E \leftarrow E_{new}$ 
    end
     $T \leftarrow \frac{T}{\alpha}$  cool the temperature
end
return  $\mathbf{x}$ 
```



EN RÉSUMÉ

- Introduction à l'algorithme de Recuit Simulé

Vous explorerez plus
durant le devoir 1

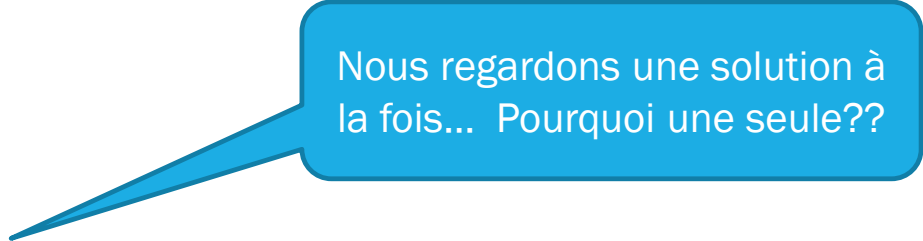
Partie 3

Algorithme génétique

MODIFICATION ALÉATOIRE

Algorithme générique:

- Débuter avec une solution initiale (gloutonne ou autre):
- Répéter pour N itérations:
 - Faire un changement local
 - Si la solution est meilleure (selon une fonction de coût):
 - Garder le changement
 - Sinon
 - Garder le changement en fonction d'une probabilité

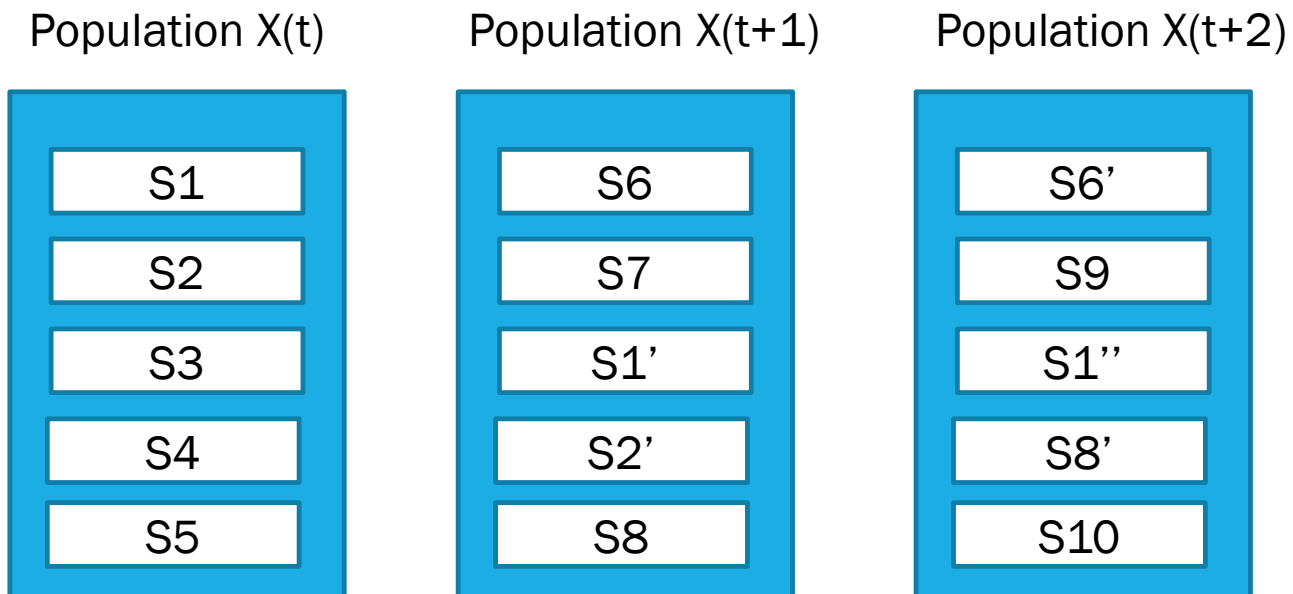


Nous regardons une solution à la fois... Pourquoi une seule??

ALGORITHME À BASE DE POPULATION

Idée de base est simple...

Plutôt que d'explorer une seule solution à la fois, **explorer M solutions en parallèle**. Les solutions sont des *individus* et l'ensemble des M solutions est une *population*.



EXEMPLES D'ALGORITHMES

Algorithmes inspirés de comportement sociaux ou de phénomènes naturels

- Comportement social des oiseaux
- Rain algorithm
- Colonie d'abeilles
- Colonie de fourmis
- Algorithme génétique

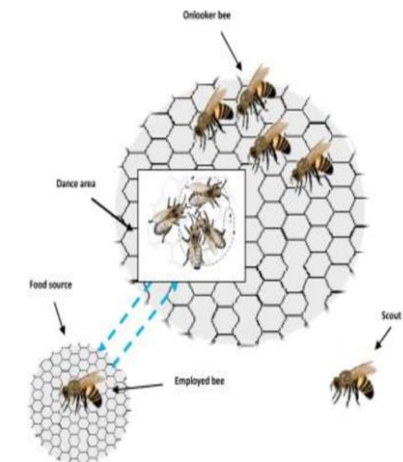


Figure 1: A Typical Bee Colony Model

Algorithme à base de population

- Générer une population initiale $X(t)$ de M individus
- Répéter
 - Sélectionner un ensemble d'individus en lien à leur « fitness »
 - Effectuer des **variations** sur ces individus pour générer de nouveaux individus qui feront partie de la population $X(t+1)$
- Jusqu'à une condition d'arrêt

Cette sélection peut se faire de différentes façons. Idée de « tournoi » (tournament selection).

Ces variations souvent impliquent plusieurs individus (combinaisons, direction par effet de masse)

(1) Nb max d'itérations
(2) Pas de meilleure solution dans N itérations
(3) Une solution meilleure qu'un seuil pré-déterminé sur la fonction de coût

Concentrons-nous sur un algorithme populaire:

Algorithme génétique

Algorithme inspiré de la génétique, soit les croisements et mutations d'ADN dans la reproduction. La population "parents" la mieux adaptée (selon la fonction de coût) est modifiée pour inclure des "enfants" qui seront ajoutés à la population.

Croisement

Échanger des portions provenant des 2 parents

Mutation

Modifier un élément d'un individu

EXEMPLE TSP

Individus
choisis de la
population

Parents

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

9	8	7	6	5	4	3	2	1
---	---	---	---	---	---	---	---	---

Offspring

					6	7	8	
--	--	--	--	--	---	---	---	--

9	5	4	3	2	6	7	8	1
---	---	---	---	---	---	---	---	---

Crossover

1-point crossover – un seul endroit
2-point crossover – 2 endroits

Mutation

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

1	2	8	4	5	6	7	3	9
---	---	---	---	---	---	---	---	---

Normalement APRÈS les crossovers.

Algorithme à base de population – Algorithme Génétique

- Générer une population initiale $X(t)$ de M individus
- Répéter
 - Sélectionner un ensemble d'individus en lien à leur « fitness » (k tournoi)
 - Effectuer des **variations** sur ces individus pour générer de nouveaux individus qui feront partie de la population $X(t+1)$
- Jusqu'à une condition d'arrêt

Cette sélection peut se faire de différentes façons. Idée de « tournoi » (tournament selection).

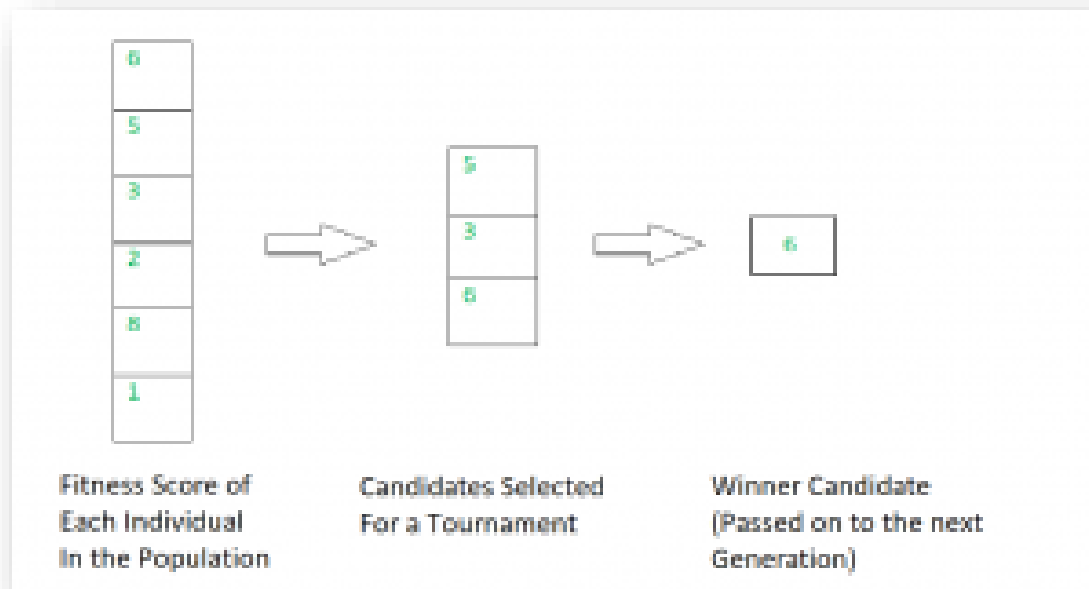
Ces variations sont avec les crossovers et les mutations.
Des probabilités (e.g. 0.7 crossovers et 0.1 mutations) sont utilisées

(1) Nb max d'itérations
(2) Pas de meilleure solution dans N itérations
(3) Une solution meilleure qu'un seuil pré-déterminé sur la fonction de coût

Tournament-Based Selection

Algorithm --

1. Select k individuals from the population and perform a tournament amongst them
2. Select the best individual from the k individuals
3. Repeat process 1 and 2 until you have the desired amount of population



Variations:

- Avec ou sans répétition.
- Déterministe ou probabiliste
- Taille de k .

Tournament-Based Selection

```
choose k (the tournament size) individuals from the population at random
choose the best individual from the tournament with probability p
choose the second best individual with probability  $p \cdot (1-p)$ 
choose the third best individual with probability  $p \cdot (1-p)^2$ 
and so on
```

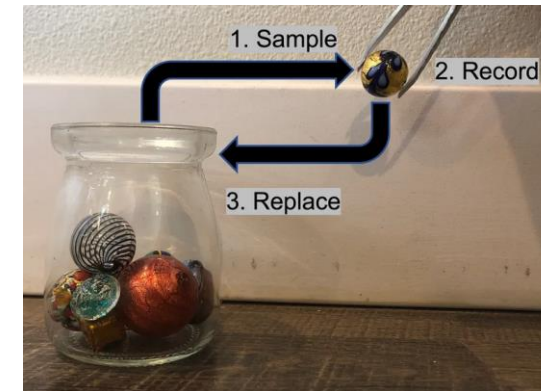
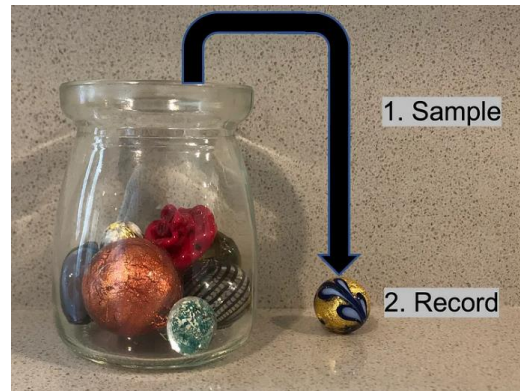
Deterministe: ($p = 1$) seulement le « fittest » est pris d'un tournoi

Probabiliste: par exemple $p=0.8$, alors pour un tournoi

- 80% de probabilité de prendre le meilleur
- 16% de probabilité de prendre le deuxième
- 3.2% de probabilité de prendre le 3^{ième}
- ...

Tournament-Based Selection

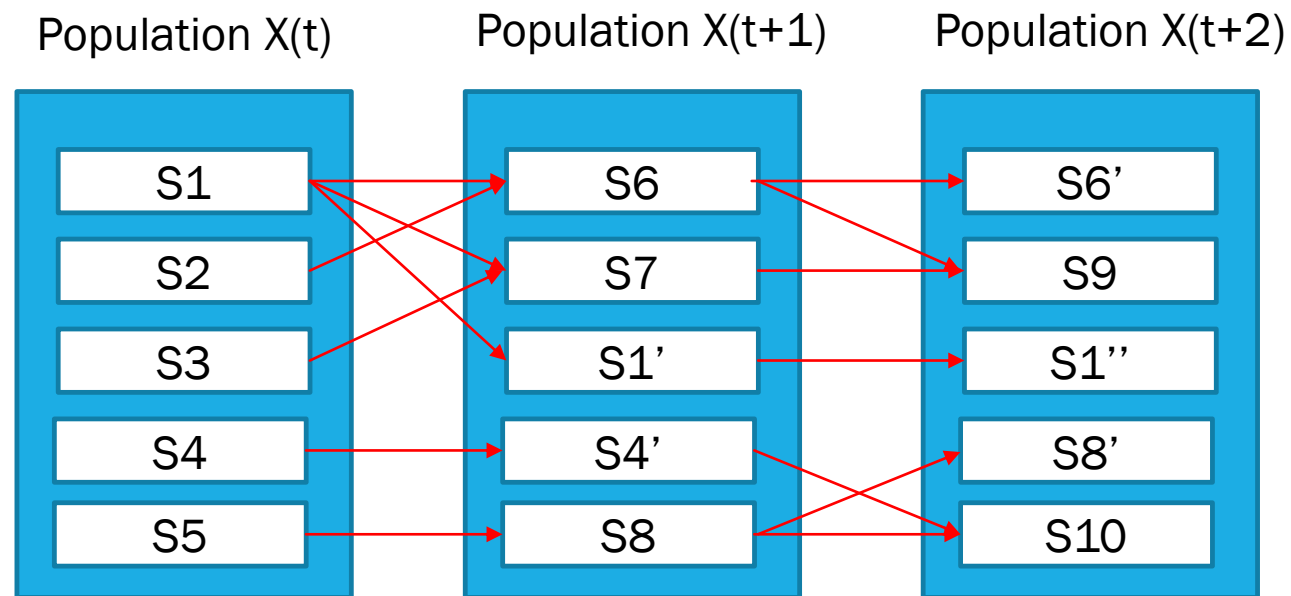
Avec ou sans remplacement



Algorithme à base de population – Algorithme Génétique

- Générer une population initiale $X(t)$ de M individus
- Répéter
 - Sélectionner un ensemble d'individus en lien à leur « fitness » (k tournoi)
 - Effectuer des ***variations*** sur ces individus pour générer de nouveaux individus qui feront partie de la population $X(t+1)$
- Jusqu'à une condition d'arrêt

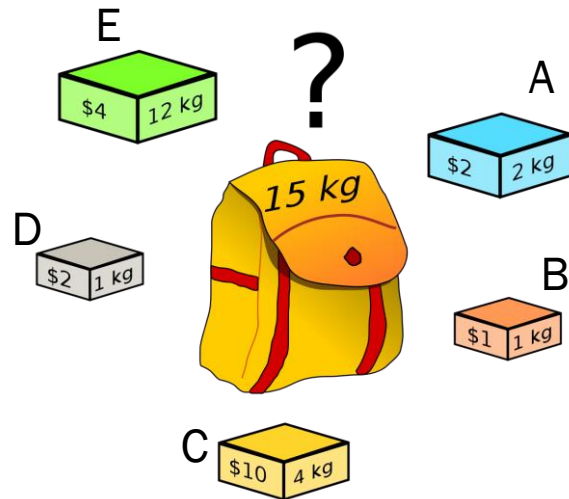
ALGORITHME GENETIQUE



Croisements
(S1/S2, S1/S3)
Mutations (S5)

Croisements
(S6/S7, S4'/S8)
Mutations (S6, S8)

PROBLÈME DU SAC À DOS



Variables: (A,B,C,D,E)

S1 : (1,1,0 // 0,1)

S2 : (1,0,1 // 1,0)



Crossover (e.g. taux 0.7)

S1': (1,1,0,0,1)

S2': (1,0,1,1,0)



Mutation
(e.g. taux 0.1 seulement S1'
modifiée)

S1':(1,0,0,0,1)

S2':(1,0,1,1,0)



EN RÉSUMÉ

- Introduction aux algorithmes par population de solutions
- Algorithme génétique

Vous explorerez plus
durant le devoir 1



EXPLORATION DE L'ESPACE DE SOLUTIONS

- Partie 1 – Propagation de contraintes
- Partie 2 – Recuit Simulé
- Partie 3 – Algorithme génétique