Etat de l’art des méthodes de modélisation des problèmes de transport

1. Introduction

Les activités de transport de personnes et de marchandises posent plusieurs problématiques à tous les niveaux décisionnels (i.e. stratégique, tactique et opérationnel). Que cela soit pour la construction de réseaux logistiques, leur gestion et leur organisation, l'optimisation de la circulation des flux (personnes et marchandises), l'optimisation de la circulation des véhicules (planification des tournées et optimisation des parcours), etc. La majeure partie des problématiques relatives à la chaîne logistique et au transport, sont des problèmes d'optimisation combinatoire et sont classées dans la catégorie des problèmes NP-difficile (Gupta & Könemann, 2011).

Une grande variété de méthodes de résolution exactes et approchées ont été proposées pour résoudre de tels problèmes. Les solveurs sont souvent en mesure de résoudre les petites et moyennes instances de manière optimale. Cependant, des modèles riches ou des instances de grande taille, ne peuvent être résolus de manière optimale, même par les solveurs les plus performants, dans des délais acceptables. Ainsi, il est nécessaire de développer des algorithmes pour une résolution approchée (Alumur , et al., 2015).

Dans ce travail de recherche, nous nous intéressons à la conception et à la modélisation d'une nouvelle solution de transport ferroviaire de marchandises en milieu urbain. Dans ce cadre, plusieurs problèmes d'optimisation combinatoire sont identifiés. La simulation à évènements discrets, en utilisant le logiciel ARENA, est proposée dans le but de comprendre le comportement de cette solution de transport et d'étudier sa dynamique dans le temps. Le modèle de simulation nous permet aussi, de reproduire les perturbations liées à la mise en exploitation d'une telle solution de transport.

Un couplage simulation / optimisation est proposée. Il permet de considérer l'évolution du système en prenant en compte la dynamique temporelle et de fournir des données actualisées au modèle d'optimisation, à intervalle régulier. L'utilisation d'une approche de replanification complète cette approche, avec l'objectif de minimiser les variations, engendrées par les recalculs, suite aux perturbations de la demande.

Ce chapitre est structuré en quatre sections. Il dresse l'état de l'art des méthodes de modélisation et de résolution, des problèmes proches du FRTSP.

2. Problèmes classiques de la littérature liés à la logistique urbaine

2.1. Problèmes classiques liés au transport

Beaucoup de travaux de recherche se sont intéressés au transport et ont utilisés différentes méthodologies et différents outils pour atteindre leurs objectifs. Ces problèmes de transport (et plus globalement, de logistique) peuvent être regroupés en plusieurs grandes familles de problèmes, tels que :

- Le problème de transport « sources-destinations » (Fortz, 2012) : ce problème concerne la détermination des quantités à livrer de produits à partir de plusieurs sources différentes à destination de plusieurs clients, sachant que : 1- chaque source

dispose d'une quantité limitée, 2- chaque client a une demande maximum et 3- chaque paire « source-destination » a un coût propre, proportionnel à la quantité transportée. L'objectif dans ce type de problème est de minimiser les coûts de transport/livraison. Aussi, des plateformes de transbordement peuvent être introduites, pour réduire les coûts de livraison, à travers la mutualisation du transport sur une certaine portion commune des parcours.

- Le problème de localisation (Melo, et al., 2009) : ce problème se pose suite aux problèmes issus de la classe précédente et dans le cas où les installations de production (sources) n'existent pas encore (donc il s'agit de les réaliser). Le problème décisionnel en question, est de définir les localisations de leur implantation, dans le but de minimiser les futurs coûts de transport ou les futures distances à parcourir pour procéder aux livraisons.
- Le problème de tournée de véhicules (Dantzig & Ramser, 1959) : cette classe de problèmes classiques en optimisation combinatoire, consiste à déterminer les tournées optimales de plusieurs véhicules, dans le but de livrer plusieurs clients (à noter que lorsqu'il n'y a qu'un seul véhicule, ce problème se réduit au problème du voyageur de commerce). Plusieurs variantes de problèmes de tournées de véhicules existent. A titre d'exemple : 1- avec fenêtre de temps, où il s'agit de livrer chaque client, sur un intervalle de temps réduit, 2- avec collecte et livraison, où il s'agit pour les véhicules de transport, de procéder à des collectes de marchandises, en plus des livraisons, 3- contraintes de capacité, où les véhicules ont une capacité limité en termes de poids / volume…
- Le problème d'affectation (Schrijver, 2005) : c'est l'un des premiers problèmes d'optimisation combinatoire. Il a été étudié par Gaspard Monge dès 1784. Ce problème est le plus souvent assimilé au problème de transport décrit précédemment. La description classique de ce problème consiste à déterminer l'affectation de tâches à des agents, sachant que chaque affectation tâche-agent a un coût, l'objectif étant de minimiser le coût total des affectations. Appliqué au transport, il s'agira de déterminer pour chaque commande (produits ou marchandises), le véhicule qui va la transporter. Une variante plus complexe de ce type de problème existe, il s'agit du problème d'affectation généralisée, qui introduit des contraintes supplémentaires. Ce problème sera décrit plus en détail dans la suite de ce chapitre.
- Le problème de bin packing (Johnson & Garey, 1985) : ce problème consiste à déterminer le meilleur placement d'objets, à l'intérieur de plusieurs contenants, dont l'objectif est la minimisation du nombre de contenants pour tous les objets. Appliqué au transport, ce problème se traduit par l'optimisation du chargement de véhicules de transport (ex : camions) ou containers (à transporter dans des bateaux ou trains), dans le but de minimiser le nombre de véhicules à utiliser pour transporter tous le produits. Des contraintes sur le volume du contenant et le poids maximum supporté par les véhicules sont à considérer.

Au-delà des autres problèmes d'optimisation combinatoire, qui peuvent être rencontrés en transport, on peut relever dans certains cas, l'interaction de plusieurs de ces problèmes, pour traduire un problème réel. Par exemple : un problème de tournées de véhicules avec contraintes de capacité, associé à un problème de bin packing pour optimiser le chargement

des véhicules. Un problème d'affectation, associé à un problème de bin packing pour optimiser l'utilisation de l'espace, etc.

Le problème principal étudié dans ce travail de recherche, sera réduit à un problème d'affectation généralisée. Ainsi, pour mieux appréhender les spécificités de ce problème, nous allons le présenter, ainsi que ses caractéristiques dans le point suivant.

2.2. Problème d'affectation généralisée

Le problème d'affectation généralisée a été décrit et étudié en détail, par plusieurs auteurs : (Martello & Toth, 1990), (Cattrysse & Van Wassenhove, 1992), (Chu & Beasley, 1997) et (Ramalhinho Lourenço & Serra, 2000). Ce problème considère la minimisation du coût d'affectation de N tâches à M machines (agents), tel que chaque tâche est affectée à une seule machine, sous la contrainte de capacité de cette dernière. Ce problème trouve des applications dans plusieurs domaines, tels que : les problèmes de localisation, les réseaux de transport, les réseaux de communication, les problèmes de planification des machines, les problèmes de tournées de véhicules, etc. La Figure III.1 montre une schématisation du problème appliqué à l'affectation de plusieurs commandes (lots) de produits de tailles différentes, à des véhicules différents, qui ont des coûts de transport et des capacités différentes. L'objectif étant la minimisation du coût de livraison de toutes les commandes.

Ce problème peut être formulé par un modèle mathématique de type PLNE. Les notations utilisées se présentent comme suit :

- On considère un atelier de production composé de M machines ($m \in [1, M]$), qui doivent réaliser N tâches ($n \in [1, N]$).
- L'exécution d'une tâche n par la machine m consomme les ressources de cette machine (le temps d'exécution par exemple). On note b_{nm} la quantité de ressource nécessaire à la réalisation de la tâche n par la machine m.
- Chaque machine à une capacité de ressources limitée notée a_m.
- L'affectation d'une tâche n à une machine m engendre un coût noté c_{nm}.

Aussi, on définit la variable de décision :

$$x_{nm} = \begin{cases} 1 \; si \; la \; tâche \; n \; est \; affectée \; à \; la \; machine \; m \\ 0 \; sinon \end{cases}$$

Le PLNE s'écrit comme suit :

$$\min f(x) = \sum_{m=1}^{M} \sum_{n=1}^{N} c_{nm} x_{nm} \tag{3.1}$$

s.c.

$$\sum_{n=1}^{N} b_{nm} x_{nm} \leq a_m \qquad \forall m \in [1, M] \tag{3.2}$$

$$\sum_{m=1}^{M} x_{nm} = 1 \qquad \forall n \in [1, N] \tag{3.3}$$

$$x_{nm} \in \{0,1\} \qquad \forall n \in [1, N], \forall m \in [1, M] \tag{3.4}$$

L'objectif (3.1) étant la minimisation du coût total de l'affectation des tâches aux machines, sous l'ensemble des contraintes (3.2) de capacité des machines et l'ensemble des contraintes (3.3), qui permettent d'assurer l'affectation de chaque tâche à une seule machine. L'ensemble des contraintes (3.4) représente les contraintes d'intégrité des variables de décision.

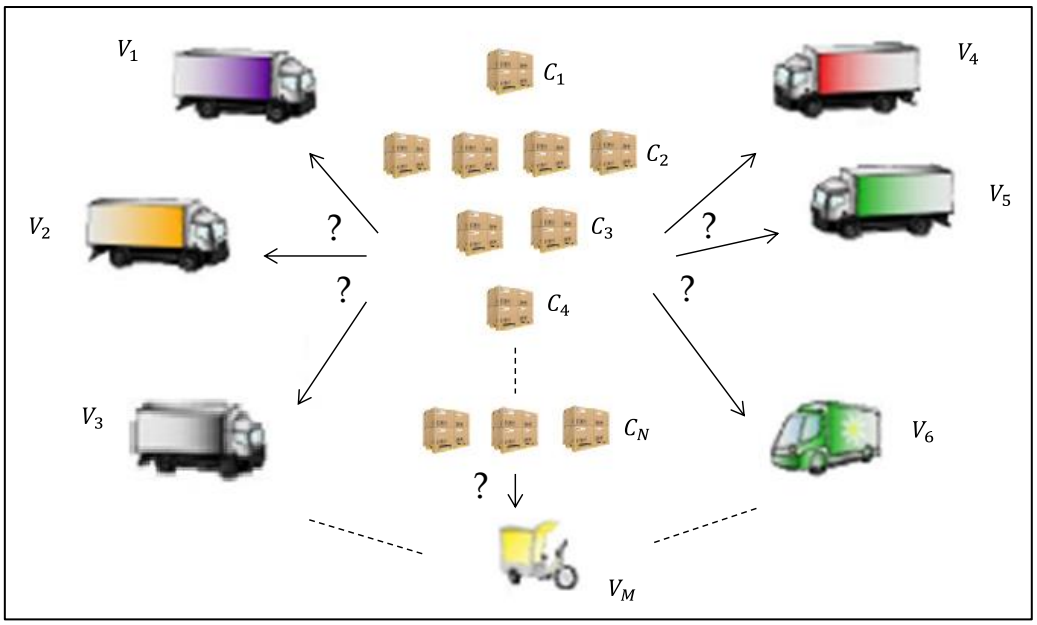

Figure III.1 : Schématisation du problème d'affectation généralisée dans le cas de marchandises à livrer

Il a été démontré que ce problème est NP-difficile dans (Fisher, et al., 1986). Toutefois, plusieurs algorithmes de résolution exacte de ce problème, pour des instances de petite et moyenne taille, ont été proposés par plusieurs auteurs dans : (Ross & Soland, 1975), (Fisher, et al., 1986), (Guignard & Rosenwein, 1989), (Martello & Toth, 1990), etc. Dans (Savelsbergh, 1997), l'auteur a proposé un algorithme qui fournit la solution optimale à un problème composé de 20 machines et 200 tâches en moins de 1 160 secondes dans le pire des cas. D'autre part, plusieurs heuristiques et métaheuristiques ont été proposées pour résoudre des instances de taille plus importante. Dans (Amini & Racer, 1994), les auteurs ont proposé une heuristique à recherche en profondeur variable, inspiré des travaux développés dans (Lin & Kernighan, 1973) pour le problème du voyageur de commerce. Dans (Cattrysse & Van Wassenhove, 1992), les auteurs ont formulé le problème sous forme d'un problème de partitionnement et ont proposé une heuristique basée sur les techniques de génération de colonnes. Dans (Chu & Beasley, 1997), un algorithme génétique est proposé pour améliorer la qualité et la faisabilité des solutions explorées. Dans (Laguna, et al., 1995) et (Dıaz & Fernández, 2001) des algorithmes de recherche tabous ont été proposé, pour la résolution du problème de base et de certaines de ses extensions. Enfin, une autre classe de métaheuristique a été utilisée par plusieurs auteurs, il s'agit des colonies de fourmis proposées, entre autres, dans (Stützel, 1998), (Stützel & Hoos, 2000) et (Ramalhinho Lourenço & Serra, 2000).

3. Méthodes de résolution de la RO

En majorité, les outils utilisés par les chercheurs dans le cadre des études relatives aux problématiques de transport, correspondent à des méthodes de résolution des problèmes d'optimisation combinatoire. Ces méthodes de résolution sont classées en deux grandes

catégories, comme le montre la Figure III.2. D'un côté, nous avons les méthodes exactes qui permettent de trouver et prouver l'optimalité. Cependant, pour la plupart des problèmes, les temps de calcul augmentent exponentiellement en fonction de leur taille. Ainsi, d'autre part, nous avons les méthodes approchées qui permettent de trouver des solutions faisables, généralement de bonne qualité, avec des temps de calculs raisonnables.

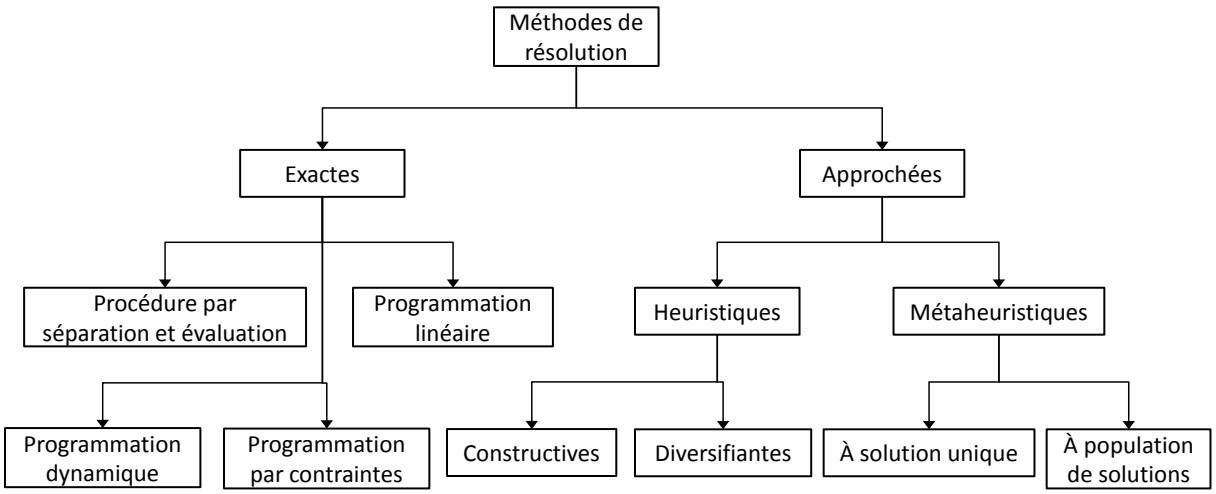

Figure III.2 : Méthodes de résolution pour l'optimisation combinatoire

3.1. Méthodes exactes

La majorité des problèmes de transport peuvent être modélisés sous la forme d'un Programme Linéaire en Nombre Entier « PLNE ». Un PLNE est un cas particulier des programmes linéaires « PL », où les variables de décision sont dans l'ensemble des entiers. En moyenne, un PL est résolu de manière assez rapide, en utilisant l'algorithme du simplexe. Cependant, même si la relaxation des contraintes d'intégrité d'un PLNE facilite sa résolution, car elle le ramène à un PL, la relaxation obtenue peut être très éloignée de l'optimum. Dans ce cas, plusieurs autres approches de résolution sont utilisées, telles que la procédure de séparation et évaluation, ou la programmation dynamique. Les points suivants permettent de détailler ces différentes notions et méthodes de résolution :

- **Programmation linéaire**

  Dans (Chvatal, 1983), une étude détaillée de la programmation linéaire, à partir de ces origines, et à travers la présentation de différentes méthodes de résolution. Ainsi, la programmation linéaire est une discipline mathématique relativement récente, qui doit son émergence à la conception de la « méthode du simplexe » par G.B. Dantzig, en 1947, pour la résolution d'un problème de planification de l'US Air Force (qui a été formulé sous la forme d'un programme linéaire).

  Un problème de programmation linéaire est un problème de maximisation (ou minimisation) d'une fonction linéaire, sujet à un nombre fini de contraintes linéaires. En général, on utilise l'indice *j* pour les *n* variables « de décisions » et l'indice *i* pour les *m* contraintes, ce qui nous permet d'écrire un PL sous la forme suivante :

$$\max \sum_{j=1}^{n} c_j x_j \qquad (3.5)$$

s.c.

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \qquad i = (1,2,\dots,m) \tag{3.6}$$

$$x_j \geq 0 \qquad j = (1,2,\dots,n) \tag{3.7}$$

Cette écriture du PL est appelée la forme standard, avec (3.5) la fonction « objectif » et (3.6) l'ensemble des *m* contraintes.

Une première méthode de résolution intuitive d'un PL, est la résolution graphique (Nemhauser & Wolsey, 1988). Cette méthode ne peut être appliquée que si le problème ne dépasse pas les 3 variables de décision. Chaque problème a un certain nombre de solutions qui vont satisfaire toutes ces contraintes. On les appelle l'ensemble des solutions réalisables *S*, cet ensemble peut être vide. Les autres solutions, qui satisfont une partie des contraintes ou aucune d'elle, sont appelées des solutions irréalisables. Ainsi, la représentation graphique des différentes contraintes, permet de définir l'ensemble *S* (comme le montre la Figure III.3). La représentation de l'objectif sous la forme d'une droite qu'on fait translater dans le sens de l'optimisation (maximisation ou minimisation), nous permet l'obtention de la solution optimale (cette solution peut être unique ou multiple). L'espace *S* est aussi appelé enveloppe convexe. Aussi, la solution optimale se trouve forcément sur l'un de ses points extrêmes.

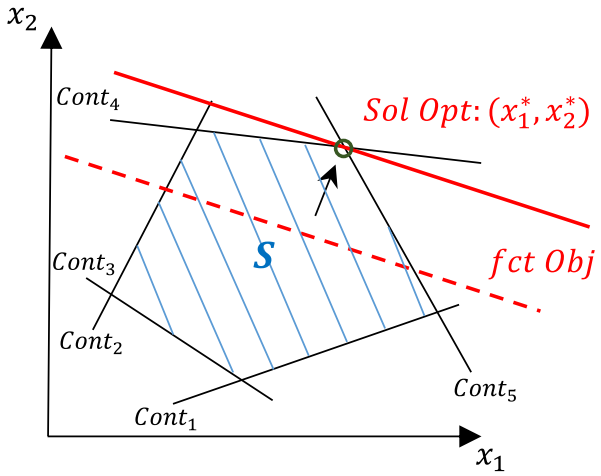

Figure III.3 : Exemple d'une résolution graphique d'un PL à 5 contraintes

L'exploration des différents points extrêmes, constitués par l'intersection des différentes contraintes du PL, a été formalisée dans le cas général (peu importe le nombre de variables), par l'algorithme du simplexe, qui a été développé par G.B. Dantzig (Dantzig, 1951). Le principe de l'algorithme du simplexe est d'explorer les points extrêmes, à travers un processus itératif, en améliorant la solution d'une itération à l'autre.

Dans certains cas, la structure du problème fait qu'il est plus intéressant de passer par la forme duale du PL, sachant que ce dernier a la même solution optimale que le problème original.

- **Programmation linéaire en nombres entiers**

Un PLNE est un PL où toutes les variables sont entières, la formulation précédente reste valable avec cette contrainte supplémentaire. Une forme intermédiaire entre PL

et PLNE existe. Il s'agit de la programmation linéaire en variables mixtes « PLVM » (Nemhauser & Wolsey, 1988). La forme standard des PLVM se présente comme suit :

$$\max \sum_{j=1}^{n} c_j x_j + \sum_{k=1}^{p} d_k y_k \qquad (3.8)$$

s.c.

$$\sum_{j=1}^{n} a_{ij} x_j + \sum_{k=1}^{p} e_{ik} y_k \leq b_i \qquad i = (1,2,\dots,m) \qquad (3.9)$$

$$x_j \in \mathbb{Z}_+ \quad j = (1,2,\dots,n), \quad y_k \in \mathbb{R}_+ \quad k = (1,2,\dots,p) \qquad (3.10)$$

Comme mentionné dans le préambule de cette sous-section, la résolution du PL obtenu par relâchement de la contrainte d'intégrité, ne coïncide que très rarement avec la solution optimale du PLNE. En effet, à la différence du PL, en général la solution optimale du PLNE se trouve à l'intérieur de l'enveloppe convexe (Figure III.3) et non pas sur sa frontière. De là, la résolution d'un PLNE requiert l'application d'approches d'énumération et / ou de décomposition (Sun & Quilliot, 1993). Parmi ces approches de résolutions : la procédure par séparation et évaluation (Lawler & Wood, 1966) et (Narendra & Fukunaga, 1977), la programmation dynamique (Bellman, 1956), les méthodes de coupes (Padberg & Rinaldi, 1991), les méthodes de génération de colonnes (Vanderbeck & Wolsey, 1996), etc. Aussi, le problème formulé par un PLNE, peut être formulé par la programmation par contraintes « PPC » (Mackworth, 1977).

- **Procédure par séparation et évaluation « B&B »**

Cette approche de résolution trouve ses racines dans (Land & Doig, 1960). Par la suite, ses principes ont été développés par plusieurs auteurs tels que dans : (Bertier & Roy, 1964), (Dakin, 1965) et (Roy, et al., 1965). Les algorithmes de type B&B consistent à énumérer de manière implicite les solutions du PLNE, pour en trouver la meilleure. Cependant, cette démarche implique des temps de calcul trop importants. A titre d'exemple, l'exploration de toutes les solutions d'un PLNE avec 60 variables de décision en (0,1), en utilisant un ordinateur qui met 10^{-9} secondes pour explorer chaque solution, il faudrait 30 ans pour évaluer toutes les solutions possibles (Minoux, 1983). Ainsi, les schémas se basant sur l'énumération, procèdent de manière non exhaustive.

La procédure de B&B a été décrite en détail par plusieurs auteurs tels que dans : (Minoux, 1983), (Sakarovitch, 1984), (Nemhauser & Wolsey, 1988) et (Dell'Amico, et al., 1997). Cette procédure se compose de deux étapes :

- La séparation : cette étape consiste à diviser le problème en plusieurs sous-problèmes. Chaque sous-problème a un ensemble de solutions réalisables, dont l'union recompose l'ensemble des solutions du problème. Cette décomposition se fait à travers la représentation de l'ensemble des solutions *S* sous la forme d'une arborescence. Les nœuds de cet arbre correspondent aux sous-problèmes (le nœud de niveau 0 est appelé la racine et correspond à l'ensemble *S*). Pour construire un niveau de l'arborescence, il faut choisir une variable de manière arbitraire et lui fixer une valeur. La suite est effectuée à travers la deuxième étape de la procédure.

- L'évaluation : pour parcourir l'arbre, un mécanisme d'évaluation est mis en place. Ce mécanisme permet l'exclusion des sous-arborescences qui ne peuvent contenir une solution optimale. De là, la performance de chaque algorithme de type B&B dépend directement de cette fonction d'évaluation, dont la performance est liée à l'écart entre la valeur qu'elle fournit et la valeur exacte de l'optimum entier. Plus cet écart est faible, plus la qualité de l'évaluation est bonne, ce qui permet de réduire le nombre de nœuds à examiner. Ainsi, le mécanisme d'évaluation permet de réduire la taille de l'arbre à explorer, en arrêtant l'expansion des branches qui ne contiennent pas une solution optimale.
  L'arborescence est parcourue soit en largeur, en profondeur ou, en priorisant les meilleurs nœuds à l'itération actuelle. A chaque séparation, la borne supérieure pour un problème de maximisation (inférieure pour un problème de minimisation) est calculée. En général, elle est obtenue à partir de la résolution du sous-problème relaxé correspondant au nœud actuel (une relaxation linéaire, par exemple).

- **Programmation dynamique**

La programmation dynamique a été proposée au début des années 1950 par R.E. Bellman (Bellman, 1957), dans le but de résoudre des problèmes de chemins optimaux (plus court ou plus long chemin). Depuis, cette méthode a été développée et est très utilisée dans la résolution des problèmes d'optimisation combinatoire (Bellman & Dreyfuce, 1962), (Denardo & Mitten, 1967), (Christofides, et al., 1981) et (Nemhauser & Wolsey, 1988).

De la même manière que le B&B, cette méthode est basée sur le principe d'énumération implicite, où les sous-ensembles de solutions sont retenus ou rejetés, sans construire toutes les solutions possibles (les sous-ensembles rejetés sont ceux qui ne présentent pas d'intérêt dans la recherche de la solution optimale).

La décomposition suivant le principe d'optimalité de Bellman est définie comme suit : si (C) est un chemin optimal reliant le point A au point B (comme le montre la Figure III.4) et si C appartient à (C) alors les sous-chemins de (C) reliant le point A au point C et le point C au point B sont aussi optimaux.

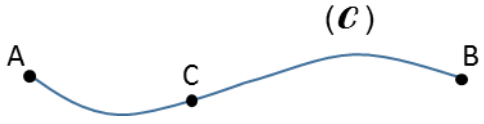

Figure III.4 : Illustration du principe d'optimalité de Bellman

La mise en œuvre de cette méthode requiert le développement d'une formulation récursive du problème. Le découpage étape par étape permet de définir la formule de récurrence. A titre d'exemple, la recherche du chemin le plus court entre deux points d'un graphe pondéré (illustré par la Figure III.5) par cette méthode se présente comme suit :

- $F(y)$: la longueur minimale de tous les chemins reliant x à y.
- $L(z, y)$: la longueur minimale de tous les chemins relient z à y.

ainsi, la formule récursive s'écrit :

$$F(y) = \min_{z \neq y}\big(F(z) + L(z,y)\big) \qquad (3.11)$$

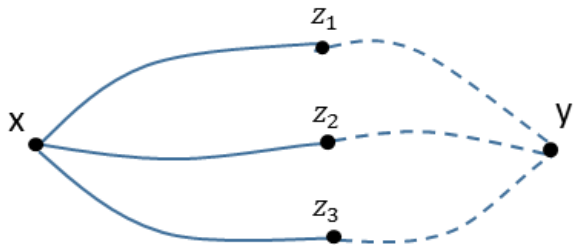

Figure III.5 : Illustration d'un graphe pour la recherche du chemin le plus court entre x et y

- **Programmation par contraintes**

La PPC a été développée vers la fin des années 1970, pour résoudre des problèmes d'optimisation combinatoire de grande taille, principalement, en planification et ordonnancement (Mackworth, 1977) et (Haralick & Elliott, 1980). La démarche de la PPC se fait en deux étapes : 1- la modélisation du problème sous la forme d'un problème de satisfaction de contraintes « CSP », puis 2- l'utilisation active des contraintes, dans la résolution de ce dernier.

Le CSP est un problème qu'on modélise sous la forme d'un ensemble de contraintes relatifs aux variables, chaque variable prend ses valeurs dans un domaine. Ainsi, un CSP est défini par trois composantes :

- $X = \{x_1, x_2, \dots, x_n\}$: l'ensemble des variables du problème.
- D : la fonction qui associe à chaque variable x_i son domaine $D(x_i)$ (i.e. l'ensemble des valeurs que peut prendre x_i).
- $C = \{C_1, C_2, \dots, C_n\}$: l'ensemble des contraintes. Chaque contrainte C_J est une relation entre certaines variables de X, qui restreint les valeurs que peuvent prendre simultanément ces variables.

La résolution d'un CSP consiste en l'affectation de valeurs aux variables, de façon à ce que toutes les contraintes soient satisfaites. Les notions associées à la résolution d'un CSP peuvent être synthétisées comme suit :

- L'affectation est le fait d'instancier certaines variables par des valeurs prises dans les domaines des variables.
- Une affectation est totale si elle instancie toutes les variables du problème. Dans le cas où elle n'en instancie qu'une partie, elle est partielle.
- Une affectation est consistante si elle ne viole aucune contrainte. Dans le cas où elle viole une ou plusieurs contraintes, elle est inconsistante.
- Une affectation totale consistante est une solution du problème.

L'utilisation des contraintes pour la résolution des problèmes, s'effectue à travers une technique appelée « propagation des contraintes » (Baptiste, et al., 2003). Cette technique procède par un mécanisme de déductions à partir des contraintes, afin de réduire les domaines des variables. Cependant, la propagation des contraintes à elle seule, n'est pas suffisante pour détecter toutes les inconsistances de l'espace de recherche, en particulier dans le cadre des problèmes NP-difficile. Cette démarche doit être complétée par un algorithme d'exploration de l'espace de recherche. Un schéma global de la démarche est présenté dans (Baptiste, et al., 2003).

3.2. Méthodes approchées

L'objectif des méthodes approchées est de produire des solutions réalisables de la meilleure qualité (voire optimale), sans se préoccuper de la preuve de l'optimalité. A la différence des méthodes exactes et l'un des avantages majeurs de ces approches, c'est la rapidité avec laquelle elles fournissent des solutions. En effet, elles sont en mesure de considérer la complexité des problèmes réels, posés dans l'industrie (principalement liée à la taille des données à traiter) et de fournir des solutions assez rapidement. Ces méthodes peuvent être classifiées comme suit :

- **Heuristiques**

  Une heuristique est une règle spécifique à un problème donné, qui permet de trouver une, voire des solutions réalisables, souvent loin de l'optimalité, mais le temps de réponse est très court, en contrepartie. Elles sont principalement utilisées dans les cas où il n'y a pas d'autres alternatives, pour la résolution de problèmes avec des contraintes de temps et d'espace.

- **Méthodes de recherche locale**

  La recherche est initiée à partir d'une solution existante, suivie d'une recherche de voisinage, pour améliorer la solution. Il y a trois types de recherche locale : 1- à voisinage proche « LS », 2- à large voisinage « LNS » et 3- à voisinage variable « VNS ». Ces méthodes sont décrites en détail dans (Siarry, 2014).

  Ce qui est appelé voisinage, c'est l'ensemble des solutions pouvant être obtenues par application d'une transformation (permutations ou mouvements) de la solution courante. Parmi les procédures de transformation : 1- la recherche du meilleur d'abord : qui consiste en la construction d'un arbre de recherche, où à chaque itération, le voisin sélectionné est celui qui permet l'amélioration de la solution (le premier à être trouvé). 2- la recherche du meilleur voisin : même chose que la première procédure, sauf que le meilleur voisin choisi est sélectionné à partir de tous les voisins de même niveau de l'arbre de recherche (i.e. tous les voisins sont testés et celui qui améliore le plus la solution est sélectionné).

  La solution obtenue à l'issue de l'application d'une méthode de recherche locale, est appelée optimum local.

- **Métaheuristiques**

  Les métaheuristiques permettent de trouver des solutions réalisables, de la meilleure qualité possible, en un temps raisonnable. De plus, elles sont génériques, puisqu'elles peuvent être adaptées à n'importe quel problème.

  Créées au début des années 1980, elles ont connu un développement très important ces dernières années. Elles sont conçues pour la résolution de problèmes d'optimisation combinatoire difficiles, qui ne peuvent être résolus à travers des méthodes exactes ou des heuristiques. Formellement, selon (Osman & Laporte, 1996), une métaheuristique est un processus itératif qui pilote une heuristique subordonnée, combinée à diverses techniques, dans le but d'exploiter et d'explorer au mieux l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver des solutions proches de l'optimum (voire optimales dans certains cas).

Deux familles de métaheuristiques sont à distinguer, à savoir, celles qui ne manipulent qu'une seule solution à la fois et celles qui manipulent une multitude de solutions.

- **Métaheuristiques à solution unique**

  Plusieurs métaheuristiques de ce type ont été développées. Parmi les plus utilisées et rapportées dans la littérature (Siarry, 2014), on retrouve :

  1- La procédure de recherche gloutonne randomisée « GRASP » : elle a été développée par (Feo, et al., 1994), son principe est de créer à chaque itération, une nouvelle solution, en faisant appel à un algorithme glouton aléatoire. Cette solution est améliorée avec une procédure de recherche locale. Le processus est renouvelé un nombre d'itération fois, à l'issue duquel la meilleure solution trouvée est retenue.

  2- La recherche locale itérative « ILS » : très utilisée par plusieurs auteurs, tels que dans (Johnson, 1990), elle applique le même principe que GRASP. Cependant, la solution de départ pour chaque itération est obtenue suite à l'application d'une perturbation aléatoire au niveau de la meilleure solution courante.

  3- Le recuit simulé « SA » : développé dans (Kirkpatrick, et al., 1983), il s'inspire d'un processus métallurgique qui consiste en le recuit thermique. Le principe adopté permet de sortir des optimums locaux et d'augmenter les chances de converger vers l'optimum global.

  4- La recherche tabou « TS » : développée dans (Glover , 1986), se base sur un processus itératif d'exploration de l'espace de recherche, en utilisant une liste « tabou » qui enregistre les dernières solutions explorées. Cette liste lui permet de sortir d'éventuels optimums locaux, ce qui peut induire le passage par des solutions de moins bonne qualité, mais dans le but de converger vers l'optimum global.

- **Métaheuristiques à population de solutions**

  Principalement inspirées par des phénomènes observés dans la nature, ces métaheuristiques présentent de très bonnes performances dans la résolution des problèmes d'optimisation combinatoire difficiles. Dans ce qui suit, quelques rappels sur les métaheuristiques les plus connues de ce type :

  1- Algorithme génétique « GA » : introduite dans (Holland, 1975), cette métaheuristique démarre à partir d'une population de solutions initiale, qu'elle améliore au fur et à mesure des itérations. C'est ce qu'on appelle des algorithmes évolutionnistes. L'inspiration des GA vient de l'évolution naturelle des espèces, grâce aux croisements, sélections et mutations, pour faire évoluer la population de solutions, qu'on représente sous la forme de chromosomes, vers l'optimum. Le mécanisme de base étant le croisement à chaque itération, de deux chromosomes parents de la population (en favorisant les plus prometteurs), pour l'obtention d'un ou deux chromosomes enfants. Aussi, des mutations à ces chromosomes enfants peuvent être appliquées, afin de diversifier l'espace de recherche et éviter les convergences prématurées.

2- Recherche dispersée « SS » : décrite dans (Martí, et al., 2006), cette métaheuristique fait partie des algorithmes évolutionnistes (de la même manière que les GA), sauf qu'elle y intègre un mécanisme de recherche locale. En effet, à chaque itération, elle génère un sous-ensemble de solutions composé d'éléments répondant à des critères de qualité et de diversité, appelé ensemble de référence. Les solutions de plusieurs ensembles de références sont combinées pour produire de nouvelles solutions. Par la suite, une recherche locale est exécutée pour améliorer la solution, en remplaçant les éléments les moins bons, tout en satisfaisant l'ensemble des critères de qualité et de diversité. L'algorithme s'arrête après un nombre donné d'itérations, en retenant la meilleure des solutions trouvées.
3- Optimisation par essaims de particules « PSO » : introduite dans (Eberhart & Kennedy, 1995), cette métaheuristique s'inspire des mouvements coordonnés qu'on retrouve dans le monde des animaux et des insectes. Tel que les déplacements en groupe des oiseaux, dans le but de chercher de la nourriture ou d'éviter des prédateurs. Les algorithmes de type PSO font collaborer les individus de l'essaim (appelés particules), dans le but de converger progressivement vers l'optimum global, à partir de plusieurs optimum locaux. La démarche adoptée pour cela, s'appuie sur une équation de mouvement pour les particules, dont les éléments sont : 1- à l'initialisation, chaque particule est positionnée aléatoirement, 2- la vitesse actuelle de la particule (pondérée par un facteur appelé inertie), 3- sa meilleure solution (pondérée par un facteur tiré aléatoirement) et 4- la meilleure solution obtenue dans son voisinage (aussi pondérée par un autre facteur tiré aléatoirement). A savoir que le PSO est plus performant pour les problèmes en variables continues.
4- Optimisation par les colonies de fourmis « ACO » : introduite dans (Dorigo, et al., 1991), cette métaheuristique a connu beaucoup de développement, du fait des résultats intéressants obtenus lors de la résolution de plusieurs problèmes d'optimisation difficiles. Comme le PSO, l'ACO s'inspire aussi du comportement du monde des insectes, à travers les colonies de fourmis et les phénomènes d'auto-organisation et d'émergence qui les caractérisent. Le principe emprunté par l'ACO aux fourmis est celui qu'elles déploient lors de la recherche de sources de nourriture et son acheminement jusqu'à leur nid. Une fois la source trouvée, les fourmis laissent derrière elles, des traces de phéromones pour guider les autres fourmis. La particularité liée à l'évaporation des phéromones, conjuguée à l'exploration de plusieurs chemins, explique l'attrait des fourmis vers le chemin le plus court. En effet, celui-ci se trouve avec la plus grande concentration de phéromones, puisqu'il sera le plus emprunté le plus rapidement, au détriment des autres chemins qui sont délaissés progressivement, en raison du phénomène d'évaporation des

phéromones. Tout cela, sans que les fourmis n'aient une vision globale du trajet.

Appliqué à un problème d'optimisation, ce principe se présente comme un mécanisme de construction d'une population de solutions à chaque itération. En effet, chaque fourmi construit une solution complète et laisse une trace de phéromone sur le chemin de construction de la solution, qui est proportionnelle à la valeur de la fonction objectif (dans le cas d'un problème de maximisation et inversement dans le cas d'une minimisation). Lors de la construction de la solution, chaque fourmi sélectionne la prochaine composante de la solution qu'elle construit, en fonction des traces de phéromones précédentes (exploitation de l'expérience acquise), ainsi qu'en fonction d'une heuristique (orientée ou aléatoire) qui permet d'explorer d'autres solutions (diversification) et ne pas converger et rester bloqué dans des optimums locaux.

Pour le problème d'affectation généralisée, on compte plusieurs tentatives de résolution avec des algorithmes de type ACO, qui ont présenté de très bonnes performances (comme cité dans la section 2.1 de ce chapitre).

Dans ce travail de recherche, nous allons adapter l'ACO pour résoudre le FRTSP.

4. Problèmes de replanification

L'absorption des perturbations qui sont assez fréquentes, durant les périodes d'exploitation, est l'un des challenges les plus importants pour les entreprises. Elle a un impact direct sur la compétitivité et la performance de l'entreprise. Ainsi et pour faire face aux perturbations, plusieurs outils d'aide à la décision existent, dont des solutions de replanification qui améliorent sensiblement la gestion des perturbations (Cauvin, et al., 2009). Assez répondu en gestion de production, la replanification trouve beaucoup d'applications, dans les différents niveaux de la chaine logistique.

Dans ce travail de recherche, nous allons considérer les changements en cours d'exploitation, en particulier celles concernant la demande, en utilisant une approche classique d'horizon glissant, pour re-planifier le FRTSP. Cette approche est basée sur une formulation mathématique de type programmation linéaire, avec des caractéristiques d'exécution spécifiques et dont l'objectif est de minimiser les changements, par rapport aux décisions prises lors de la prise de décision originale.

Le modèle mathématique sujet de la replanification est soit un PL, PLNE ou PLVM, reprenant les caractéristiques présentées dans la section 2.2.1. Cependant, sa mise à jour requiert l'utilisation de l'horizon glissant (Hétreux, 1996), pour la prise en charge des changements de la demande.

4.1. Le principe d'horizon glissant et résolution d'un problème de replanification

Le principe d'horizon glissant a été décrit et utilisé par plusieurs auteurs, pour prendre en charge plusieurs problématiques réelles. Dans (Chand, et al., 2002), une revue bibliographique des travaux majeurs, ayant contribué à l'essor de cette approche.

Dans toutes les activités liées à la production et à la logistique, les données concernant les commandes des clients ne sont que rarement connues avec certitude. Lorsque c'est le cas, la demande est connue sur un horizon de temps restreint (l'étendue de l'horizon varie en fonction de l'activité et de l'industrie concernées). D'où, le premier paramètre à définir, c'est l'horizon de planification « H_P ». Pour comprendre le principe de cette approche dans le cas général, on suppose que cet horizon est composé de « P » périodes.

De plus, même sur une période de temps donnée, aussi bien en production qu'en logistique (plus globalement), les aléas et perturbations (telles que des défaillances humaines ou matérielles, voire évènements exceptionnels externes) sont des éléments avec lesquels les entreprises doivent composer. La préservation de leur performance est fortement liée à leur capacité de prise en considération de ces derniers. C'est pourquoi chaque nombre donné de périodes « ΔP », un réajustement des décisions prises au plan précédent, doit être effectué. Pour cela, un nouveau paramètre est défini, c'est le pas de planification « r ». En théorie, la valeur maximale que peut prendre r est H_P, cependant et selon (Thierry, 2003), il est préférable d'avoir une fréquence de planification plus importante, dans le but de prendre en compte les aléas au plus tôt. Même si dans (Galasso, 2007), l'auteur est d'accord avec ce raisonnement, il note toutefois que choisir un pas de planification long, entrainant un gel des décisions lointaines, permet d'assurer la stabilité des plans précédents.

Enfin, même si une entreprise doit intégrer les conséquences de l'occurrence d'aléas de tous les types, dans ses plans d'activités, il ne reste pas moins nécessaire d'assurer la stabilité des systèmes sur un certain horizon, où les décisions prises lors du dernier plan, ne peuvent être modifiées. D'où, la dernière notion à définir est l'horizon gelé « H_G ». L'intérêt majeur de H_G est la limitation de la nervosité du système, à travers la non considération d'éventuelles modifications tardives (Fontan, et al., 2001). A savoir que le complément de H_G dans H_P est appelé l'horizon libre « H_L » (i.e. les décisions prises pour une application dans H_L peuvent être remises en question par la nouvelle planification).

Les différentes notions définies, sont illustrées sur une échelle temporelle, par la Figure III.6. Trois itérations sont schématisées, telles que :

- L'itération 1 représente le plan initial, qui est établi à travers la résolution d'un modèle mathématique, dont l'objectif est d'obtenir le meilleur plan initial, au vu des données disponibles.
- Les itérations 2 et 3 (et le suivantes) représentent les plans réajustés, en considérant les changements durant la période r écoulée et ayant un impact sur les périodes futures. Ces plans sont obtenus à travers la résolution d'un modèle mathématique de

replanification, dont l'objectif est de minimiser les changements engendrés par la prise en compte des perturbations.

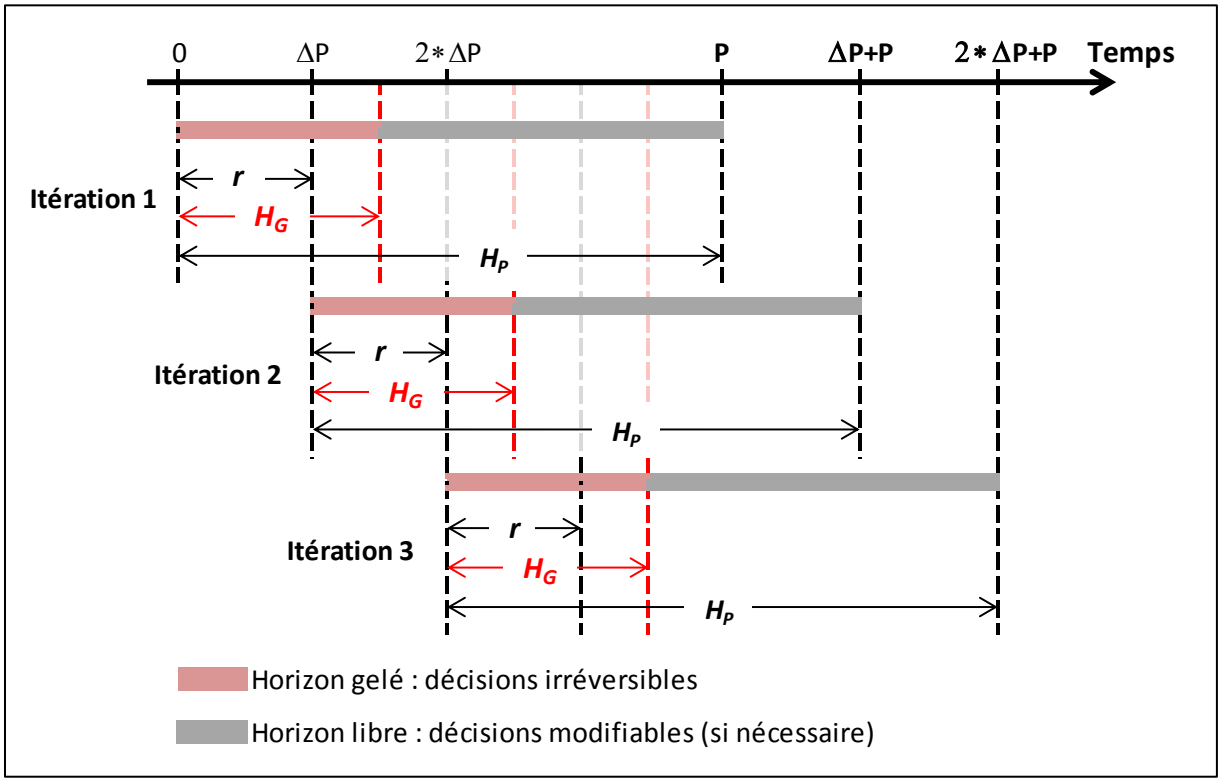

Figure III.6 : Illustration de l'approche par horizon glissant

4.2. La replanification dans la gestion de la chaine logistique

Aux différents niveaux de la chaine logistique, plusieurs travaux ont traités les aspects dynamiques, ainsi que l'intégration des aléas, par des approches de replanification implémentées en utilisant plusieurs outils. En effet, l'environnement tout au long de la chaine logistique est incertains et souvent affecté par des évènements aléatoires tels que : les commandes de dernière minute, annulation de commande, retard dans la production, machines en panne, rupture de stock des matières premières, etc. Tous ces évènements peuvent rendre les plans d'approvisionnement et de production initiaux obsolètes. Pour les rétablir et les rendre à nouveau opérationnels, la replanification offre un outil assez performant. Parmi les premiers travaux menés sur cette approche en production, ceux de (Wu, et al., 1992), où il s'agit de procéder à la replanification des activités d'un atelier de production, soumis à des perturbations.

Dans un contexte de production, plusieurs travaux ont décliné des approches de replanification dans le cadre d'ateliers de production, avec diverses organisations (job shop, machines parallèles, machine unique…) tels que les articles de (Guo, et al., 2016) et (Akkan, 2015) qui traitent des problèmes de replanification, dans un contexte d'atelier à une seule machine. Dans le premier article, les pièces produites peuvent nécessiter des retouches sur la machine (pour rectifier certains défauts). Le problème étant de les insérer dans le plan de production initial, avec un objectif de minimisation des temps d'attente. Quant au second article, il s'agit d'intégrer de nouvelles commandes, avec un objectif de minimisation du retard de production maximum. Dans un autre article (Yin, et al., 2016), un atelier à machines

identiques parallèles est considéré. Dans un premier temps, un plan de production initial est établi, avec un objectif de réduction de la durée totale de production. Puis, durant la phase de production, des perturbations liées aux pannes des machines sont considérées. Un modèle de replanification est proposé, avec un objectif de réduction des changements par rapport au plan initial. Pour cela, les dates d'achèvement de chaque tâche dans le premier plan, deviennent des dates échues du modèle de replanification, ainsi, l'objectif de ce dernier devient la réduction des retards « virtuels » de production.

Dans (Sabar, et al., 2009), les auteurs ont proposé une approche multi-agents, pour traiter un problème de replanification de l'affectation du personnel dans un centre d'assemblage. L'affectation s'effectue en fonction de la compétence des employés, de leur mobilité et de leur préférence, avec un objectif de minimisation des coûts opérationnels et de l'insatisfaction des employés. Les perturbations considérées pour la replanification de l'affectation sont celles relatives à l'absentéisme des employés. Le principe adopté dans cet article est d'établir un plan d'affectation initial, avant le début de la période de production. Puis, au début de la production et après constat des personnes absentes de leur poste de travail, le modèle de replanification n'est lancé que sur les activités devant être exécutées par les absents. Ces activités doivent être affectées au personnel présent avec les mêmes contraintes et objectifs, que dans le plan initial. Cette approche a montré une bonne performance, tant sur le plan de la qualité de la solution proposée, que sur le temps de calcul nécessaire à l'élaboration du nouveau plan.

Un autre problème de planification logistique a été étudié dans (Silva, et al., 2008). Ce problème concerne la considération d'annulation de certaines commandes. L'article effectue une comparaison entre deux méthodes de résolution : GA et ACO, dans la résolution de divers problèmes d'optimisation (problème du voyageur de commerce, problème d'affectation quadratique, problème de tournée de véhicules et problème de job-shop), dans un contexte statique, puis dynamique (avec replanification dans le cas des perturbations). Cette étude a montré que les deux métaheuristiques étaient performantes (i.e. elles fournissent des solutions équivalentes), cependant, la métaheuristique GA fournit plus rapidement les solutions dans le cas des problèmes statiques, à contrario, l'ACO est beaucoup plus rapide dans le cas dynamique, puisqu'elle capitalise les calculs du cas statistique, grâce à la phéromone, pour poursuivre la recherche et fournir un réajustement de la solution.

D'autre part, et dans un contexte où le challenge environnemental est au centre de toutes les préoccupations, dans (Salido, et al., 2017), les auteurs proposent un modèle de replanification dans un atelier de type job-shop. Mêmes préoccupations que celles présentées par les autres travaux, à savoir, dans un environnement dynamique, les perturbations font partie du quotidien de l'entreprise. Suite à l'occurrence de ces perturbations, les plans initiaux ne sont plus optimaux et requiert des réajustements. Pour cela, il est proposé un modèle de replanification implémenté à travers un algorithme mimétique, dont l'objectif est double. En effet, il s'agit de préserver le temps de production total, tout en réduisant la consommation énergétique sur la période impactée par les changements dus à la replanification.

Dans (Vieira, et al., 2003) et (Herrmann, 2006), une revue de littérature de plusieurs travaux en replanification a été réalisée et une formalisation des concepts et définitions usités a été proposée.

4.3. La replanification dans le transport ferroviaire

La replanification dans la gestion des perturbations en temps réel, dans le secteur du transport ferroviaire, est un domaine très actif dans la recherche opérationnelle (Cacchiani, et al., 2014). Cela concerne principalement, le transport ferroviaire de voyageurs. Dans le cas d'occurrence des perturbations, les spécificités du système ferroviaire et de son exploitation, font qu'il est nécessaire de procéder à une replanification séquentielle, comme suit :

1- Replanification de la circulation des trains : le planning de circulation des trains contient les itinéraires des trains (i.e. stations de départ et d'arrivée, et les stations intermédiaires), ainsi que les heures de départ et d'arrivée. Les problématiques soulevées à ce niveau sont très complexes et peu d'algorithmes sont en mesure de considérer cette complexité (Sato, et al., 2013). Ceci oblige les transporteurs à faire appel à des experts, pour procéder aux replanifications, en fonction des perturbations et des différents paramètres. Pour répondre à ce challenge, plusieurs travaux de recherche sont menés à ce niveau, considérant plusieurs types de perturbations et appliqués à diverses configurations de réseaux ferroviaires (lignes uniques, multi-lignes, transport urbain, ou encore à grande vitesse).

   Dans un contexte d'un système ferroviaire performant et durable (i.e. caractérisé par une haute sécurité et accessibilité, une haute performance énergétique et offrant des services fiables et ponctuels), dont l'amortissement des coûts importants qu'il génère, doivent être couverts par une fréquence élevée des trains. Le challenge de la prise en charge des conséquences des perturbations éventuelles est d'autant plus important. Au-delà de la nécessité de fournir une bonne solution lors de la replanification du planning des trains, dans (Törnquist Krasemann, 2012) une heuristique qui fournit des solutions rapidement a été développée, pour répondre aux exigences liées à ce contexte. Dans (Altazin, et al., 2017), la replanification est proposée pour la réduction de la propagation des retards en présence de perturbations. Un PLNE est développé pour réajuster le plan de circulation des trains, en supprimant certains arrêts. L'objectif étant la minimisation du temps de retour au plan de transport théorique, suite à l'occurrence d'une perturbation, tout en réduisant le temps d'attente des passagers. D'autres auteurs ont proposé des heuristiques dans ce même contexte, tels que (Dollevoet & Huisman, 2014) avec un objectif de minimisation du temps de voyage des passagers. Dans (Corman, et al., 2012), les auteurs considèrent un double objectif de minimisation des retards des trains et correspondances ratées des voyageurs.

   Etant difficile d'évaluer l'optimalité des solutions fournies par les heuristiques et les métaheuristiques, certains auteurs ont proposé, dans des contextes spécifiques, des PL en variables mixtes ou des PLNE. L'objectif est de minimiser les retards des trains ou l'insatisfaction des voyageurs, tel que dans (Xie & Li, 2012), où les auteurs ont considéré les données récoltées par les trains précédents à travers un PLNE, pour ajuster au mieux le planning de circulation des trains à venir. Dans (Pellegrini, et al.,

2012), les auteurs ont proposé un PL en variables mixtes pour minimiser les retards des trains. Même chose dans (Rodriguez, 2007), où l'auteur a proposé deux modèles PPC, pour planifier le passage des trains par des nœuds ferroviaires. L'objectif étant de minimiser les retards liés aux conflits.

2- Replanification du matériel roulant (trains, wagons et locomotives) : la recherche au niveau de la replanification des ressources utilisées par le transport ferroviaire, aussi bien matérielles (infrastructure et matériel roulant) qu'humaines, n'est pas aussi dense que dans le cas précédent (Nielsen, et al., 2012). Dans (Huisman, et al., 2005), les auteurs ont fait une revue de littérature des travaux dans ce domaine.

   Un premier modèle commercial a été élaboré pour le compte de la Société Nationale des Chemins de Fer Français « SNCF », par SABRE Technology Solutions (Ben-Khedher, et al., 1998), pour la planification aux niveaux stratégique et tactique, de l'activité de transport ferroviaire à grande vitesse. Dans un premier temps, ce modèle propose des plannings basés sur les prévisions de la compagnie. Dans un second temps, un réajustement des plannings et du matériel roulant utilisé, est effectué en fonction de la variation de la demande effective et des dernières prévisions.

   Dans (Nielsen, et al., 2012), une approche par horizon glissant est proposée, pour la replanification du matériel roulant, dans le cas d'une perturbation majeure, au niveau du système ferroviaire. L'objectif du modèle de replanification développé, étant la minimisation de la déviation dans l'inventaire de tous les types de matériels roulants et dans toutes les stations de la ligne, par rapport à l'inventaire du plan d'activité initial.

3- Replanification du personnel : la replanification du personnel est un axe de recherche assez récent, qui intervient plus, dans les cas de perturbations de type majeur (ou interruption de service). Un premier travail de recherche (Walker, et al., 2005), s'est intéressé à l'établissement des plannings de circulation des trains, en planifiant le matériel roulant et les équipages des trains (organisés par équipe) simultanément, avec un objectif de minimisation des temps morts des trains, durant les heures d'activité de chaque équipage. Un modèle mathématique de type PLNE a été développé pour cela. Puis, lors de l'occurrence de perturbations majeures, un second PLNE, construit sur la base du premier, a été établi avec un objectif de minimisation des changements par rapport au plan initial. Le problème est résolu par une approche de type génération de colonnes.

   Dans un autre article (Veelenturf, et al., 2012), la replanification du personnel se réduit à celle des conducteurs de trains. La replanification à ce niveau, peut nécessiter une replanification au niveau du matériel roulant, voire, au niveau du planning de circulation des trains. En effet, s'il n'y a pas de conducteur pour conduire un train planifié, il sera nécessaire de procéder à un réajustement. Cette étude se focalise sur l'occurrence de perturbations majeures. Pour cela, le modèle de replanification se présente sous la forme d'un PLNE, dont la fonction « objectif » à minimiser, se compose de trois membres : la déviation par rapport au plan des conducteurs initial, les pénalités liées à l'annulation de certaines tâches des conducteurs, ainsi que les pénalités liées aux retards. Le problème est résolu par une approche de type génération de colonnes, couplée à une heuristique lagrangienne. Pour réduire les temps de calcul, seul un nombre limité de trains est considéré lors de chaque replanification.

Dans un autre contexte, considérant des perturbations mineures, durant la période d'exploitation, dans (Carosi, et al., 2015), les auteurs proposent l'utilisation de la recherche tabou, dans la planification des trains en temps réel, dont l'objectif est l'optimisation de la régularité du service. Dans un second temps et pour faire correspondre le planning des trains avec celui du personnel, il est proposé un modèle de replanification du personnel, avec un objectif de minimisation des heures de travail supplémentaire. Ce modèle est résolu par une approche de type génération de colonnes.

Dans le cadre du présent travail, nous allons nous focaliser sur la replanification du service de transport de marchandises par les trains, en milieu urbain. Cependant, nous ne considérons pas la replanification des trains eux-mêmes. Les changements dans le planning des trains (qui dépend du service original de transport de voyageurs) sont des données d'entrées pour notre modèle.

5. Simulation à évènements discrets – ARENA

L'une des interfaces les plus réussies entre la recherche opérationnelle et l'informatique, est la simulation à évènements discrets (Fu, 2002). Comme outil d'aide à la décision, la simulation joue un rôle très important. En effet, cette technique aide à mieux cerner les conséquences des choix potentiels d'actions, afin de mieux les maîtriser et d'améliorer ainsi, le pilotage de la chaine logistique. Dans ce travail, les avantages de l'utilisation de la simulation interviennent à plusieurs niveaux, tels que :

- L'étude de l'intégration d'un nouveau service, dans un système existant, dont une expérimentation sur le terrain serait trop coûteuse.
- Formaliser la dynamique du nouveau service et étudier les flux y afférents.
- L'étude de différents scénarii, traduisant plusieurs stratégies de gestion du nouveau service.
- Le couplage avec des modèles d'optimisation, pour évaluer les performances de la solution optimale au vu de la dynamique du système, en considérant d'autres aspects.
- Utiliser le modèle de simulation pour valider les modèles d'optimisation.
- Les résultats visuels sont un moyen de compréhension et de confiance, qui permettent de faire évoluer le modèle et, par conséquent, le nouveau service proposé.
- Le modèle de simulation est un démonstrateur pour les décideurs des parties prenantes.

5.1. Généralités sur la simulation

On trouve dans la littérature plusieurs définitions de la simulation. Ces définitions sont parfois proches, équivalentes ou complémentaires. Une synthèse de différentes définitions de la simulation est proposée dans (Bakalem, 1996) : *« la simulation est une méthode de mesure et d'étude consistant à remplacer un phénomène ou un système à étudier par un modèle informatique plus simple mais ayant un comportement analogue »*. Ce modèle permet de réaliser des tests, afin de comprendre le comportement dynamique du système étudié. Il est implémenté sur ordinateur, et évolue d'un état vers un autre en fonction de certaines règles de changement d'état bien définies. La simulation est une technique expérimentale appropriée à

l'étude des systèmes de grande taille composés de plusieurs éléments en interaction. Elle permet ainsi, de répondre à certains problèmes, à chaque fois que l'expérimentation en grandeur nature, se révèle impossible et/ou trop coûteuse (Carrie, 1988).

Simuler revient à créer un modèle informatique fidèle au système étudié de par son comportement. A l'aide du modèle défini, on peut mettre au point de véritables plans d'expériences pour faire des tests sur certaines décisions à prendre, certains paramétrages modifiés (Zeigler, 1976). Selon l'article (Pritsker, et al., 1989), l'objectif de la simulation sur ordinateur est de reproduire et anticiper, prédire le comportement dynamique d'un système existant (ou futur), modélisé de façon virtuelle, donc sans aucun risque.

Le processus de simulation détaillé a été décrit dans (Pritsker, 1986), il se compose de dix étapes, suivant un processus incrémental. Ce dernier peut être schématisé à travers la Figure III.7.

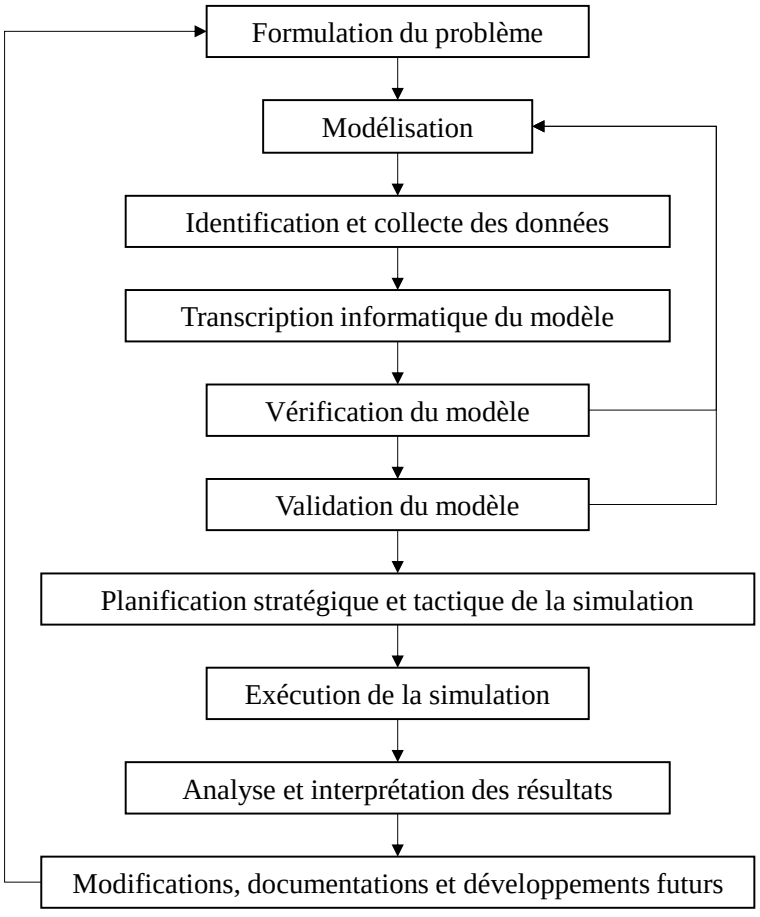

Figure III.7 : Schématisation du processus de simulation (Pritsker, 1986)

5.2. Démarche de réalisation d'un modèle de simulation

La réalisation d'un modèle de simulation requiert au préalable, la définition des objectifs de l'étude, ainsi que des simplifications de la réalité qu'il est raisonnable de faire (Le Moigne, 1990). Réalisé en fonction des objectifs de l'étude, le modèle de simulation permet d'obtenir des résultats numériques (éventuellement statistiques) qui sont ensuite analysés pour permettre la prise de décisions concernant le système réel comme le montre la Figure III.8.

La démarche de réalisation d'un modèle de simulation est structurée en plusieurs étapes, dont le nombre varie d'un auteur à un autre (Le Moigne, 1990), (Anu, 1997), (Vernadat, 1999), etc. Nous proposons de regrouper les différentes étapes, en trois étapes principales :

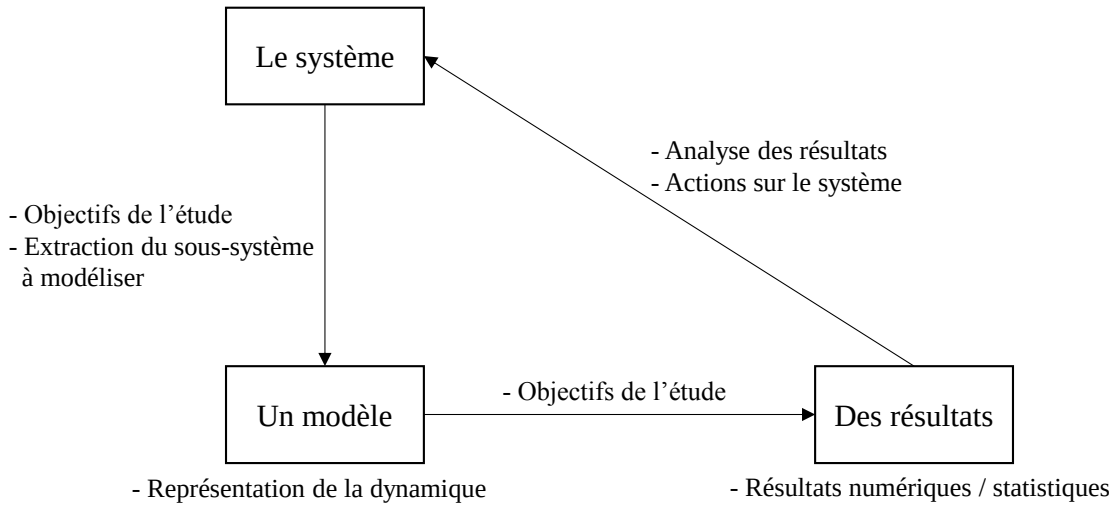

Figure III.8 : Réalisation d'un modèle de simulation

1- La première étape consiste à décrire précisément les entités du modèle et ses règles de gestion. On décrit ensuite la dynamique du système. Dans le cas d'un système simple avec un faible nombre d'entités, une représentation graphique peut s'avérer suffisante. Dans le cas de systèmes plus complexes, on peut être amené à utiliser des outils de formalisation comme les réseaux de Petri. Cette étape de formalisation est très importante car elle évite de remettre en cause le modèle de simulation ultérieurement.
2- La deuxième étape concerne le codage. Elle s'appuie sur la description du système réalisée précédemment.
3- La troisième étape concerne la vérification du programme afin de s'assurer qu'il fait ce que l'on veut qu'il fasse. Il s'agit de vérifier le code et sa logique. La validation du modèle est une étape très importante et délicate car, dans le cas d'un système complexe, il n'existe pas de méthode permettant de s'assurer que le modèle représente correctement le système de départ. On s'appuie alors sur :
    - Des techniques d'animation du modèle montrant les entités en déplacement conformément aux règles de fonctionnement.
    - Des comparaisons avec des résultats mathématiques théoriques lorsque ces derniers sont disponibles.
    - Des comparaisons avec les résultats fournis par des modèles conçus indépendamment.
    - Enfin, dans le cas d'un modèle de système existant, il faut souvent recourir à des spécialistes du domaine pour interpréter les résultats obtenus par simulation, et vérifier qu'ils sont en conformité avec le comportement du système réel.

Cette démarche peut être formalisée par la Figure III.9.

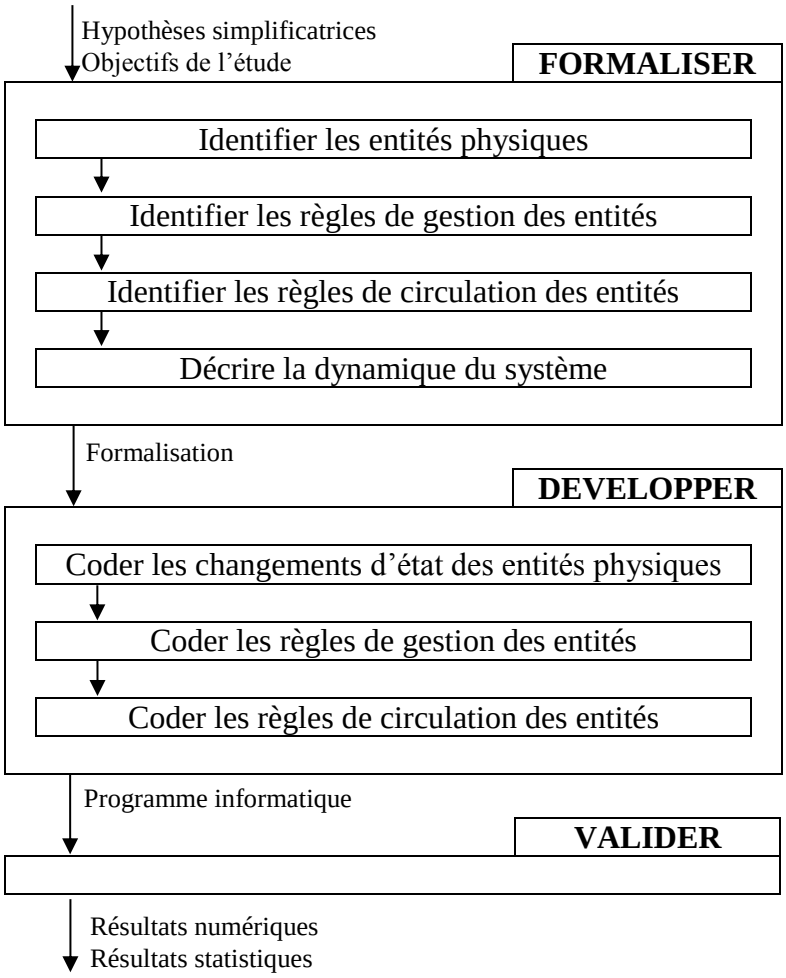

Figure III.9 : Détail de la démarche pour la réalisation d'un modèle de simulation

5.3. Simulation avec ARENA

ARENA est un logiciel dédié de modélisation des flux, en considérant des files d'attente et des ressources, et de simulation des systèmes de production, des procédés, des chaines logistiques, de la distribution, des stocks, etc. Il présente une interface graphique simplifiant son usage, en étant un simulateur de haut niveau, et offre de grandes possibilités de modélisation, en utilisant le langage de simulation SIMAN et le script Microsoft Visual Basic. Les composants d'ARENA sont programmés en SIMAN, ce qui permet de créer des blocs spécifiques si nécessaire et octroie, ainsi, plus de flexibilité au simulateur. Le simulateur comprend des animations graphiques et différents composants de suivi de la simulation et d'analyse des résultats, tels que des tableaux et des courbes. Le logiciel ARENA ainsi que ses différentes fonctionnalités, sont décrites en détail dans (Altiok & Melamed, 2007) et (Rossetti, 2015).

5.3.1. *Eléments d'un modèle de simulation avec ARENA*

Dans un modèle de simulation, nous trouvons :

- **Les entités :** ce sont les premiers éléments à identifier dans un système, pour pouvoir les modéliser. Les entités sont les objets dynamiques de la simulation. Elles sont créées et peuvent être dupliquées, regroupées ou disposées du système. Selon le

système à modéliser, les entités représentent des éléments réels ou fictifs. Elles peuvent modéliser les pannes d'une machine donnée. Comme elles peuvent être assimilées à des pièces mécaniques, des personnes, des encours de production, etc.

- **Les attributs :** ils servent à personnaliser les entités. Un attribut est généralement commun entre toutes les entités, mais sa valeur change d'une entité à une autre. De ce fait, les attributs sont considérés comme des variables locales pour chaque entité.
- **Les variables (globales) :** les variables contiennent des informations sur le système. Nous distinguons deux types de variables globales :
  - Les variables prédéfinies (états des files d'attente, états des machines, etc.).
  - Les variables définies par l'utilisateur (variables spécifiques au modèle).

  Contrairement aux attributs, les variables n'appartiennent pas à une entité spécifique, mais à tout le système.
- **Les ressources :** une ressource est affectée à une entité, afin de la transformer, de lui rajouter de la valeur ou juste pour la faire attendre. Une ressource peut être une personne, une machine, etc. Elle peut contenir un ensemble de serveurs identiques en parallèle.
- **Les files d'attente :** lorsqu'une entité doit attendre la libération d'une unité appartenant à une ressource, elle est placée dans une file d'attente. Cette dernière se caractérise par une capacité et une règle de priorité.
- **Les accumulateurs des statistiques :** ils servent à mesurer les différentes grandeurs et indices de performance. Parmi eux, nous distinguons :
  - Le temps moyen de séjour des entités dans le système.
  - Le temps de séjour des entités dans une file d'attente.
  - Le nombre moyen d'entités dans une file d'attente.
  - Le taux d'occupation d'une ressource.

  Les accumulateurs se basent sur le temps actualisé (TNOW), pour leur calcul.
- **Les évènements :** un évènement est une action de durée nulle, suite à laquelle, les valeurs des attributs, des variables ou des accumulateurs sont changées. Nous citons comme exemples d'évènements :
  - L'entrée de nouvelles entités dans le système.
  - La fin de traitement d'une tâche.

  Dans ARENA, les évènements programmés pour une simulation sont stockés dans un calendrier d'évènements. Pour chaque futur évènement, on enregistre dans le calendrier des évènements :
  - Une identification de l'entité concernée.
  - Le temps d'exécution de l'évènement.
  - Le type d'évènement.

  Les évènements dans le calendrier sont triés selon leurs dates d'occurrence, de manière à ce que les prochains évènements soient les premiers dans la liste. Lorsque la date d'un évènement arrive, son enregistrement est effacé de la liste du calendrier des évènements, et les informations que contient son enregistrement sont utilisées pour produire l'évènement voulu.
- **L'horloge de la simulation :** le temps actualisé de la simulation est contenu dans la variable « horloge de la simulation ». Cette variable ne balaie pas toutes les valeurs du

temps. Elle interagit avec le calendrier des évènements et elle actualise sa valeur uniquement lorsqu'un évènement se produit. Ceci pour éviter un comptage intégral en vain, puisque généralement nous n'avons pas besoin des valeurs du temps entre évènements. Cependant, cette manière de faire présente l'inconvénient de « sauter » les conditions temporelles.

- **Le générateur de nombres aléatoires « GNA » :** l'efficacité du simulateur à donner des résultats valides et proche de la réalité, est directement liée à sa capacité à générer l'aléa. La génération de l'aléa par un programme informatique, permet d'obtenir des nombres pseudo-aléatoires. Le principe consiste à générer des nombres, d'une distribution continue et uniforme entre 0 et 1. Les nombres générés sont des observations. Par la suite, et selon la loi choisie par l'utilisateur, les valeurs correspondantes des variables aléatoires seront calculées par la fonction inverse de la loi dans le cas continu, et par une décomposition en intervalles, des probabilités d'occurrence de chaque valeur que peut prendre la variable aléatoire, dans le cas discret.

  Les dernières versions d'ARENA utilisent un nouveau code de génération de nombres aléatoires, dit CMRG (combined multiple recursive generator). Il utilise un système de flots et sous-flots, pour subdiviser un cycle de génération d'une longueur de $3.1 * 10^{57}$. Lorsque nous aurons à effectuer plusieurs réplications, ARENA bascule automatiquement d'un sous-flot à l'autre, lors de chaque nouvelle réplication.

5.3.2. *L'application Microsoft Visual Basic « VBA » dans ARENA*

La flexibilité d'ARENA est consolidée par l'utilisation du langage de programmation Microsoft Visual Basic. Ce dernier est intégré dans ARENA et peut automatiser certaines de ses fonctionnalités, tout en intégrant la possibilité de faire appel aux fonctions de logiciels tierces.

La programmation en VBA peut être utilisée dans ARENA de deux manières différentes :

- En insérant un bloc VBA dans le modèle logique. Ainsi, à chaque fois qu'une entité passe par ce bloc, le code correspondant s'exécute.
- En l'ouvrant avec l'éditeur de VBA directement à partir d'ARENA, et en implémentant le code au moment opportun (la Figure III.10 montre les moments opportuns d'implémentation d'un code).

Les primitives de VBA sont reliées aux différents évènements apparaissant lors d'une simulation, voire, avant le début de la simulation. Les évènements sont classés en trois grandes familles :

- Les évènements d'avant exécution de la simulation : DocumentOpen, DocumentSave…
- Les évènements d'initialisation de la simulation : RunBegin, RunBeginSimulation…
- Les évènements lors de la simulation : VBA_Block_Fire, OnKeyStroke…

Dans le cadre de ce travail, nous allons principalement utiliser des blocs VBA, associés à la fonction ModelLogic_VBA_Block_Fire. Cette dernière, s'exécute à chaque fois qu'une entité passe par le bloc VBA concerné.

1. RunBegin
   2. ARENA teste et initialise le modèle
   3. RunBeginSimulation
      4. RunBeginReplication
         5. ARENA exécute la réplication
            OnKeyStroke
            UserFunction
      6. RunEndReplication
   7. RunEndSimulation
   8. ARENA termine la simulation
9. RunEnd

Figure III.10 : Evènements VBA lors de l'exécution de la simulation

6. Couplage simulation / optimisation

Le couplage simulation / optimisation est un outil très puissant d'analyse et d'optimisation des systèmes réels complexes (Wang & Shi, 2013). La majeure partie des logiciels de simulation actuels, intègrent des modules d'optimisation, sous formes d'algorithmes heuristiques ou métaheuristiques, tels que : les algorithmes génétiques, la recherche tabou, algorithmes de recherche locale, etc. En ce qui concerne ARENA, il intègre le module d'optimisation OptQuest, qui combine trois métaheuristiques (recherche tabou, réseau de neurones et recherche dispersée), en une seule heuristique de recherche. Ceci a ouvert un champ de recherche très large, présentant un éventail riche de possibilités de couplage entre les deux approches. Le couplage simulation / optimisation a un large champ d'application, dans différents domaines, tels que : la production, la gestion des stocks, le transport, la logistique, les services, la finance, etc. Dans la littérature, on retrouve un nombre important de définitions et de descriptions des processus de couplage (direct / indirect), tel que dans : (Meketon, 1987), (Jacobson & Schruben, 1989), (Fu, 1994), (Carson & Maria, 1997), (Kleijnen & Wan, 2007), (Bierlaire, 2015), (Amaran, et al., 2016). Généralement, le couplage développé dépend grandement des caractéristiques du problème étudié. Dans (Figueira & Almada-Lobo, 2014), les auteurs ont réalisé un vaste travail de synthèse, des différentes démarches qui ont été développées dans ce domaine et ont présenté des classifications à plusieurs échelles. Ces différentes classifications ont été consolidées dans (Borodin, et al., 2017) et structurées en trois types de couplage possibles, comme suit :

- L'optimisation pour la simulation : ce premier type de couplage, consiste à optimiser le système étudié, à travers l'optimisation de son modèle de simulation. Pour cela, la simulation évalue la performance de plusieurs configurations, que propose le modèle d'optimisation, dans le but de trouver celle qui produit la meilleure solution. La démarche est itérative, et teste les propositions du modèle d'optimisation, jusqu'à l'obtention de celle qui fournit des solutions satisfaisantes. Généralement, le processus d'optimisation est interne au logiciel de simulation.

  La Figure III.11 présente la démarche de ce type de couplage.

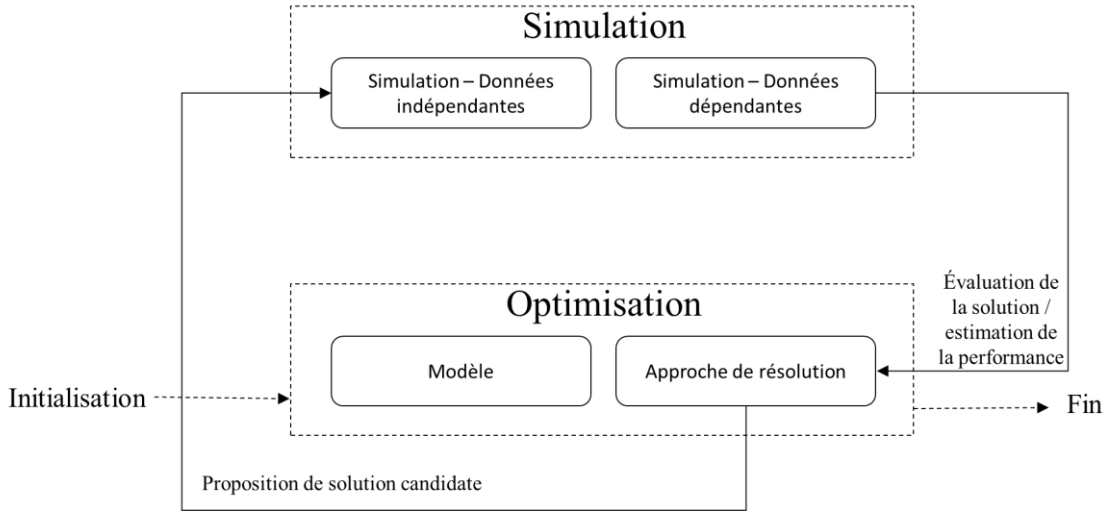

Figure III.11 : Illustration de la démarche de couplage – optimisation pour simulation (Borodin, et al., 2017)

- La simulation pour l'optimisation : ce type de couplage est mis en œuvre, principalement, dans le cadre de la programmation stochastique. L'utilisation de la simulation permet soit :
  - La génération de données suivant des scénarios aléatoires.
  - La reproduction de différentes configurations d'un système sujet à des perturbations. Dans ce cas, la simulation permet la validation des solutions fournies par l'optimisation stochastique, ou bien le réajustement des modèles d'optimisation. Ceci est rendu possible, par la possibilité d'effectuer un nombre important de réplications, sur le modèle de simulation.

  La Figure III.12 présente la démarche de ce type de couplage.

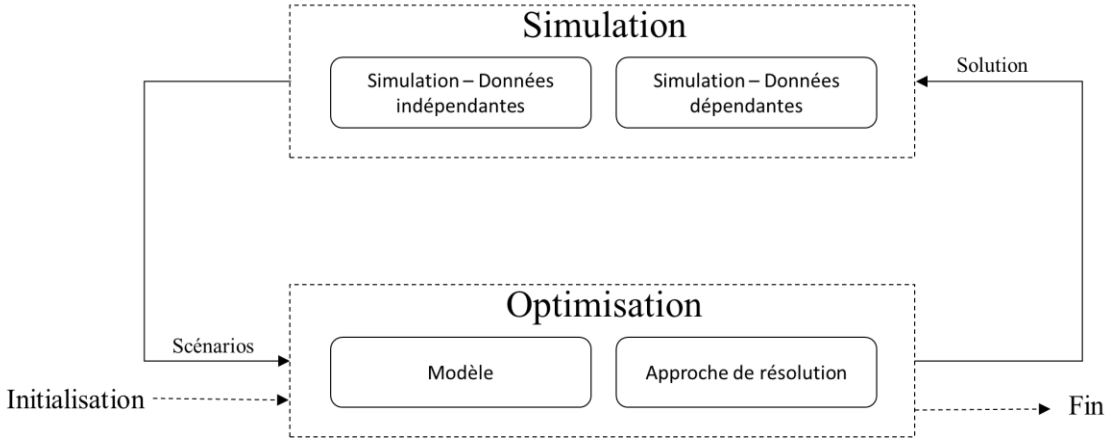

Figure III.12 : Illustration de la démarche de couplage – simulation pour optimisation (Borodin, et al., 2017)

- La simulation et l'optimisation : la simulation permet d'évaluer les performances de plusieurs règles décisionnelles, sans pouvoir construire des solutions, et encore moins de les améliorer. A la différence des deux types de couplage précédents (le premier : l'optimisation fournit les paramètres du modèle de simulation et le second : la simulation fournit les données pour le modèle d'optimisation), ce type de couplage, intègre les deux approches, de manière à optimiser les prises de décision du système