Les embouteillages

Année 2017 – 2018

GONZALES Swan, LEGARCON Hugo, COLLET Mario et MICHAUD Océane, élèves de seconde.

Encadrés par COUSIN Jildaz

Établissement : Lycée Edouard Branly **Chatellerault**

1. Présentation du sujet

Modéliser et simuler la formation et l'évolution d'un embouteillage. Quelle est l'influence de la densité du trafic ? De la vitesse ?

2. Annonce des conjectures et résultats obtenus

Une vitesse élevée et un grand nombre de véhicules favorisent la présence d'embouteillages.

3. Texte de l'article

Modélisation

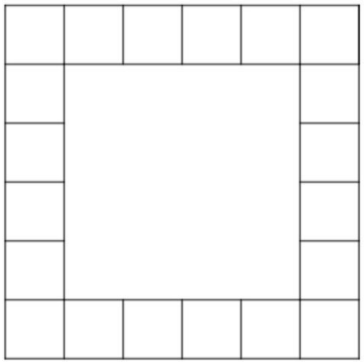


Illustration 1: Le circuit

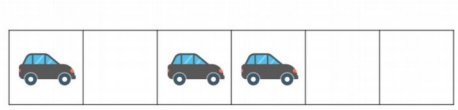
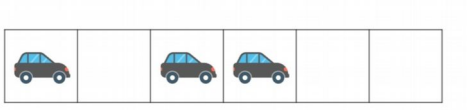
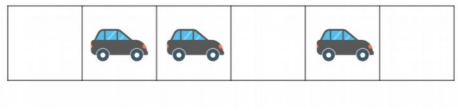
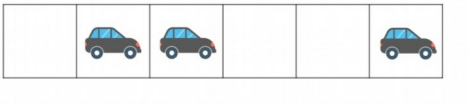
Nous avons décidé de faire nos simulations sur un circuit fermé, afin d'éviter de devoir faire un générateur aléatoire de véhicules dans notre algorithme de simulation.

Ce circuit est composé de cases.

A chaque étape, chaque véhicule avancera du nombre de cases correspondant à sa vitesse, sauf si un autre véhicule, placé devant, l'empêche d'avancer. Dans ce cas, le véhicule devra attendre la prochaine étape pour pouvoir enfin avancer.

Les véhicules ne peuvent pas se doubler. (1)

Exemples :

	Tous les véhicules ont une vitesse de 1	Tous les véhicules ont une vitesse de 2
Étape e		
Étape $e+1$		

Définitions

Un **embouteillage** se produit lorsqu'au moins un véhicule ne peut pas avancer car il y a une voiture juste devant lui.
Dans chacun des deux exemples précédents, il y a un embouteillage.

Un **ralentissement** se produit lorsqu'au moins un véhicule est contraint à avancer moins vite que lorsque sa vitesse est maximale. Par exemple si un véhicule qui avance à 2 cases/étape s'approche d'un véhicule à l'arrêt une case devant lui, il est contraint à ralentir pour ne pas rentrer dans le véhicule qui le précède.

Exemple : pour une vitesse de 2

Étape e	**Étape** $e+1$
Le véhicule situé sur la case en haut à gauche devrait avancer de deux cases mais comme il n'y a qu'une case libre devant, il va devoir ralentir.  *Illustration 2: Exemple vitesse 2*	 *Illustration 3: Exemple vitesse 2*

(2)

La **densité** du trafic représente le pourcentage de voitures par rapport au nombre de cases du circuit.

Simulations

Nous avons commencé par représenter le circuit à la main sur le tableau, pour réaliser des tests et essayer de comprendre le lien entre le nombre de véhicules, le nombre de cases et la présence d'embouteillages et en tirer des propriétés.

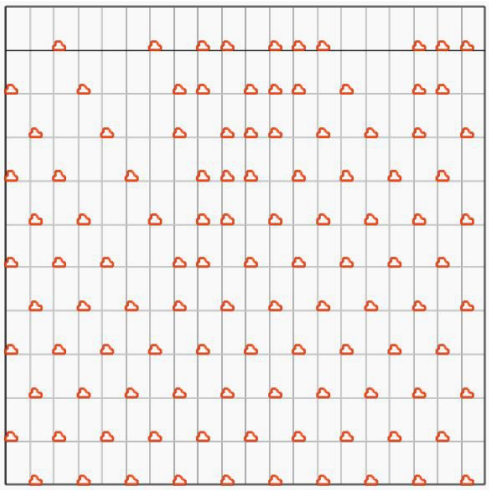
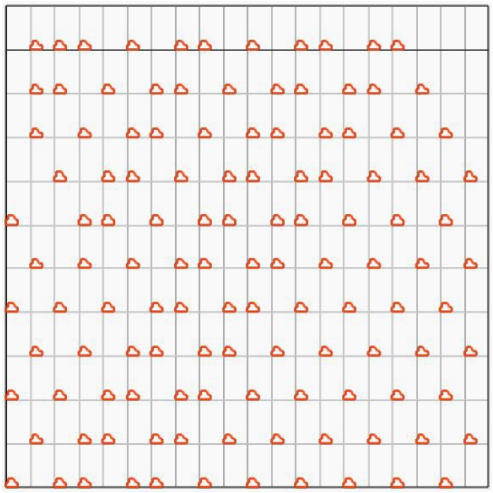
Nous avons ensuite commencé à travailler sur un algorithme, d'abord sur Scratch et finalement sur Algobox, afin de réaliser nos tests plus facilement et plus rapidement.

Conjectures pour une vitesse de 1

Tous les véhicules ont une vitesse d'une case par étape.

Nous avons observé sur les simulations que certains embouteillages se résorbent mais pas dans tous les cas.
On s'est rendu compte que lorsque le nombre de véhicules est strictement supérieur à la moitié du nombre de cases, il y aura au moins un embouteillage qui ne se résorbera pas.

Soit n le nombre de véhicules et c le nombre de cases. (3)

$n=\frac{c}{2}$	$n>\frac{c}{2}$
 *Illustration 4: L'embouteillage se résorbe*	 *Illustration 5: L'embouteillage ne se résorbe pas*
On voit que dans ce cas (ici $n=10$ et $c=20$), il n'y a aucun embouteillage à la fin.	On voit que dans ce cas (ici $n=11$ et $c=20$) , il y a 2 embouteillages qui ne se résorbent pas et 2 embouteillages qui se sont résorbés.

Propriétés et démonstrations

On peut prouver à l'aide de démonstrations mathématiques plusieurs propriétés de l'embouteillage, observées grâce aux simulations sur Algobox.
Nous travaillons dans cette partie avec une vitesse de 1 case par étape.

Propriété 1 :

Un embouteillage ne peut pas apparaître spontanément au cours du déroulement des étapes.

Démonstration :

Montrons que :
Si il y a un embouteillage à l'étape $e+1$ alors il y en avait un à l'étape e.
Par contraposée :
Si un véhicule est arrêté à l'étape $e+1$, c'est qu'un autre véhicule se trouve juste devant lui, véhicule qui était donc arrêté à l'étape e. Cela veut donc dire qu'il y avait un embouteillage à l'étape e.

(4)

Le nombre d'embouteillages ne peut donc pas augmenter : il ne peut que diminuer ou rester constant.

Propriété 2 :

Si un embouteillage ne diminue pas, alors il recule.

Démonstration :

Dans l'embouteillage, à chaque étape le véhicule à l'avant sort.
Donc si un embouteillage ne diminue pas, cela veut dire qu'il y en a un qui arrive derrière ($n-1+1=n$). Cet échange crée un mouvement vers l'arrière.

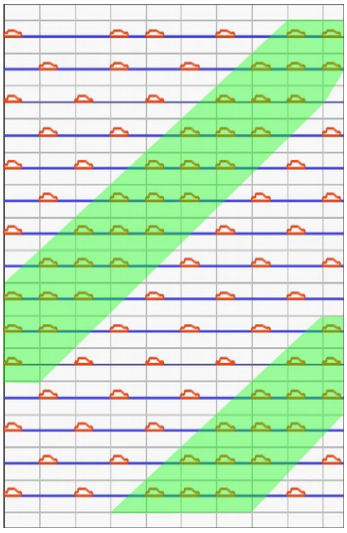


Illustration 6: L'embouteillage recule

Propriété 3 :

Pour une vitesse de 1 case par étape, si la densité du trafic est supérieure à 50 % alors il y a au moins un embouteillage qui ne se résorbera pas.

Démonstration :

- n est le nombre de voitures
- c est le nombre de cases du circuit fermé
- t est le nombre de cases vides

On veut démontrer que pour $n>\frac{c}{2}$, il y a un embouteillage.

Nous allons démontrer cela par l'absurde : on suppose que pour $n>\frac{c}{2}$, il n'y a pas d'embouteillage.

D'après notre définition, s'il n' y a pas d'embouteillage, il doit y avoir au moins une case vide devant chaque véhicule. Donc il y a plus de cases vides que de cases avec un véhicule : $t \geqslant n$

$c=n+t \geqslant n+n=2n$ donc $n \leqslant \frac{c}{2}$ contradiction avec $n>\frac{c}{2}$

Donc notre supposition de départ est fausse donc pour $n>\frac{c}{2}$, il y a au moins un embouteillage.

Propriété 4:

Pour une vitesse de 1 unité par étape, si la densité du trafic est inférieure à 50% alors il n'y a pas d'embouteillage permanent.

Remarque :

Un embouteillage permanent est un embouteillage qui ne se résorbe pas. Dans la situation de la propriété 4, il peut y avoir un embouteillage au départ mais il va se résorber avec le temps.

Démonstration :

On suppose que pour une densité inférieure à 50 %, il y a un embouteillage permanent, autrement dit, pour $n \leqslant \frac{c}{2}$, il y a un embouteillage permanent.

- b : nombre de véhicules bloqués dans un embouteillage (le véhicule à la tête de l'embouteillage n'est pas bloqué)

Remarque : s'il n'y a pas de nouvelles arrivées dans un embouteillage alors il se résorbe en b étapes.

1er cas : un seul embouteillage à l'initialisation

Supposons qu'il ne diminue jamais.
A chaque étape le véhicule de tête quitte l'embouteillage.
Si deux véhicules sont séparés par deux cases vides, lorsque le premier arrive dans l'embouteillage, à l'étape suivante, le véhicule de tête quitte l'embouteillage mais aucun véhicule n'arrive.
Pour qu'un embouteillage ne diminue jamais , il faut qu'aucun véhicule libre n'ait plus de 1 case devant lui.
La composition du circuit est alors

$$c = b + (n-b) + (n-b) = 2n - b$$

Véhicules libres Cases vides

Donc $n = \frac{c+b}{2} > \frac{c}{2}$

Donc pour qu'il y ait un embouteillage permanent, il faut que $n > \frac{c}{2}$, il y a contradiction avec une densité inférieure a 50%.
Notre supposition de départ est donc fausse donc au bout d'un certain temps, l'embouteillage diminue et finira par se résorber.

2ème cas : plusieurs embouteillages

On suppose qu'aucun des embouteillages ne diminue.
Pour qu'aucun embouteillage ne diminue jamais, il faut qu'aucun véhicule libre n'ait plus de une case devant lui.
On retrouve l'égalité $c = b + (n-b) + (n-b)$ et on retrouve la même conclusion qu'avec un seul embouteillage. Les embouteillages vont donc diminuer les uns après les autres, jusqu'à se résorber complètement.

On peut conclure de ces 2 propriétés que pour une vitesse globale de 1 unité :

- Si $n \leq \frac{c}{2}$ alors tous les embouteillages se résorbent
- Si $n > \frac{c}{2}$ alors il y au au moins un embouteillage permanent

On dit que la valeur $\frac{c}{2}$ est le **seuil de saturation du trafic**.

Le seuil de saturation pour une vitesse de 1 est donc de 50 %.

Augmentation de la vitesse

Nous avons constaté que la vitesse influait sur le seuil de saturation du trafic. Nous avons donc étudié le rapport entre ce seuil de saturation et la vitesse.

Comme vu précédemment, pour une vitesse globale de 1 case par étape, le seuil de densité critique est de 50%. Nous avons cherché une formule permettant de trouver le seuil de densité critique pour une vitesse v quelconque.

Pour qu'un véhicule puisse avancer, il doit avoir v cases libres devant lui.

Pour que les n véhicules puissent avancer, il faut donc $c \geqslant n+nv$

$c \geqslant n(1+v)$ donc $\frac{n}{c} \leqslant \frac{1}{1+v}$ donc la densité doit être inférieure ou égale à $\frac{1}{1+v}$.

Pour une vitesse de 2 le seuil de saturation du trafic est de 33%
Pour une vitesse de 3 le seuil de saturation du trafic est de 25%
Pour une vitesse de 4 le seuil de saturation du trafic est de 20%
Pour une vitesse de 5 le seuil de saturation du trafic est de 16%
Pour une vitesse de 6 le seuil de saturation du trafic est de 14%

(5)

Conclusion

D'après notre étude, si tous les véhicules ont la même vitesse alors la présence d'embouteillage est liée à la densité du trafic. Plus le trafic est dense, plus il faut limiter la vitesse pour réduire les embouteillages. Pour être plus proche de la réalité, il faudrait envisager de nouvelles situations, par exemple :

- des véhicules ayant des vitesses différentes
- des intersections
- la possibilité de dépasser

(6)

Notes de l'édition

(1) Lorsque l'on découpe le circuit en cases, un véhicule ne peut prendre qu'un nombre fini de positions alors qu'en réalité sur un circuit il y a une infinité de positions. On dit qu'on a « discrétisé » le circuit. De même, le temps a été discrétisé en une suite d'étapes.Ce procédé de discrétisation est très utilisé pour simuler des phénomènes réels.

(2) Précision : les déplacements ont lieu entre deux étapes successives. Si à une étape donnée un véhicule est juste derrière un autre alors il ne peut pas se déplacer comme c'est le cas pour le véhicule central dans les deux exemples.

(3) Dans les illustrations qui suivent chaque ligne représente l'état du circuit à une étape donnée. Le véhicule situé sur la dernière case d'une ligne se trouvera à l'étape suivante sur la première case car le circuit est fermé.

(4) Ce n'est pas une démonstration par contraposée, La contraposée de la propriété étant : s'il n'y a pas d'embouteillage à l'étape e alors il n'y en a pas à l'étape e+1. Le lecteur est invité à faire quelques schémas pour comprendre le raisonnement.

(5) Le raisonnement précédent est incomplet. Il suppose que tous les véhicules avancent à leur vitesse maximale mais il peut y avoir des ralentissements sans création d'embouteillage. Le lecteur le vérifiera sur l'exemple suivant avec une vitesse v = 2.
Les véhicules sont A, B et C. _ désigne une case vide. La situation initiale est A_ _B_C_.

(6) Ce sujet de modélisation mériterait d'être poursuivi. En particulier le modèle proposé ne permet pas la création d'embouteillages ni l'augmentation de la taille d'un embouteillage ce que chaque vacancier a déjà expérimenté.

Annexe : code de l'algorithme

Cet algorithme permet de simuler des embouteillages en fonction :
- du nombre de véhicules
- du nombre de cases
- de la vitesse
- du nombre d'étapes représentées

Et en suivant ces contraintes :
- utilisation d'un système vitesse/cases, comme un plateau de jeu
- les véhicules ne peuvent pas se doubler
- un véhicule ne peut aller sur la case occupée par le véhicule suivant à l'étape précédente

```
VARIABLES
  n_cases EST_DU_TYPE NOMBRE
  Plan_des_voitures EST_DU_TYPE LISTE
  i EST_DU_TYPE NOMBRE
  n_étape EST_DU_TYPE NOMBRE
  n_étape_max EST_DU_TYPE NOMBRE
  Plan_des_voitures_2 EST_DU_TYPE LISTE
  n_véhicules EST_DU_TYPE NOMBRE
  r EST_DU_TYPE NOMBRE
  e EST_DU_TYPE NOMBRE
  taille_voiture EST_DU_TYPE NOMBRE
  V EST_DU_TYPE NOMBRE
  Vf EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  taille_voiture PREND_LA_VALEUR 0.1
  LIRE n_cases
  LIRE n_véhicules
  LIRE V
  Vf PREND_LA_VALEUR 0
  SI (n_véhicules>n_cases) ALORS
    DEBUT_SI
    AFFICHER "Il ne peut pas y avoir plus de véhicules que de cases,"
    AFFICHER "veuillez rentrer de nouveau les valeurs :"
    LIRE n_cases
    LIRE n_véhicules
    FIN_SI
  LIRE n_étape_max
  e PREND_LA_VALEUR 0
  n_étape PREND_LA_VALEUR 0
  POUR i ALLANT_DE 0 A n_cases-1
    DEBUT_POUR
    Plan_des_voitures[i] PREND_LA_VALEUR 0
    FIN_POUR
  TANT_QUE (e<n_véhicules) FAIRE
    DEBUT_TANT_QUE
    r PREND_LA_VALEUR ALGOBOX_ALEA_ENT(0,n_cases-1)
    SI (Plan_des_voitures[r]==0) ALORS
      DEBUT_SI
      Plan_des_voitures[r] PREND_LA_VALEUR 1
      e PREND_LA_VALEUR e+1
      FIN_SI
    FIN_TANT_QUE
  POUR i ALLANT_DE 0 A n_cases-1
    DEBUT_POUR
    SI (Plan_des_voitures[i]>=1) ALORS
      DEBUT_SI
      TRACER_SEGMENT (i,-n_étape)->(i+5*taille_voiture,-n_étape)
      TRACER_SEGMENT (i+5*taille_voiture,-n_étape)->(i+5*taille_voiture,-n_étape+taille_voiture)
      TRACER_SEGMENT (i+taille_voiture*5,-n_étape+taille_voiture)->(i+4*taille_voiture,-n_étape+taille_voiture)
      TRACER_SEGMENT (i+4*taille_voiture,-n_étape+taille_voiture)->(i+3*taille_voiture,-n_étape+2*taille_voiture)
      TRACER_SEGMENT (i+3*taille_voiture,-n_étape+2*taille_voiture)->(i+1*taille_voiture,-n_étape+2*taille_voiture)
      TRACER_SEGMENT (i+1*taille_voiture,-n_étape+2*taille_voiture)->(i+1*taille_voiture,-n_étape+1*taille_voiture)
      TRACER_SEGMENT (i+1*taille_voiture,-n_étape+1*taille_voiture)->(i+0*taille_voiture,-n_étape+1*taille_voiture)
      TRACER_SEGMENT (i+0*taille_voiture,-n_étape+1*taille_voiture)->(i+0*taille_voiture,-n_étape+0*taille_voiture)
      FIN_SI
    FIN_POUR
  n_étape PREND_LA_VALEUR n_étape+1
  TANT_QUE (n_étape<n_étape_max) FAIRE
    DEBUT_TANT_QUE
    POUR i ALLANT_DE 0 A n_cases-1
      DEBUT_POUR
      Plan_des_voitures_2[i] PREND_LA_VALEUR 0
      FIN_POUR
    POUR i ALLANT_DE 0 A n_cases-1
      DEBUT_POUR
      SI (Plan_des_voitures[i]>=1) ALORS
        DEBUT_SI
        SI (i==n_cases-1) ALORS
          DEBUT_SI
          SI (Plan_des_voitures[0]==0) ALORS
            DEBUT_SI
            SI (Plan_des_voitures[i+2]==0 ET V>=2) ALORS
              DEBUT_SI
              SI (Plan_des_voitures[i+3]==0 ET V==3) ALORS
                DEBUT_SI
                Plan_des_voitures_2[2] PREND_LA_VALEUR 1
                FIN_SI
                SINON
                  DEBUT_SINON
                  Plan_des_voitures_2[1] PREND_LA_VALEUR 1
                  FIN_SINON
              FIN_SI
              SINON
                DEBUT_SINON
                Plan_des_voitures_2[0] PREND_LA_VALEUR 1
                FIN_SINON
            FIN_SI
            SINON
              DEBUT_SINON
```

```
              Plan_des_voitures_2[i] PREND_LA_VALEUR 1
              FIN_SINON
          FIN_SI
          SINON
            DEBUT_SINON
            SI (Plan_des_voitures[i+1]==0) ALORS
              DEBUT_SI
              SI (Plan_des_voitures[i+2]==0 ET V>=2) ALORS
                DEBUT_SI
                SI (Plan_des_voitures[i+3]==0 ET V==3) ALORS
                  DEBUT_SI
                  Plan_des_voitures_2[i+3] PREND_LA_VALEUR 1
                  FIN_SI
                  SINON
                    DEBUT_SINON
                    Plan_des_voitures_2[i+2] PREND_LA_VALEUR 1
                    FIN_SINON
                FIN_SI
                SINON
                  DEBUT_SINON
                  Plan_des_voitures_2[i+1] PREND_LA_VALEUR 1
                  FIN_SINON
              FIN_SI
              SINON
                DEBUT_SINON
                Plan_des_voitures_2[i] PREND_LA_VALEUR 1
                FIN_SINON
            FIN_SINON
        FIN_SI
      FIN_POUR
    POUR i ALLANT_DE 0 A n_cases-1
      DEBUT_POUR
      SI (Plan_des_voitures_2[i]==1) ALORS
        DEBUT_SI
        TRACER_SEGMENT (i,-n_étape)->(i+5*taille_voiture,-n_étape)
        TRACER_SEGMENT (i+5*taille_voiture,-n_étape)->(i+5*taille_voiture,-n_étape+taille_voiture)
        TRACER_SEGMENT (i+taille_voiture*5,-n_étape+taille_voiture)->(i+4*taille_voiture,-n_étape+taille_voiture)
        TRACER_SEGMENT (i+4*taille_voiture,-n_étape+taille_voiture)->(i+3*taille_voiture,-n_étape+2*taille_voiture)
        TRACER_SEGMENT (i+3*taille_voiture,-n_étape+2*taille_voiture)->(i+1*taille_voiture,-n_étape+2*taille_voiture)
        TRACER_SEGMENT (i+1*taille_voiture,-n_étape+2*taille_voiture)->(i+1*taille_voiture,-n_étape+1*taille_voiture)
        TRACER_SEGMENT (i+1*taille_voiture,-n_étape+1*taille_voiture)->(i+0*taille_voiture,-n_étape+1*taille_voiture)
        TRACER_SEGMENT (i+0*taille_voiture,-n_étape+1*taille_voiture)->(i+0*taille_voiture,-n_étape+0*taille_voiture)
        FIN_SI
      FIN_POUR
    n_étape PREND_LA_VALEUR n_étape+1
    POUR i ALLANT_DE 0 A n_cases-1
      DEBUT_POUR
      Plan_des_voitures[i] PREND_LA_VALEUR 0
      FIN_POUR
    POUR i ALLANT_DE 0 A n_cases-1
      DEBUT_POUR
      SI (Plan_des_voitures_2[i]>=1) ALORS
        DEBUT_SI
        SI (i==n_cases-1) ALORS
          DEBUT_SI
          SI (Plan_des_voitures_2[0]==0) ALORS
            DEBUT_SI
            SI (V>=2 ET Plan_des_voitures_2[i+2]==0) ALORS
              DEBUT_SI
              SI (V==3 ET Plan_des_voitures_2[i+3]==0) ALORS
                DEBUT_SI
                Plan_des_voitures[2] PREND_LA_VALEUR 1
                FIN_SI
                SINON
                  DEBUT_SINON
                  Plan_des_voitures[1] PREND_LA_VALEUR 1
                  FIN_SINON
              FIN_SI
              SINON
                DEBUT_SINON
                Plan_des_voitures[0] PREND_LA_VALEUR 1
                FIN_SINON
            FIN_SI
            SINON
              DEBUT_SINON
              Plan_des_voitures[i] PREND_LA_VALEUR 1
              FIN_SINON
          FIN_SI
          SINON
            DEBUT_SINON
            SI (Plan_des_voitures_2[i+1]==0) ALORS
              DEBUT_SI
              SI (V>=2 ET Plan_des_voitures_2[i+2]==0) ALORS
                DEBUT_SI
                SI (V==3 ET Plan_des_voitures_2[i+3]==0) ALORS
                  DEBUT_SI
                  Plan_des_voitures[i+3] PREND_LA_VALEUR 1
                  FIN_SI
                  SINON
                    DEBUT_SINON
                    Plan_des_voitures[i+2] PREND_LA_VALEUR 1
                    FIN_SINON
                FIN_SI
                SINON
                  DEBUT_SINON
                  Plan_des_voitures[i+1] PREND_LA_VALEUR 1
                  FIN_SINON
              FIN_SI
              SINON
                DEBUT_SINON
                Plan_des_voitures[i] PREND_LA_VALEUR 1
                FIN_SINON
            FIN_SINON
        FIN_SI
      FIN_POUR
    POUR i ALLANT_DE 0 A n_cases-1
      DEBUT_POUR
      SI (Plan_des_voitures[i]==1) ALORS
        DEBUT_SI
        TRACER_SEGMENT (i,-n_étape)->(i+5*taille_voiture,-n_étape)
        TRACER_SEGMENT (i+5*taille_voiture,-n_étape)->(i+5*taille_voiture,-n_étape+taille_voiture)
```

```
201         TRACER_SEGMENT (i+taille_voiture*5,-n_étape+taille_voiture)->(i+4*taille_voiture,-n_étape+taille_voiture)
202         TRACER_SEGMENT (i+4*taille_voiture,-n_étape+taille_voiture)->(i+3*taille_voiture,-n_étape+2*taille_voiture)
203         TRACER_SEGMENT (i+3*taille_voiture,-n_étape+2*taille_voiture)->(i+1*taille_voiture,-n_étape+2*taille_voiture)
204         TRACER_SEGMENT (i+1*taille_voiture,-n_étape+2*taille_voiture)->(i+1*taille_voiture,-n_étape+1*taille_voiture)
205         TRACER_SEGMENT (i+1*taille_voiture,-n_étape+1*taille_voiture)->(i+0*taille_voiture,-n_étape+1*taille_voiture)
206         TRACER_SEGMENT (i+0*taille_voiture,-n_étape+1*taille_voiture)->(i+0*taille_voiture,-n_étape+0*taille_voiture)
207         FIN_SI
208       FIN_POUR
209     n_étape PREND_LA_VALEUR n_étape+1
210     FIN_TANT_QUE
211 FIN_ALGORITHME
```