(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

COURBEVOIE

(11) **N° de publication :** **3 095 042**

(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **19 03913**

(51) Int Cl[8] : ***G 01 C 21/34*** *(2019.01), G 01 C 21/20, G 06 Q 10/04, H 04 L 12/701, 12/721, 29/02, G 06 F 17/10, G 06 N 3/04, 20/00*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 12.04.19.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 16.10.20 Bulletin 20/42.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

( ) **Demande(s) d'extension :**

(71) **Demandeur(s) :** *SAFRAN ELECTRONICS & DEFENSE SAS* — FR.

(72) **Inventeur(s) :** OSANLOU Kevin.

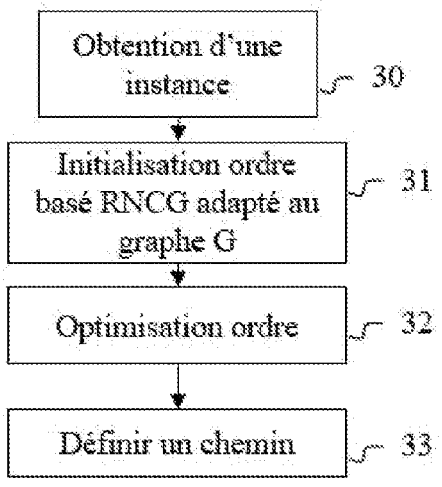
(73) **Titulaire(s) :** SAFRAN ELECTRONICS & DEFENSE SAS.

(74) **Mandataire(s) :** LE GUEN ET ASSOCIES.

(54) PROCEDE DE DEFINITION D'UN CHEMIN.

(57) Procédé de définition d'un chemin à suivre par un véhicule dans un environnement représenté par un graphe de nœuds reliés par des arêtes, chaque nœud dudit graphe représentant une position pouvant être prise par ledit véhicule, chaque arête entre deux nœuds étant associée à un coût de transition entre les deux nœuds. Le procédé comprend: obtenir (30) une instance dudit graphe, chaque instance étant représentée par un nœud de départ, un nœud d'arrivée et un ensemble de nœuds obligatoires par lesquels le véhicule doit passer, lesdits nœuds de départ, d'arrivée et obligatoires formant les nœuds de l'instance; obtenir (31) un ordre initial de parcours desdits nœuds obligatoires en appliquant une procédure utilisant un réseau de neurones convolutif de graphe adapté audit graphe à ladite instance; exécuter (32) une procédure d'optimisation locale de l'ordre des nœuds obligatoires afin d'obtenir un ordre optimisé, ladite procédure étant initialisée avec l'ordre initial; définir (33) un chemin pour ladite instance à résoudre à partir de l'ordre optimisé et pour chaque paire de nœuds de ladite instance, d'un plus court chemin entre les nœuds de ladite paire.

Fig. 3



FR 3 095 042 - A1

Description

Titre de l'invention : PROCEDE DE DEFINITION D'UN CHEMIN

Domaine technique

[0001] L'invention concerne un procédé de définition d'un chemin à suivre par un véhicule dans un environnement représenté par un graphe de nœuds reliés par des arêtes, chaque nœud dudit graphe représentant une position pouvant être prise par ledit véhicule, chaque arête entre deux nœuds étant associée à un coût de transition entre les deux nœuds, un dispositif et un système mettant en œuvre le procédé.

Technique antérieure

[0002] Ces dernières années ont vu apparaître des véhicules dits autonomes, capables de se déplacer dans un environnement de manière autonome, c'est-à-dire sans intervention humaine ou de manière semi-autonome, c'est-à-dire avec une intervention humaine limitée. Ces véhicules autonomes regroupent des véhicules automobiles capables de circuler sur des routes en compagnie d'autres véhicules et des drones volants, roulants ou naviguant sous ou sur l'eau.

[0003] On demande en général à ces véhicules autonomes d'atteindre un point, appelé aussi nœud, d'arrivée à partir d'un point de départ sans lui préciser le chemin à suivre pour atteindre le point d'arrivée. Dans certains cas, le chemin peut être contraint, le véhicule autonome devant passer par une ou plusieurs positions obligatoires entre le point de départ et le point d'arrivée.

[0004] Généralement, le nombre de positions que peut adopter le véhicule autonome dans son environnement n'est pas infini, ce qui permet de représenter ledit environnement sous forme d'un graphe de nœuds reliés par des arêtes. Dans ce graphe, chaque arête reliant deux nœuds peut être associée à un poids représentatif d'un coût de transition entre les deux nœuds.

[0005] La Fig. 1 illustre un graphe G représentatif d'un environnement dans lequel peut évoluer un véhicule.

[0006] Le graphe G comprend « 8 » nœuds reliés par des arêtes. Chaque arête est associée à un poids. Par exemple, l'arête entre le nœud 1 et le nœud 2 est associée à un poids de valeur « 2 ». Dans un exemple, un véhicule V doit circuler dans le graphe G depuis un nœud 1 vers un nœud 6 en passant obligatoirement au moins une fois par un nœud 3. On appelle par la suite un triplet constitué d'un nœud de départ, d'un nœud d'arrivée et d'un ensemble de nœuds obligatoires, une instance du graphe G, notée I(S, D, M), S étant le nœud de départ, D étant le nœud d'arrivée et M étant l'ensemble des nœuds obligatoires. Par ailleurs, nous appelons résolution de l'instance I(S, D, M) une procédure permettant de rechercher un meilleur chemin dans un graphe G pour une

instance I(S, D, M) dudit graphe. Une résolution d'instance consiste à rechercher le chemin minimisant un coût de transition global, le coût de transition global étant une somme de coûts de transition entre chaque paire de nœuds successifs composant ledit chemin. Dans l'exemple de la Fig. 1, l'instance à résoudre est donc I (nœud 1, nœud 6, {nœud 3}). La résolution de cette instance donne un ordre de parcours d'un ensemble de nœuds, du nœud 1 au nœud 6. L'ordre de parcours suivant est ainsi obtenu :

[0007] {nœud 1, nœud 2, nœud 5, nœud 3, nœud 5, nœud 6}.

[0008] Dans l'exemple de la Fig. 1, le graphe G comprend peu de nœuds. Pour un nombre de nœuds de cet ordre, la résolution d'une instance est relativement simple, même si cette instance comprend des nœuds obligatoires. Il est en effet possible de tester exhaustivement dans un temps raisonnable chaque combinaison possible des suites de nœuds pour trouver celle qui résout l'instance avec un coût minimal. Une telle méthode d'optimisation serait inapplicable pour des graphes comprenant un grand nombre de nœuds, *a fortiori* avec des instances à résoudre comprenant un grand nombre de nœuds obligatoires.

[0009] Des méthodes connues existent pour résoudre des instances dans un graphe comprenant un grand nombre de nœuds telles que les méthodes 2-opt, 3-opt, l'heuristique de Lin-Kernighan, etc. Ces méthodes ont toutefois des temps de résolution très importants, où, lorsque le temps de résolution est contraint, génèrent des solutions très sous-optimales.

[0010] Il est souhaitable de pallier ces inconvénients de l'état de la technique. Il est notamment souhaitable de proposer une méthode qui permette de résoudre une instance comprenant un ensemble de nœuds obligatoires important dans un graphe comprenant un grand nombre de nœuds de manière optimale ou quasi optimale dans un temps raisonnable.

Exposé de l'invention

[0011] Selon un premier aspect de la présente invention, la présente invention concerne un procédé de définition d'un chemin à suivre par un véhicule dans un environnement représenté par un graphe de nœuds reliés par des arêtes, chaque nœud dudit graphe représentant une position pouvant être prise par ledit véhicule, chaque arête entre deux nœuds étant associée à un coût de transition entre les deux nœuds. Le procédé comprend : obtenir une instance dudit graphe, dite instance à résoudre, chaque instance étant représentée par un nœud de départ, un nœud d'arrivée et un ensemble de nœuds obligatoires par lesquels le véhicule doit passer, lesdits nœuds de départ, d'arrivée et obligatoires formant les nœuds de l'instance à résoudre ; obtenir une information représentative d'un ordre initial de parcours desdits nœuds obligatoires en appliquant une procédure utilisant un réseau de neurones convolutif de graphe, dit RNCG, adapté

audit graphe à ladite instance à résoudre; exécuter une procédure d'optimisation locale de l'ordre des nœuds obligatoires afin d'obtenir un ordre des nœuds obligatoires optimisé, ladite procédure étant initialisée avec ladite information représentative de l'ordre initial; définir un chemin pour ladite instance à résoudre à partir de l'ordre des nœuds obligatoires optimisé et pour chaque paire de nœuds de ladite instance à résoudre, d'un plus court chemin entre les nœuds de ladite paire.

[0012] Selon un mode de réalisation, la procédure utilisant le RNCG est une procédure itérative prenant une instance du graphe en entrée, dite instance d'entrée et générant une information représentative d'un ordre initial de parcours des nœuds obligatoires de l'instance d'entrée en sortie, dite information de sortie, et comprenant à chaque itération : appliquer le RNCG à ladite instance d'entrée afin d'obtenir pour chaque nœud du graphe une valeur représentative d'une probabilité d'être le prochain nœud obligatoire à traverser à la suite du nœud de départ de l'instance d'entrée ; identifier le nœud obligatoire de l'ensemble de nœuds obligatoires de l'instance d'entrée possédant la probabilité la plus élevée ; insérer le nœud obligatoire identifié dans une liste ordonnée de nœuds obligatoires; remplacer le nœud de départ de l'instance d'entrée par le nœud obligatoire identifié et retirer le nœud obligatoire identifié de l'ensemble de nœuds obligatoires de l'instance d'entrée ; les itérations prenant fin lorsqu'un nombre de nœuds dans l'ensemble de nœuds obligatoires de l'instance d'entrée est égal à un, le nœud obligatoire restant dans l'ensemble de nœuds obligatoires de l'instance d'entrée étant inséré dans ladite liste ordonnée et ladite liste ordonnée formant l'information de sortie.

[0013] Selon un mode de réalisation, la procédure d'optimisation locale de l'ordre des nœuds obligatoires est une procédure itérative prenant une instance, dite instance initiale, et une information représentative d'un ordre de nœuds obligatoires, dit ordre initial, en entrée et générant une information représentative d'un ordre de nœuds obligatoires optimisé en sortie, dit ordre optimisé, chaque information représentative d'un ordre de nœuds obligatoires étant une liste de nœuds obligatoires ordonnés, et comprend à chaque itération: déterminer une paire de nœuds obligatoires, dits nœuds pivots, parmi les nœuds obligatoires de la liste de nœuds obligatoires ordonnée de l'ordre initial, dite liste initiale, et obtenir une liste de nœuds obligatoires ordonnée modifiée, dite liste modifiée, dans laquelle les nœuds obligatoires de la liste initiale situés entre les deux nœuds pivots ont été intervertis symétriquement, la paire de nœuds pivots déterminée étant différente de toutes paires de nœuds pivots déjà utilisées pour modifier l'ordre des nœuds obligatoires dans la liste initiale ; définir une information représentative de l'ordre des nœuds obligatoires dans la liste modifiée comme étant un ordre optimisé temporaire lorsqu'un coût de transition global d'un chemin entre les nœuds de départ et d'arrivée de l'instance initiale en suivant l'ordre

des nœuds obligatoires dans la liste modifiée est inférieur à un coût de transition global dudit chemin en suivant l'ordre des nœuds obligatoires dans la liste initiale; et définir l'ordre optimisé temporaire comme étant l'ordre initial ; mettre fin aux itérations de la procédure d'optimisation locale lorsqu'une condition d'arrêt est remplie, l'ordre optimisé étant le dernier ordre optimisé temporaire obtenu par la procédure d'optimisation locale.

[0014] Selon un mode de réalisation, le RNCG a été adapté audit graphe lors d'une phase préalable d'adaptation, la phase préalable d'adaptation comprenant une procédure d'apprentissage permettant d'adapter des paramètres d'une phase de convolution et d'une phase de combinaison linéaire dudit RNCG, dits paramètres du RNCG, la procédure d'apprentissage étant une procédure itérative comprenant à chaque itération : générer une instance aléatoire du graphe ; appliquer la procédure utilisant le RNCG à l'instance aléatoire afin d'obtenir une première information représentative d'un ordre des nœuds obligatoires de l'instance aléatoire, dite première information ; générer une deuxième information représentative d'un ordre des nœuds obligatoires de l'instance aléatoire définie aléatoirement, dite deuxième information ; obtenir une première et une deuxième informations représentatives d'un ordre des nœuds obligatoires optimisées, dites première et deuxième informations optimisées, en appliquant la procédure d'optimisation locale respectivement à l'ordre des nœuds obligatoires correspondant à la première et à la deuxième informations; sélectionner l'ordre représenté par ladite première ou par ladite deuxième information optimisée permettant de minimiser un coût de transition global d'un chemin entre les nœuds de départ et d'arrivée de l'instance aléatoire ; former une nouvelle instance à partir de chaque nœud obligatoire de l'instance aléatoire, chaque nouvelle instance ayant pour nœud de départ le nœud obligatoire correspondant à la nouvelle instance, pour nœud d'arrivée le nœud d'arrivée de l'instance aléatoire et pour un ensemble de nœuds obligatoires, les nœuds suivant le nœud obligatoire utilisé comme nœud de départ de la nouvelle instance dans l'ordre sélectionné ; et pour l'instance aléatoire et chaque nouvelle instance, former un couple en associant ladite instance à un prochain nœud obligatoire à traverser pour ladite instance, ledit prochain nœud obligatoire à traverser étant le nœud obligatoire suivant le nœud de départ de ladite instance dans l'ordre sélectionné ; mettre à jour une fenêtre d'apprentissage avec les couples ainsi formés; faire évoluer les paramètres du RNCG pour minimiser une erreur de prédiction par le RNCG d'un prochain nœud obligatoire à traverser en utilisant le prochain nœud obligatoire de couples de la fenêtre d'apprentissage comme référence de prochains nœuds obligatoires à traverser à obtenir par le RNCG.

[0015] Selon un deuxième aspect de l'invention, l'invention concerne un dispositif pour définir un chemin à suivre par un véhicule dans un environnement représenté par un

graphe de nœuds reliés par des arêtes, chaque nœud dudit graphe représentant une position pouvant être prise par ledit véhicule, chaque arête entre deux nœuds étant associée à un coût de transition entre les deux nœuds. Le dispositif comprend : des moyens d'obtention pour obtenir une instance dudit graphe, dite instance à résoudre, chaque instance étant représentée par un nœud de départ, un nœud d'arrivée et un ensemble de nœuds obligatoires par lesquels le véhicule doit passer, lesdits nœuds de départ, d'arrivée et obligatoires formant les nœuds de l'instance à résoudre ; des moyens d'obtention pour obtenir une information représentative d'un ordre initial de parcours desdits nœuds obligatoires en appliquant une procédure utilisant un réseau de neurones convolutif de graphe, dit RNCG, adapté audit graphe à ladite instance à résoudre; des moyens d'exécution pour exécuter une procédure d'optimisation locale de l'ordre des nœuds obligatoires afin d'obtenir un ordre des nœuds obligatoires optimisé, ladite procédure étant initialisée avec ladite information représentative de l'ordre initial; des moyens de définition pour définir un chemin pour ladite instance à résoudre à partir de l'ordre des nœuds obligatoires optimisé et pour chaque paire de nœuds de ladite instance à résoudre, d'un plus court chemin entre les nœuds de ladite paire.

[0016] Selon un troisième aspect de l'invention, l'invention concerne un véhicule comprenant un dispositif selon le deuxième aspect.

[0017] Selon un quatrième aspect de l'invention, l'invention concerne un programme d'ordinateur, comprenant des instructions pour mettre en œuvre, par un dispositif, le procédé selon le premier aspect, lorsque ledit programme est exécuté par un processeur dudit dispositif.

[0018] Selon un cinquième aspect de l'invention, l'invention concerne des moyens de stockage, caractérisés en ce qu'ils stockent un programme d'ordinateur comprenant des instructions pour exécuter, par un dispositif, le procédé selon le premier aspect, lorsque ledit programme est exécuté par un processeur dudit dispositif.

Brève description des dessins

[0019] Les caractéristiques de l'invention mentionnées ci-dessus, ainsi que d'autres, apparaîtront plus clairement à la lecture de la description suivante d'un exemple de réalisation, ladite description étant faite en relation avec les dessins joints, parmi lesquels :

[0020] [fig.1] illustre un graphe représentatif d'un environnement dans lequel peut évoluer un véhicule ;

[0021] [fig.2] illustre schématiquement une architecture matérielle d'un module de traitement apte à mettre en œuvre l'invention ;

[0022] [fig.3] illustre schématiquement un procédé de définition d'un chemin d'un véhicule selon l'invention ;

[0023] [fig.4] illustre schématiquement une procédure d'application d'un réseau de neurones convolutif de graphe à une instance d'un graphe ;

[0024] [fig.5] illustre schématiquement une procédure d'apprentissage de paramètres d'un réseau de neurones convolutif de graphe ;

[0025] [fig.6] illustre schématiquement une procédure itérative de détermination d'un ordre de parcours de nœuds obligatoires pour une instance d'un graphe basée sur le réseau de neurones convolutif de graphe ; et,

[0026] [fig.7] illustre schématiquement une procédure d'optimisation locale d'un ordre des nœuds obligatoires.

[0027] EXPOSE DETAILLE DE MODES DE REALISATION

[0028] L'invention est décrite par la suite dans un contexte de véhicule autonome roulant circulant dans un environnement représenté par un graphe tel que le véhicule *V* dans le graphe *G* décrit en relation avec la Fig. 1, le graphe *G* n'étant pas orienté. Un graphe est dit orienté lorsqu'au moins un coût de transition entre deux nœuds du graphe dépend du sens de la transition entre les deux nœuds. L'invention est toutefois adaptée à d'autres types de véhicules autonomes ou semi-autonomes, tels qu'un drone volant ou naviguant sur ou sous l'eau, et évoluant dans un environnement pouvant être représenté par un graphe orienté ou pas.

[0029] Le procédé de définition d'un chemin à suivre par un véhicule dans son environnement se traduit par un procédé de résolution d'une instance d'un graphe. Comme décrit par la suite, ce procédé peut fonctionner en temps réel, ce procédé étant capable de résoudre une instance très rapidement et de fournir un chemin quasi-optimal au véhicule *V* en s'appuyant sur un réseau de neurones convolutif de graphe, dit *RNCG*. Le RNCG comprend des paramètres obtenus en utilisant une procédure d'apprentissage itérative pouvant être mise en œuvre pendant une phase non temps réelle, préalablement à tout déplacement du véhicule *V*.

[0030] Dans un mode de réalisation, le véhicule *V* comprend un module de traitement 10 exécutant le procédé de définition d'un chemin selon l'invention. Dans ce mode de réalisation, le module de traitement 10 exécute aussi la procédure d'apprentissage itérative permettant de déterminer les paramètres du réseau de neurones.

[0031] Dans d'autres modes de réalisation, le module de traitement 10 est un module externe au véhicule *V* qui reçoit une instance à résoudre et communique au véhicule *V* un résultat du procédé de définition d'un chemin pour ladite instance, c'est-à-dire un chemin à suivre par le véhicule *V*.

[0032] La **Fig. 2** illustre schématiquement un exemple d'une architecture matérielle du module de traitement 10.

[0033] Selon l'exemple d'architecture matérielle représenté à la Fig. 2, le module de traitement 10 comprend alors, reliés par un bus de communication 100 : un processeur

ou CPU (« Central Processing Unit » en anglais) 101 ; une mémoire vive RAM (« Random Access Memory » en anglais) 102 ; une mémoire morte ROM (« Read Only Memory » en anglais) 103 ; une unité de stockage telle qu'un disque dur ou un lecteur de support de stockage, tel qu'un lecteur de cartes SD (« Secure Digital » en anglais) 104 ; au moins une interface de communication 105 permettant par exemple au module de traitement 10 de communiquer des caractéristiques d'un chemin à suivre par le véhicule *V* à un module contrôlant les déplacements du véhicule *V*.

[0034] Le processeur 101 est capable d'exécuter des instructions chargées dans la RAM 102 à partir de la ROM 103, d'une mémoire externe (non représentée), d'un support de stockage (tel qu'une carte SD), ou d'un réseau de communication. Lorsque le module de traitement 10 est mis sous tension, le processeur 101 est capable de lire de la RAM 102 des instructions et de les exécuter. Ces instructions forment un programme d'ordinateur causant l'exécution complète ou partielle, par le processeur 101, du procédé décrit ci-après en relation avec la Fig. 3 et du procédé décrit en relation avec la Fig. 6.

[0035] Le procédé décrit en relation avec la Fig. 3 et le procédé décrit en relation avec la Fig. 6 peuvent être implémentés sous forme logicielle par exécution d'un ensemble d'instructions par une machine programmable, par exemple un DSP (« Digital Signal Processor » en anglais) ou un microcontrôleur, ou être implémentés sous forme matérielle par une machine ou un composant dédié, par exemple un FPGA (« Field-Programmable Gate Array » en anglais) ou un ASIC (« Application-Specific Integrated Circuit » en anglais).

[0036] La **Fig. 3** illustre schématiquement un procédé de définition d'un chemin d'un véhicule selon l'invention.

[0037] Dans une étape 30, le module de traitement 10 obtient une instance à résoudre $I_R(S_R, D_R, M_R)$ d'un graphe *G*. L'instance $I_R(S_R, D_R, M_R)$ est donc représentée par un nœud de départ S_R, un nœud d'arrivée D_R et un ensemble de nœuds obligatoires M_R. L'ensemble de nœuds obligatoires M_R peut être vide ou comprendre un ou plusieurs nœuds obligatoires. Le nœud de départ S_R, le nœud d'arrivée D_R et l'ensemble de nœuds obligatoires M_R forment les *nœuds de l'instance* $I_R(S_R, D_R, M_R)$.

[0038] Dans une étape 31, le module de traitement 10 obtient une information représentative d'un ordre initial de parcours des nœuds obligatoires de l'ensemble de nœuds obligatoires M_R entre le nœud de départ S_R et le nœud d'arrivée D_R en appliquant une procédure décrite en relation avec la Fig. 6 à l'instance $I_R(S_R, D_R, M_R)$. La procédure de la Fig. 6 utilise un RNCG adapté au graphe *G*. Une application dudit RNCG à une instance d'un graphe est décrite en relation avec la Fig. 4. L'adaptation

du RNCG au graphe *G* passe par une phase préliminaire d'adaptation. Cette phase préliminaire d'adaptation s'appuie sur une procédure d'apprentissage itérative décrite en relation avec la Fig. 5 durant laquelle des paramètres du RNCG sont déterminés. L'information représentative d'un ordre initial de parcours des nœuds obligatoires de l'ensemble de nœuds obligatoires M_R est une liste ordonnée de nœuds obligatoires.

[0039] Dans une étape 32, le module de traitement 10 exécute une procédure d'optimisation locale de l'ordre des nœuds obligatoires afin d'obtenir un ordre des nœuds obligatoires optimisé, ladite procédure étant initialisée avec l'information représentative de l'ordre initial obtenue lors de l'étape 31. La procédure d'optimisation locale de l'ordre des nœuds obligatoires est décrite en relation avec la Fig. 7.

[0040] Dans une étape 33, le module de traitement 10 définit un chemin pour l'instance $I_R(S_R, D_R, M_R)$ à partir de l'ordre des nœuds obligatoires optimisé et, pour chaque paire de nœuds de l'instance $I_R(S_R, D_R, M_R)$, d'un plus court chemin entre les nœuds de ladite paire.

[0041] La **Fig. 4** illustre schématiquement une procédure d'application d'un RNCG à une instance *I(S, D, M)* du graphe *G*.

[0042] Dans une étape 401, le module de traitement 10 encode une instance *I(S, D, M)* du graphe *G* sous forme d'un vecteur X_i à une dimension. Dans un mode de réalisation, lors de cet encodage, chaque nœud du graphe *G* est encodé suivant un nombre $NB_A = 3$ d'attributs (A1, A2, A3) :

[0043] - A1=1 si le nœud est un nœud de départ pour l'instance *I(S, D, M)*, A1=0 sinon ;

[0044] - A2=1 si le nœud est un nœud d'arrivée pour l'instance *I(S, D, M)*, A2=0 sinon ;

[0045] - A3=1 si le nœud est un nœud obligatoire pour l'instance *I(S, D, M)*, A3=0 sinon.

[0046] On suppose ici que les nœuds du graphe *G* sont ordonnés. A la fin de l'encodage, les nœuds du graphe *G* apparaissent dans le vecteur X_i dans cet ordre sous forme de triplets d'attributs (A1, A2, A3).

[0047] Dans une étape 402, le module de traitement 10 met le vecteur X_i sous forme d'une matrice MAT_i ayant un nombre de lignes égal au nombre de nœuds NB_G dans le graphe *G* et un nombre de colonnes égal au nombre NB_A d'attributs utilisés. On passe donc d'un vecteur à une dimension (X_i) à un vecteur à deux dimensions (MAT_i).

[0048] Dans une étape 403, le module de traitement 10 applique le RNCG à la matrice MAT_i.

[0049] Une application d'un réseau de neurones convolutif comprend en général une pluralité de phases dont les phases suivantes :

[0050] - Une phase de convolution ou plusieurs phases de convolution consécutives ;

[0051] - Une phase de vectorisation (mise sous forme de vecteur) des données de sortie de la

ou des phases de convolution ;

[0052] - Une phase de combinaison linéaire, dite « fully connected ».

[0053] Chaque phase de convolution se décompose généralement en trois sous-phases :

[0054] - une sous-phase de convolution par un noyau de convolution de données d'entrée permettant d'obtenir un ensemble de caractéristiques représentatives desdites données d'entrées. Les caractéristiques obtenues ne sont pas prédéfinies mais apprises par le réseau de neurones lors d'une phase d'entraînement. Pendant la phase d'entraînement, le noyau de convolution évolue de manière à « apprendre » à extraire des caractéristiques pertinentes pour un problème donné.

[0055] - une sous-phase de « mise en commun » (« pooling » en terminologie anglo-saxonne) ayant pour but de réduire la taille des données qu'elle reçoit en entrée (issue de la convolution), tout en préservant les caractéristiques importantes de ces données d'entrées. La phase de « mise en commun » permet de réduire le nombre de calculs dans le réseau de neurones convolutif.

[0056] - une sous-phase d'application d'une fonction d'activation consistant à appliquer une fonction mathématique, dite *fonction d'activation*, aux données obtenues en sortie de chaque convolution (ou de la phase de mise en commun s'il y en a une). Le terme de *fonction d'activation* vient d'un équivalent biologique *potentiel d'activation*, qui représente un seuil de stimulation qui, une fois atteint, entraîne une réponse d'un neurone.

[0057] Lorsqu'un réseau de neurones est utilisé dans une méthode de classification ou de prédiction de valeurs, la phase de combinaison linéaire constitue toujours la dernière phase d'un réseau de neurones, qu'il soit convolutif ou non. Cette phase reçoit un vecteur en entrée, dit *vecteur d'entrée*, et produit un nouveau vecteur en sortie, dit *vecteur de sortie*. Pour cela, elle applique une combinaison linéaire aux composantes du vecteur d'entrée. La phase de combinaison linéaire permet de classifier les données d'entrées du réseau de neurones suivant un nombre de classes prédéfinies N. Elle renvoie donc un vecteur de sortie de taille N. Chaque composante du vecteur de sortie est associée à une classe et est représentative d'une probabilité pour les données d'entrées du réseau de neurones d'appartenir à ladite classe. Chaque composante du vecteur d'entrée peut contribuer différemment au vecteur de sortie. Pour ce faire, lors de l'application de la combinaison linéaire, un poids différent est appliqué à chaque composante en fonction d'une importance qu'on souhaite donner à la caractéristique que cette composante représente. La combinaison linéaire de la phase de combinaison linéaire est généralement suivie d'une phase de transformation du vecteur de sortie en distribution de probabilité. Le réseau de neurones convolutif apprend les valeurs des poids de la phase de combinaison linéaire de la même manière qu'il apprend à faire évoluer le noyau de convolution de chaque phase de convolution.

[0058] Comme nous le décrivons par la suite, le RNCG utilisé dans l'invention applique une pluralité de phases de convolution à un graphe. Lesdites phases de convolution ne comprennent pas de sous-phase de « mise en commun » et comprennent une sous-phase d'application d'une fonction d'activation mise en œuvre par la fonction ReLU() telle que ReLU(x)=max (0, x).

[0059] Dans un mode de réalisation, le noyau de convolution utilisé dans l'invention est tiré de l'article *Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. ICLR 2017.* Cet article propose une relation combinant convolution et application d'une fonction d'activation.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

[0060] $H^{(l+1)}$ représente des données de sortie issues d'une application d'une convolution par un noyau de convolution et d'une fonction d'activation à des données d'entrées $H^{(l)}$. $\tilde{A} = A + I$, A étant une matrice d'adjacence du graphe G et I étant une matrice d'identité. $\tilde{D}$ est une matrice de degrés de $\tilde{A}$ c'est-à-dire une matrice diagonale telle que $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $W^{(l)}$ est une matrice de poids spécifique à une phase de convolution l. Ces poids évoluent pendant la phase d'entraînement de manière à apprendre au réseau de neurones à extraire des caractéristiques pertinentes de chaque instance du graphe G. La phase préliminaire d'adaptation au graphe G utilisant la procédure d'apprentissage itérative décrite en relation avec la Fig. 5 a pour but de déterminer les poids (ou paramètres) de chaque matrice $W^{(l)}$. σ est une fonction d'activation. Dans un mode de réalisation, la fonction d'activation σ utilisée est la fonction ReLU() telle que ReLU(x)=max (0, x).

[0061] Dans un mode de réalisation, le RNCG appliqué lors de l'étape 403 est multicouches, c'est-à-dire qu'il comprend plusieurs phases de convolution. Par exemple, le RNCG utilise « 4 » phases de convolution.

[0062] Chaque phase de convolution du RNCG reçoit en entrée une matrice, dite matrice d'entrée, de *NBC_IN* colonnes et de *NBL_IN* lignes et génère en sortie une matrice, dite matrice de sortie, de *NBL_OUT* lignes et *NBC_OUT* colonnes. La matrice d'entrée $H^{(0)}$ de la première phase de convolution du RNCG est la matrice MAT_i. Lorsque deux phases de convolution de RNCG se suivent, la matrice de sortie de la première phase de convolution devient la matrice d'entrée de la phase de convolution suivante. Chaque composante d'une matrice de sortie est représentative de caractéristiques des composantes de la matrice d'entrée (et donc de la matrice MAT_i). Chaque matrice d'entrée (respectivement chaque matrice de sortie) comprend *NBL_IN*=NB_G lignes (respectivement *NBL_OUT*=NB_G lignes). Chaque matrice

de sortie (et donc respectivement chaque matrice d'entrée à partir de la deuxième phase de convolution du RNCG) peut comprendre un nombre de colonnes *NBC_OUT* (respectivement *NBC_IN*) fixe ou variable d'une phase de convolution à l'autre. Dans un mode de réalisation *NBC_OUT*=10.

[0063] On considère ici que chaque phase de convolution classifie les données d'entrée en un nombre de classes égal au nombre de composantes de la matrice de sortie (c'est-à-dire en un nombre de classes égal à $NBL\_OUT \times NBC\_OUT$).

[0064] Dans une étape 404, le module de traitement 10 met la matrice de sortie issue de la dernière phase de convolution du RNCG sous forme d'un vecteur Y_i de $NBL\_OUT \times NBC\_OUT$ composantes en mettant bout à bout chaque ligne de ladite matrice de sortie.

[0065] Dans une étape 405, le module de traitement 10 applique une phase de combinaison linéaire (« fully connected » en terminologie anglo-saxonne) au vecteur Y_i. La phase de combinaison linéaire génère un vecteur O_i de composantes O_{ij} (*j* allant de « 1 » à NB_G) à partir du vecteur Y_i. Chaque composante O_{ij} est calculée de la manière suivante :

$$O_{ij} = Y_i.W_{ij} + b_{ij}$$

[0066] où W_{ij} est un poids d'un vecteur W_i et b_{ij} est un biais d'un vecteur de biais b_i et $Y_i.W_{ij}$ est un produit scalaire. Chaque nœud du graphe G est associé à une composante du vecteur O_i, chaque composante étant un nombre réel représentatif d'une probabilité que le nœud correspondant soit le prochain nœud obligatoire de l'ensemble de nœuds obligatoires *M* de l'instance *I(S, D, M)* devant être traversé par le véhicule *V* après avoir quitté le nœud de départ *S* de l'instance *I(S, D, M)*.

[0067] Dans une étape 406, le module de traitement 10 transforme le vecteur O_i en un vecteur de probabilité P_i. Pour ce faire, dans un mode de réalisation, une fonction exponentielle normalisée (« Softmax function » en terminologie anglo-saxonne) est appliquée à chaque composante du vecteur O_i. Une fonction exponentielle normalisée prend en entrée un vecteur *z* de *K* nombres réels z_j et génère en sortie un vecteur $\Omega(z)$ de *K* nombres réels strictement positifs et de somme « 1 » tel que :

$$\Omega(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \forall j \in \{1, \ldots, K\}$$

[0068] En sortie du RNCG *(i.e.* en sortie de l'étape 406), le module de traitement 10 obtient pour chaque nœud du graphe G, une probabilité que ledit nœud soit le prochain nœud obligatoire de l'ensemble de nœuds obligatoires *M* de l'instance *I(S, D, M)* devant être

traversé par le véhicule *V* après avoir quitté le nœud de départ *S* de l'instance *I(S, D, M)*.

[0069] Dans un mode de réalisation, l'instance *I(S, D, M)* est encodée différemment, par exemple, avec un nombre NB_A d'attributs différent, ou avec un nombre de lignes NB_G différent, par exemple, en ne faisant apparaître que les nœuds de départ, d'arrivée et obligatoires dans le vecteur X_i.

[0070] Dans un mode de réalisation, le module de traitement 10 encode directement l'instance *I(S, D, M)* sous forme de matrice MAT_i.

[0071] La **Fig. 5** illustre schématiquement une procédure d'apprentissage de paramètres d'un réseau de neurones convolutif de graphe.

[0072] Dans une étape 501, le module de traitement 10 génère une fenêtre d'apprentissage *F* . La fenêtre d'apprentissage *F* comprend des couples, la fenêtre pouvant comprendre un nombre de couples limité à un nombre prédéfini *q*. Chaque couple associe une instance à un prochain nœud obligatoire à visiter. Dans un mode de réalisation, la fenêtre d'apprentissage *F* est initialisée à l'ensemble vide et *q*=50000.

[0073] Dans une étape 502, le module de traitement 10 génère une instance aléatoire *I(S,D ,M)*.

[0074] Dans une étape 503, le module de traitement 10 applique le procédé décrit en relation avec la Fig. 6 à l'instance aléatoire *I(S,D,M)* afin d'obtenir une liste ordonnée LO_{RNCG} de nœuds obligatoires à traverser par le véhicule *V* entre le nœud de départ *S* et le nœud d'arrivée *D*. La liste ordonnée LO_{RNCG} est représentative d'un premier ordre de parcours des nœuds obligatoires. Le procédé décrit en relation avec la Fig. 6 fait intervenir le RNCG pour définir la liste ordonnée LO_{RNCG}.

[0075] L'étape 503 est suivie par une étape 504 au cours de laquelle le module de traitement 10 génère une liste ordonnée LO_{RAND} pour l'instance aléatoire *I(S,D,M)* dans laquelle l'ordre des nœuds obligatoires de l'ensemble de nœuds obligatoires *M* est défini aléatoirement. La liste ordonnée LO_{RAND} est représentative d'un deuxième ordre de parcours des nœuds obligatoires.

[0076] Dans une étape 505, le module de traitement 10 obtient une liste ordonnée optimisée LO_{RNCG}^{opt} en partant de la liste ordonnée LO_{RNCG}. Pour ce faire, le module de traitement 10 utilise par exemple la procédure d'optimisation locale de l'ordre des nœuds obligatoires que nous décrivons par la suite en relation avec la Fig. 7. Lors de l'étape 505, le module de traitement 10 calcule un coût de transition global C_{RNCG}^{opt} du chemin entre le nœud de départ et le nœud d'arrivée de l'instance aléatoire *I(S,D,M)* générée lors de l'étape 502. Le coût de transition global d'un chemin entre un nœud de départ et un nœud d'arrivée d'une instance d'un graphe dépend de l'ordre de parcours

des nœuds obligatoires de cette instance. Le coût de transition global d'un chemin entre un nœud de départ et un nœud d'arrivée d'une instance d'un graphe est une somme de coûts de plus court chemin entre chaque paire de nœuds successifs de l'instance formant ledit chemin. Le calcul du coût de transition global C_{RNCG}^{opt} du chemin s'appuie sur un tableau précalculé *TAB* comprenant pour chaque paire de nœuds du graphe *G*, un plus court chemin reliant les deux nœuds de ladite paire et un coût dudit plus court chemin. Chaque coût de plus court chemin est calculé en utilisant le coût de transition associé à chaque arête du graphe *G* utilisée pour former ledit plus court chemin. Le plus court chemin entre chaque paire de nœuds du graphe *G* a par exemple été déterminé en utilisant un algorithme de Dijkstra décrit dans le document *Dijkstra, E. W., « A note on two problems in connexion with graphs », Numerische Mathematik, vol. 1, 1959, p. 269–271*. La liste ordonnée LO_{RNCG}^{opt} est représentative d'un premier ordre de parcours optimisé des nœuds obligatoires.

[0077] Dans une étape 506, le module de traitement 10 obtient une liste ordonnée optimisée LO_{RAND}^{opt} en partant de la liste ordonnée aléatoire LO_{RAND} obtenue à l'étape 504. Pour ce faire, le module de traitement 10 utilise par exemple la procédure d'optimisation locale de l'ordre des nœuds obligatoires que nous décrivons par la suite en relation avec la Fig. 7. Lors de l'étape 506, le module de traitement 10 calcule un coût de transition global C_{RAND}^{opt} du chemin entre le nœud de départ et le nœud d'arrivée de l'instance aléatoire *I(S,D,M)* générée lors de l'étape 502. Comme lors de l'étape 505, le calcul du coût de transition global C_{RAND}^{opt} du chemin s'appuie sur le tableau précalculé *TAB*. La liste ordonnée LO_{RAND}^{opt} est représentative d'un deuxième ordre de parcours optimisé des nœuds obligatoires.

[0078] Dans une étape 507, le module de traitement 10 compare le coût C_{RNCG}^{opt} au coût C_{RAND}^{opt}. Si le coût C_{RNCG}^{opt} est supérieur au coût C_{RAND}^{opt}, le module de traitement 10 sélectionne la liste ordonnée LO_{RAND}^{opt} dans une étape 508. Sinon, le module de traitement 10 sélectionne la liste ordonnée LO_{RNCG}^{opt} dans une étape 509. La liste ordonnée sélectionnée est alors notée LO_{SELECT}^{opt}.

[0079] Dans une étape 510, le module de traitement 10 initialise une variable *j* à l'unité.

[0080] Dans une étape 511, le module de traitement initialise une instance temporaire *i(s,d,m)* à la valeur de l'instance aléatoire *I(S,D,M)* générée lors de l'étape 502.

[0081] Dans une étape 512, le module de traitement 10 identifie le nœud obligatoire m_j de l'ensemble de nœuds obligatoires *m* en position *j* dans la liste ordonnée sélectionnée

LO_{SELECT}^{opt}. Lors du premier passage par l'étape 512, la variable *j* étant égale à « 1 », le module de traitement 10 choisit le nœud obligatoire m_1 situé en première position dans la liste ordonnée sélectionnée LO_{SELECT}^{opt}.

[0082] Dans une étape 513, le module de traitement 10 sauvegarde le couple formé par l'instance temporaire *i(s,d,m)* et le nœud obligatoire m_j en dernière position de la fenêtre d'apprentissage *F*. Si avant la sauvegarde dudit couple dans la fenêtre d'apprentissage *F*, la fenêtre contenait *q* couples, le couple situé en première position de la fenêtre d'apprentissage *F* est supprimé de ladite fenêtre. Ainsi, la fenêtre d'apprentissage *F* comprend toujours un nombre *q* de couples.

[0083] Dans une étape 514, le module de traitement 10 donne au nœud de départ *s* de l'instance temporaire *i(s,d,m)* la valeur m_j et retire le nœud obligatoire m_j de l'ensemble de nœuds obligatoires *m*.

[0084] Dans une étape 515, le module de traitement 10 détermine si l'ensemble de nœuds obligatoires *m* est vide.

[0085] Si l'ensemble de nœuds obligatoires *m* n'est pas vide, le module de traitement 10 incrémente la variable *j* d'une unité dans une étape 516 et retourne à l'étape 512.

[0086] Si l'ensemble de nœuds obligatoires *m* est vide, le module de traitement 10 exécute une étape 517. Lors de l'étape 517, le module de traitement 10 lance une procédure d'entraînement itérative du RNCG. La procédure d'entraînement itérative comprend K_{ITER} itérations. Dans un mode de réalisation, K_{ITER}=10000. A chaque itération, le module de traitement 10 sélectionne aléatoirement un nombre B_c de couples associant une instance à un prochain nœud obligatoire à visiter dans la fenêtre d'apprentissage *F*. Dans un mode de réalisation, le nombre de couples B_c=32. Pour chaque couple sélectionné, le module de traitement 10 applique le RNCG à l'instance représentée dans le couple tel que décrit en relation avec la Fig. 4 en ajustant (*i.e.* en apprenant selon la terminologie des réseaux de neurones) des paramètres du RNCG afin que le nœud obligatoire du couple soit associé à la probabilité la plus élevée d'être le prochain nœud obligatoire à visiter. Les paramètres entraînés sont les poids de chaque matrice de poids $W^{(l)}$ utilisée dans le RNCG et les poids utilisés lors de la phase de combinaison linéaire de l'étape 405. Chaque itération de la procédure d'entraînement des paramètres du RNCG comprend une minimisation d'une fonction de coût $H(\theta)$ par descente stochastique de gradient, θ étant l'ensemble des paramètres du RNCG à apprendre.

$$H(\theta) = \frac{1}{B_c} \sum_{i=1}^{B_c} \sum_{j=1}^{NB_G} -t_{i,j} . \log\left(f(X_i, \theta)_j\right)$$

[0087] où NB_G est le nombre de nœuds dans le graphe *G*, $t_{i,j}$ est une variable qui vaut « 1 » si pour l'instance numéro *i*, le prochain nœud obligatoire à visiter est le nœud du graphe *G* d'indice *j*, et « 0 » sinon. X_i est le vecteur résultant d'un encodage de l'instance numéro *i* tel que décrit en relation avec l'étape 401. L'instance numéro *i* est une instance correspondant à un des B_c couples sélectionnés aléatoirement dans la fenêtre d'apprentissage *F*, le prochain nœud obligatoire à visiter étant le nœud associé à l'instance numéro *i* dans ledit couple. Les instances de la fenêtre d'apprentissage *F* sont des instances considérées comme résolues sur lesquelles la procédure d'apprentissage s'appuie pour apprendre les paramètres du réseau de neurones. La composante $\sum_{j=1}^{NB_G} -t_{i,j}.\log\left(f(X_i,\theta)_j\right)$ de la fonction de coût $H(\theta)$ représente une erreur de prédiction par le RNCG du prochain nœud obligatoire à traverser correspondant à l'instance numéro *i*. la fonction de coût $H(\theta)$ est donc une moyenne d'erreurs de prédiction par le RNCG de prochains nœuds à traverser pour un ensemble de B_c instances. Lors de la première itération de la procédure d'apprentissage de paramètres du RNCG (et donc lors de la première itération de la procédure d'entraînement des paramètres du RNCG), le module de traitement 10 utilise des paramètres du RNCG déterminés aléatoirement pour le démarrage de la descente stochastique de gradient. Lors des itérations suivantes de la procédure d'apprentissage des paramètres du RNCG ou de la procédure d'entraînement des paramètres du RNCG, le module de traitement 10 initialise la descente stochastique de gradient avec les paramètres du RNCG déterminés lors de la dernière exécution de la procédure d'entraînement des paramètres du RNCG.

[0088] L'étape 517 est suivie d'une étape 518 au cours de laquelle le module de traitement 10 détermine si une condition d'arrêt de la procédure d'apprentissage des paramètres du RNCG est respectée. Dans une mode de réalisation, la condition d'arrêt est une durée maximum d'exécution de la procédure d'apprentissage de paramètres du RNCG. Si à l'étape 518 cette durée maximum est atteinte, la procédure d'apprentissage des paramètres du RNCG s'arrête lors d'une étape 519. Sinon, le module de traitement 10 retourne à l'étape 502. Dans un mode de réalisation, la durée maximum est égale à « 24 » heures.

[0089] A la fin de la procédure d'apprentissage de paramètres du RNCG, le RNCG est considéré comme entraîné.

[0090] La **Fig. 6** illustre schématiquement une procédure itérative de détermination d'un ordre de parcours de nœuds obligatoires pour une instance d'un graphe basée sur le RNCG décrit en relation avec la Fig. 4. Le procédé de la Fig. 6 est utilisé dans la procédure d'apprentissage décrite en relation avec la Fig. 5, mais aussi, une fois l'apprentissage effectué, pour déterminer un ordre des nœuds obligatoires pour une

instance lors de l'étape 31.

[0091] Dans une étape 3100, le module de traitement 10 obtient une instance courante à résoudre $I_c(S_c, D_c, M_c)$. Lors de l'étape 503, l'instance courante $I_c(S_c, D_c, M_c) = I(S, D, M)$. Lors de l'étape 31, l'instance courante $I_c(S_c, D_c, M_c) = I_R(S_R, D_R, M_R)$.

[0092] Dans une étape 3101, le module de traitement 10 détermine si l'ensemble de nœuds obligatoires M_c est vide. Si c'est le cas, l'étape 3101 est suivie d'une étape 3102 au cours de laquelle le module de traitement 10 génère une liste ordonnée LO_{RNCG} vide et met fin à l'étape 503.

[0093] Si l'ensemble de nœuds obligatoires M_c n'est pas vide, le module de traitement 10 détermine lors d'une étape 3103, si l'ensemble de nœuds obligatoires M_c contient un nœud obligatoire (*i.e.* si le cardinal de l'ensemble de nœuds obligatoires M_c est égal à « 1 »). Si l'ensemble de nœuds obligatoires M_c ne contient qu'un nœud obligatoire, le module de traitement 10 génère une liste ordonnée LO_{RNCG} contenant uniquement ce nœud obligatoire et met fin à l'étape 503.

[0094] Si l'ensemble de nœuds obligatoires M_c contient une pluralité de nœuds obligatoires, le module de traitement 10 initialise une instance temporaire $i_c(s_c, d_c, m_c)$ à la valeur de l'instance courante $I_c(S_c, D_c, M_c)$ lors d'une étape 3105.

[0095] Dans une étape 3106, le module de traitement 10 initialise une variable *j* à l'unité.

[0096] Dans une étape 3107, le module de traitement 10 applique le RNCG décrit en Fig. 4 à l'instance temporaire $i_c(s_c, d_c, m_c)$. L'application du RNCG à l'instance temporaire $i_c(s_c, d_c, m_c)$ permet d'obtenir un vecteur de probabilité associant à chaque nœud du graphe *G* une valeur de probabilité.

[0097] Dans une étape 3108, le module de traitement 10 détermine le nœud obligatoire m_{c_j} de l'ensemble de nœuds obligatoires m_c ayant la probabilité la plus élevée d'être le prochain nœud obligatoire à traverser à la suite du nœud de départ s_c. Le module de traitement 10 insère alors le nœud obligatoire m_{c_j} dans la liste ordonnée LO_{RNCG} des nœuds obligatoires dans une étape 3109.

[0098] Dans une étape 3110, le module de traitement 10 donne au nœud de départ s_c de l'instance temporaire $i_c(s_c, d_c, m_c)$ la valeur du nœud obligatoire m_{c_j} et retire le nœud obligatoire m_{c_j} de l'ensemble de nœuds obligatoires m_c.

[0099] Dans une étape 3111, le module de traitement 10 détermine si le nombre de nœuds obligatoires restant dans l'ensemble de nœuds obligatoires m_c est égal à l'unité. Si

c'est le cas, le module de traitement 10 ajoute le nœud obligatoire restant à la fin de la liste ordonnée LO_{RNCG} dans une étape 3113. A cette étape, tous les nœuds obligatoires de l'instance $I_c(S_c, D_c, M_c)$ apparaissent dans la liste ordonnée LO_{RNCG} dans l'ordre proposé par le RNCG. La liste ordonnée LO_{RNCG} est alors une information représentative d'un ordre de parcours de nœuds obligatoires de l'ensemble de nœuds obligatoires M_c de l'instance $I_c(S_c, D_c, M_c)$.

[0100] Si le nombre de nœuds obligatoires restant dans l'ensemble de nœuds obligatoires m_c est supérieur à l'unité, le module de traitement 10, incrémente la variable *j* d'une unité lors d'une étape 3112 et retourne à l'étape 3107. Le module de traitement 10 repart donc avec une nouvelle instance ayant pour nœud de départ s_c le nœud obligatoire m_{c_j} déterminé lors de l'itération précédente, pour nœud d'arrivée, le même nœud d_c que dans l'itération précédente et une liste de nœuds obligatoires m_c identique à la précédente de laquelle a été supprimée m_{c_j}.

[0101] La **Fig. 7** illustre schématiquement un exemple de procédure d'optimisation locale de l'ordre des nœuds obligatoires. La procédure de la Fig. 7 est donc une procédure d'optimisation locale. L'initialisation de cette procédure par un ordre de parcours de nœuds obligatoires obtenu par un réseau de neurones adapté au graphe *G*, permet d'obtenir quasiment systématiquement un ordre de parcours des nœuds obligatoires meilleur qu'en initialisant ladite procédure avec un ordre de parcours de nœuds obligatoires définis aléatoirement.

[0102] La procédure d'optimisation locale de l'ordre des nœuds obligatoires est utilisée lors des étapes 505, 506 et 32. Cette procédure reçoit en entrée une instance $I^{opt}(S^{opt}, D^{opt}, M^{opt})$ et une liste ordonnée à optimiser LO^{opt}.

[0103] Dans une étape 3200, le module de traitement 10 crée une liste ordonnée variable lo^{opt}.

[0104] Dans une étape 3201, le module de traitement 10 calcule un coût de transition global C^{opt} du chemin entre le nœud de départ S^{opt} et le nœud d'arrivée D^{opt} de l'instance aléatoire $I^{opt}(S^{opt}, D^{opt}, M^{opt})$ en prenant en compte l'ordre des nœuds obligatoires de la liste ordonnée LO^{opt}. Le calcul du coût de transition global C^{opt} du chemin s'appuie sur le tableau précalculé *TAB*.

[0105] Dans une étape 3202, le module de traitement 10 change l'ordre dans la liste ordonnée LO^{opt} et sauvegarde la liste ordonnée obtenue dans la liste ordonnée variable lo^{opt}. Lors de l'étape 3202, le module de traitement 10 détermine une paire d'éléments, dits éléments pivots, dans la liste ordonnée LO^{opt} et intervertit les

éléments de la liste ordonnée LO^{opt} situés entre les deux éléments pivots de manière symétrique. La paire d'éléments pivots déterminée est différente de toutes paires d'éléments pivots déjà utilisées pour la liste ordonnée LO^{opt}. A partir de la liste ordonnée LO^{opt} on obtient alors une liste ordonnée variable lo^{opt} dans laquelle les éléments de la liste ordonnée LO^{opt} situés entre les deux éléments pivots ont été intervertis symétriquement.

[0106] Dans une étape 3203, le module de traitement 10 calcule un coût de transition global c^{opt} du chemin entre le nœud de départ S^{opt} et le nœud d'arrivée D^{opt} de l'instance aléatoire $I^{opt}(S^{opt}, D^{opt}, M^{opt})$ en prenant en compte l'ordre des nœuds obligatoires de la liste ordonnée variable lo^{opt} obtenue lors de l'étape 3202.

[0107] Dans une étape 3204, le module de traitement 10 compare le coût de transition global c^{opt} au coût de transition global C^{opt}. Si $c^{opt}<C^{opt}$, lors d'une étape 3206, le module de traitement 10 modifie la liste ordonnée LO^{opt} de la manière suivante :

$$LO^{opt} = lo^{opt}$$

[0108] On obtient alors une nouvelle liste ordonnée LO^{opt} pour laquelle aucune paire d'éléments pivots n'a été testée. L'étape 3206 est alors suivie de l'étape 3200.

[0109] Sinon, le module de traitement 10 vérifie lors d'une étape 3205 si toutes les paires d'éléments pivots possibles de la liste ordonnée LO^{opt} ont été testées. Si toutes les paires d'éléments pivots possibles de la liste ordonnée LO^{opt} ont été testées, le module de traitement 10 met fin à la procédure itérative d'optimisation d'un ordre de nœuds obligatoires lors d'une étape 3207. Sinon, le module de traitement 10 revient à l'étape 3202.

[0110] Dans un mode de réalisation, lors de l'étape 3205, le module de traitement 10, en plus de tester si toutes les paires d'éléments pivots possibles de la liste ordonnée LO^{opt} ont été testées, teste une condition d'arrêt. La condition d'arrêt est ici une durée prédéterminée d'exécution de la procédure itérative d'optimisation d'un ordre de nœuds obligatoires. Dans un mode de réalisation, cette durée prédéterminée est égale à « 60 » secondes. La procédure d'optimisation d'un ordre de nœuds obligatoires s'arrête lorsque la durée prédéterminée est atteinte. A la fin de cette procédure, la dernière liste ordonnée LO^{opt} obtenue est la liste ordonnée optimisée.

Revendications

[Revendication 1] Procédé de définition d'un chemin à suivre par un véhicule dans un environnement représenté par un graphe de nœuds reliés par des arêtes, chaque nœud dudit graphe représentant une position pouvant être prise par ledit véhicule, chaque arête entre deux nœuds étant associée à un coût de transition entre les deux nœuds, caractérisé en ce qu'il comprend :

- obtenir (30) une instance dudit graphe, dite instance à résoudre, chaque instance étant représentée par un nœud de départ, un nœud d'arrivée et un ensemble de nœuds obligatoires par lesquels le véhicule doit passer, lesdits nœuds de départ, d'arrivée et obligatoires formant les nœuds de l'instance à résoudre ;
- obtenir (31) une information représentative d'un ordre initial de parcours desdits nœuds obligatoires en appliquant une procédure utilisant un réseau de neurones convolutif de graphe, dit RNCG, adapté audit graphe à ladite instance à résoudre ;
- exécuter (32) une procédure d'optimisation locale de l'ordre des nœuds obligatoires afin d'obtenir un ordre des nœuds obligatoires optimisé, ladite procédure étant initialisée avec ladite information représentative de l'ordre initial ;
- définir (33) un chemin pour ladite instance à résoudre à partir de l'ordre des nœuds obligatoires optimisé et pour chaque paire de nœuds de ladite instance à résoudre, d'un plus court chemin entre les nœuds de ladite paire.

[Revendication 2] Procédé selon la revendication 1, caractérisé en ce que la procédure utilisant le RNCG est une procédure itérative prenant une instance du graphe en entrée, dite instance d'entrée et générant une information représentative d'un ordre initial de parcours des nœuds obligatoires de l'instance d'entrée en sortie, dite information de sortie, et comprenant à chaque itération:

- appliquer (3107) le RNCG à ladite instance d'entrée afin d'obtenir pour chaque nœud du graphe une valeur représentative d'une probabilité d'être le prochain nœud obligatoire à traverser à la suite du nœud de départ de l'instance d'entrée ;
- identifier (3108) le nœud obligatoire de l'ensemble de nœuds obligatoires de l'instance d'entrée possédant la probabilité la plus élevée ;
- insérer (3109) le nœud obligatoire identifié dans une liste ordonnée de

nœuds obligatoires ;
- remplacer (3110) le nœud de départ de l'instance d'entrée par le nœud obligatoire identifié et retirer le nœud obligatoire identifié de l'ensemble de nœuds obligatoires de l'instance d'entrée ;
- les itérations prenant fin lorsqu'un nombre de nœuds dans l'ensemble de nœuds obligatoires de l'instance d'entrée est égal à un, le nœud obligatoire restant dans l'ensemble de nœuds obligatoires de l'instance d'entrée étant inséré dans ladite liste ordonnée et ladite liste ordonnée formant l'information de sortie.

[Revendication 3] Procédé selon la revendication 1 ou 2, caractérisé en ce que la procédure d'optimisation locale de l'ordre des nœuds obligatoires est une procédure itérative prenant une instance, dite instance initiale, et une information représentative d'un ordre de nœuds obligatoires, dit ordre initial, en entrée et générant une information représentative d'un ordre de nœuds obligatoires optimisé en sortie, dit ordre optimisé,
- chaque information représentative d'un ordre de nœuds obligatoires étant une liste de nœuds obligatoires ordonnés, et comprend à chaque itération :
- déterminer (3202) une paire de nœuds obligatoires, dits nœuds pivots, parmi les nœuds obligatoires de la liste de nœuds obligatoires ordonnée de l'ordre initial, dite liste initiale, et obtenir une liste de nœuds obligatoires ordonnée modifiée, dite liste modifiée, dans laquelle les nœuds obligatoires de la liste initiale situés entre les deux nœuds pivots ont été intervertis symétriquement, la paire de nœuds pivots déterminée étant différente de toutes paires de nœuds pivots déjà utilisées pour modifier l'ordre des nœuds obligatoires dans la liste initiale ;
- définir (3204) une information représentative de l'ordre des nœuds obligatoires dans la liste modifiée comme étant un ordre optimisé temporaire lorsqu'un coût de transition global (3203) d'un chemin entre les nœuds de départ et d'arrivée de l'instance initiale en suivant l'ordre des nœuds obligatoires dans la liste modifiée est inférieur à un coût de transition global dudit chemin en suivant l'ordre des nœuds obligatoires dans la liste initiale; et définir (3206) l'ordre optimisé temporaire comme étant l'ordre initial ; et,
- mettre fin (3207) aux itérations de la procédure d'optimisation locale lorsqu'une condition d'arrêt est remplie (3205), l'ordre optimisé étant le dernier ordre optimisé temporaire obtenu par la procédure d'optimisation locale.

[Revendication 4] Procédé selon la revendication 1, 2 ou 3, caractérisé en ce que le RNCG a été adapté audit graphe lors d'une phase préalable d'adaptation, la phase préalable d'adaptation comprenant une procédure d'apprentissage permettant d'adapter des paramètres d'une phase de convolution et d'une phase de combinaison linéaire dudit RNCG, dits paramètres du RNCG, la procédure d'apprentissage étant une procédure itérative comprenant à chaque itération :

- générer (502) une instance aléatoire du graphe ;
- appliquer (503) la procédure utilisant le RNCG à l'instance aléatoire afin d'obtenir une première information représentative d'un ordre des nœuds obligatoires de l'instance aléatoire, dite première information ;
- générer (504) une deuxième information représentative d'un ordre des nœuds obligatoires de l'instance aléatoire défini aléatoirement, dite deuxième information ;
- obtenir une première et une deuxième informations représentatives d'un ordre des nœuds obligatoires optimisées, dites première et deuxième informations optimisées, en appliquant la procédure d'optimisation locale respectivement à l'ordre des nœuds obligatoires correspondant à la première et à la deuxième informations ;
- sélectionner (507, 508, 509) l'ordre représenté par ladite première ou par ladite deuxième information optimisée permettant de minimiser un coût de transition global d'un chemin entre les nœuds de départ et d'arrivée de l'instance aléatoire ;
- former une nouvelle instance à partir de chaque nœud obligatoire de l'instance aléatoire, chaque nouvelle instance ayant pour nœud de départ le nœud obligatoire correspondant à la nouvelle instance, pour nœud d'arrivée le nœud d'arrivée de l'instance aléatoire et pour ensemble de nœuds obligatoires, les nœuds suivant le nœud obligatoire utilisé comme nœud de départ de la nouvelle instance dans l'ordre sélectionné ; et pour l'instance aléatoire et chaque nouvelle instance, former un couple en associant ladite instance à un prochain nœud obligatoire à traverser pour ladite instance, ledit prochain nœud obligatoire à traverser étant le nœud obligatoire suivant le nœud de départ de ladite instance dans l'ordre sélectionné ;
- mettre à jour une fenêtre d'apprentissage avec les couples ainsi formés ;
- faire évoluer les paramètres du RNCG pour minimiser une erreur de prédiction par le RNCG d'un prochain nœud obligatoire à traverser en

utilisant le prochain nœud obligatoire de couples de la fenêtre d'apprentissage comme référence de prochains nœuds obligatoires à traverser à obtenir par le RNCG.

[Revendication 5] Dispositif pour définir un chemin à suivre par un véhicule dans un environnement représenté par un graphe de nœuds reliés par des arêtes, chaque nœud dudit graphe représentant une position pouvant être prise par ledit véhicule, chaque arête entre deux nœuds étant associée à un coût de transition entre les deux nœuds, caractérisé en ce qu'il comprend :

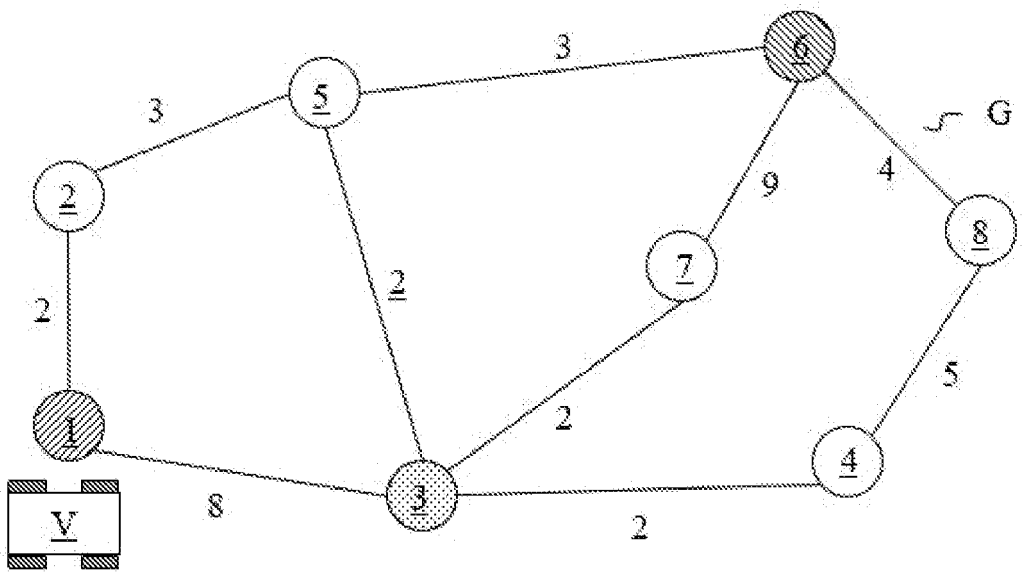
- des moyens d'obtention pour obtenir (30) une instance dudit graphe, dite instance à résoudre, chaque instance étant représentée par un nœud de départ, un nœud d'arrivée et un ensemble de nœuds obligatoires par lesquels le véhicule doit passer, lesdits nœuds de départ, d'arrivée et obligatoires formant les nœuds de l'instance à résoudre ;
- des moyens d'obtention pour obtenir (31) une information représentative d'un ordre initial de parcours desdits nœuds obligatoires en appliquant une procédure utilisant un réseau de neurones convolutif de graphe, dit RNCG, adapté audit graphe à ladite instance à résoudre ;
- des moyens d'exécution pour exécuter (32) une procédure d'optimisation locale de l'ordre des nœuds obligatoires afin d'obtenir un ordre des nœuds obligatoires optimisé, ladite procédure étant initialisée avec ladite information représentative de l'ordre initial ;
- des moyens de définition pour définir (33) un chemin pour ladite instance à résoudre à partir de l'ordre des nœuds obligatoires optimisé et pour chaque paire de nœuds de ladite instance à résoudre, d'un plus court chemin entre les nœuds de ladite paire.

[Revendication 6] Véhicule comprenant un dispositif selon la revendication 5.

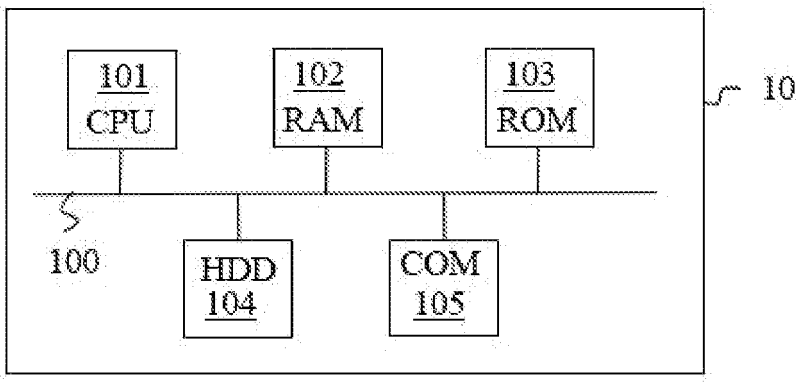
[Revendication 7] Programme d'ordinateur, caractérisé en ce qu'il comprend des instructions pour mettre en œuvre, par un dispositif, le procédé selon l'une quelconque des revendications 1 à 4, lorsque ledit programme est exécuté par un processeur dudit dispositif.

[Revendication 8] Moyens de stockage, caractérisés en ce qu'ils stockent un programme d'ordinateur comprenant des instructions pour exécuter, par un dispositif, le procédé selon l'une quelconque des revendications 1 à 4, lorsque ledit programme est exécuté par un processeur dudit dispositif.

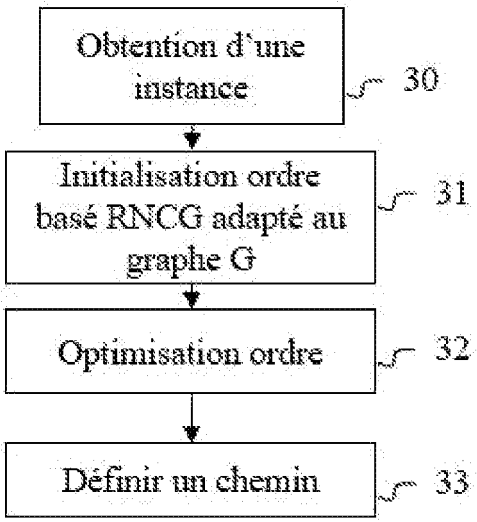
[Fig. 1]



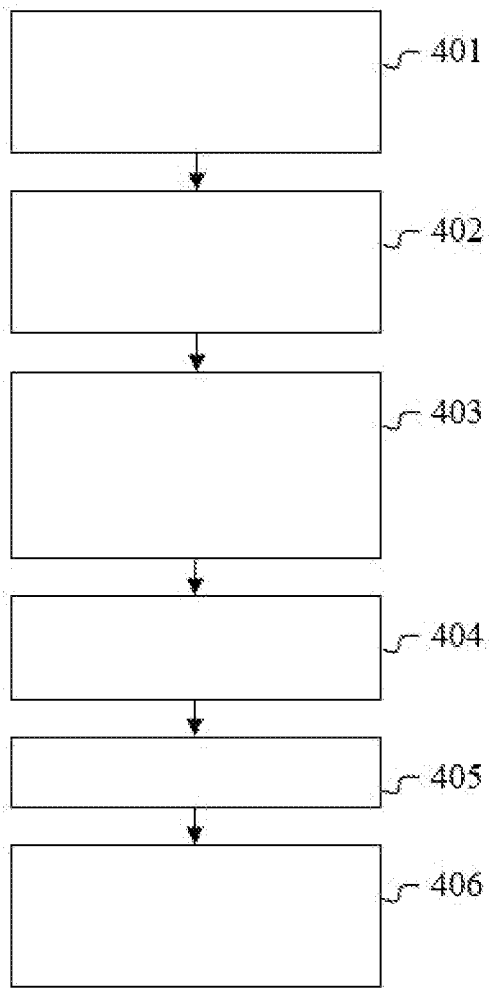
[Fig. 2]



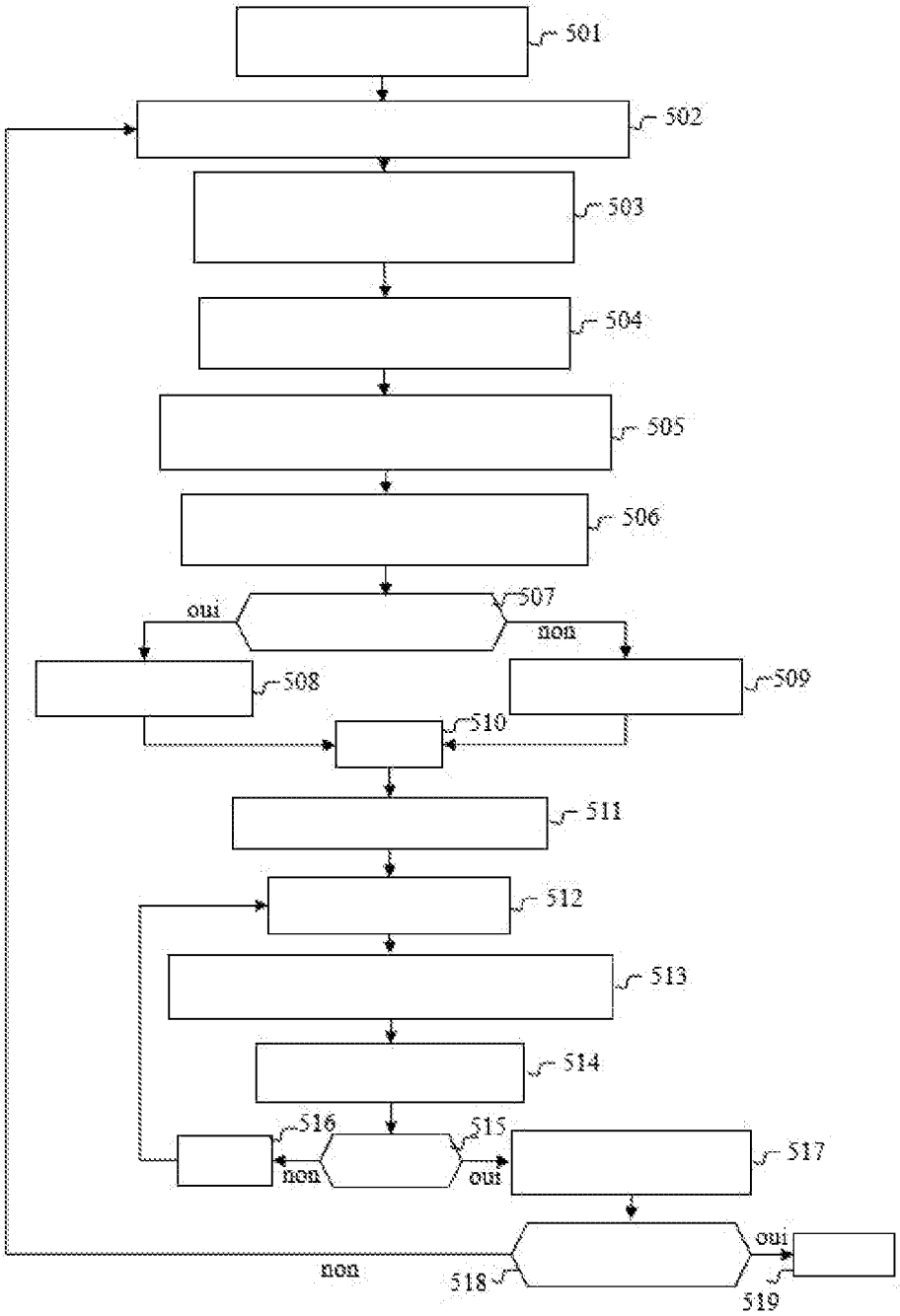
[Fig. 3]



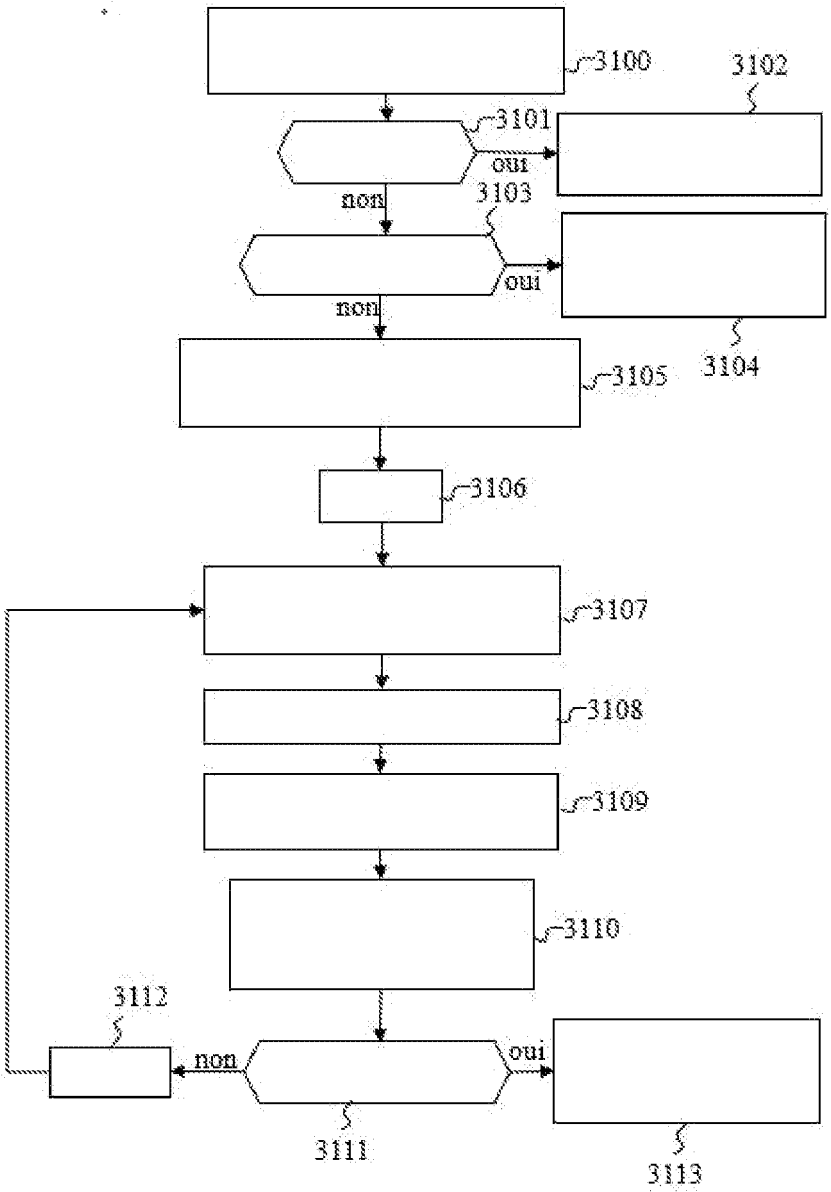
[Fig. 4]



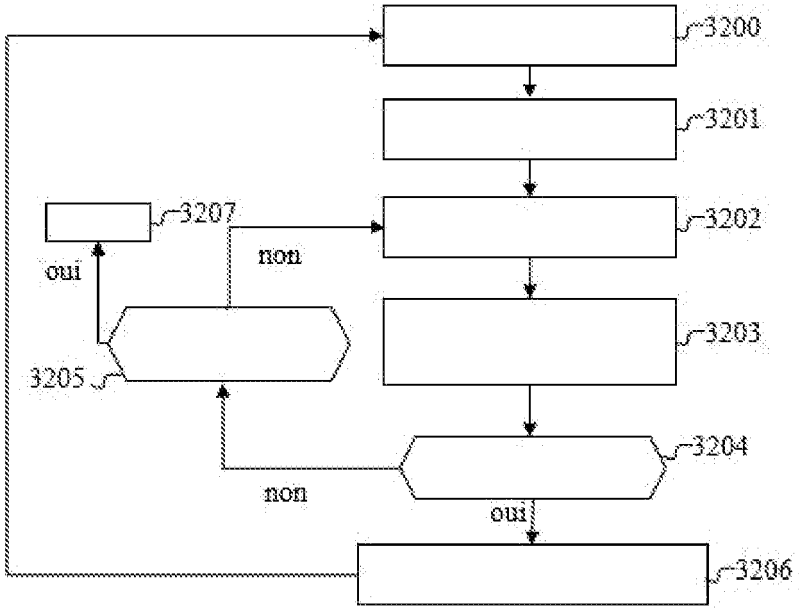
[Fig. 5]


501
502
503
504
505
506
507
oui
non
508
509
510
511
512
513
514
516
515
non
oui
517
oui
non
518
519

[Fig. 6]


3100
3102
3101
oui
non
3103
oui
non
3104
3105
3106
3107
3108
3109
3110
3112
non
oui
3111
3113

[Fig. 7]



RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

N° d'enregistrement national

FA 869232
FR 1903913

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	Kevin Osanlou ET AL: "Constrained Shortest Path Search with Graph Convolutional Neural Networks", , 1 janvier 2018 (2018-01-01), XP055672759, Extrait de l'Internet: URL:http://www.lamsade.dauphine.fr/~cazenave/papers/pathcnn.pdf [extrait le 2020-03-02]	1,2,4-8	G01C21/34 G01C21/20 G06Q10/04 H04L12/701 H04L12/721 H04L29/02 G06F17/10 G06N3/04 G06N20/00
A	* abrégé * * figures 1-5 * * 2 Context and problem presentation * * 3.3 Global search algorithm * * 4 Neural network training * * 5 Data generation * -----	3	

DOMAINES TECHNIQUES RECHERCHÉS (IPC)

G01C
G06Q
G06N

Date d'achèvement de la recherche	Examinateur
3 mars 2020	Toth, Rémy

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)