université de BORDEAUX

UNITÉ DE FORMATION Mathématiques & interactions

PROJET PROFESSIONNEL

Problème du Batch-Picking

Equipe :
AJOT EMMANUEL
CHERIF HANEN
GRANDA SEBASTIAN

Enseignants :
CLAUTIAUX FRANÇOIS
SCHLICK CHRISTOPHE

M2 ROAD
Année scolaire 2023 - 2024

1 Description du problème

Le problème du Batch-Picking dans le contexte des opérations d'entrepôt consiste à regrouper un ensemble de commandes en batches, et à déterminer la séquence de positions de stockage à visiter pour chaque batch de manière à minimiser la distance totale parcourue par les préparateurs de commandes. Nous présentons deux formulations pour résoudre ce problème: jointe et séquentielle.

Dans la formulation jointe, les problèmes de batching et de picking sont résolus simultanément comme un problème de routage de véhicules groupés (clustered vehicle routing). Dans la formulation séquentielle, le problème de batching est résolu en premier comme un problème de partitionnement avec des contraintes de capacité. La solution de batching est une entrée pour le problème de picking, qui est résolu comme un ensemble de problèmes de voyageur de commerce (traveling salesman problems, TSP), un pour chaque batch.

La formulation séquentielle est une approximation de l'approche jointe, puisque les optima locaux du problème de picking sont nécessaires pour évaluer la qualité de la solution de batching. Cependant, cette perte de qualité est compensée par la réduction du temps de calcul de l'approche séquentielle. Par conséquent, en abordant ce compromis, nous décrivons un algorithme heuristique dans lequel la formulation séquentielle est utilisée pour obtenir la solution initiale, et une procédure de recherche locale est effectuée pour améliorer la qualité de la solution.

Tout au long de ce document, l'ensemble des commandes est désigné par R, et l'ensemble des articles, par P. Chaque commande $r \in R$ est composée d'un ensemble d'articles $P(r) \subseteq P$ et a un volume v_r et une quantité d'articles q_r. Un article $p \in P$ est associé à un emplacement de stockage dans l'entrepôt telle que plusieurs articles peuvent être stockés dans le même emplacement mais appartenant à des commandes différentes. Les préparateurs de commandes utilisent un véhicule avec une capacité de C_{vol} volume et C_{qty} nombre d'articles.

La disposition de l'entrepôt est modélisée comme un graphe complet $G = (V, E)$, où $V = P \cup \{0, n-1\}$ contient l'ensemble des articles P et le début et la fin du trajet 0 et $n-1$, respectivement, et E représente les déplacements possibles entre les positions. La distance d_{ij} entre les emplacements de stockage des articles $i, j \in V$ est le plus court chemin entre eux, en tenant compte de la disposition de l'entrepôt et des contraintes de mobilité des préparateurs de commandes. Cette hypothèse nous permet de ne pas tenir compte des points de Steiner dans le graphe, qui sont des points intermédiaires ajoutés pour coder la disposition en bloc de l'entrepôt. Soient $\delta^+(S)$ et $\delta^-(S)$ les arcs entrants et sortants d'un sous-ensemble de nœuds $S \subset V$, respectivement, pour les arcs $(i, j) \in V \setminus S$.

La borne inférieure sur le nombre de batches, notée $\underline{K}$, est calculée comme le nombre minimal nécessaire pour que toutes les commandes soient affectées à un batch en fonction des capacités de volume et de quantité avec un écart de α pour cent (1).

$$\underline{K} = (1+\alpha) \cdot \max \left\{ \left\lceil \frac{\sum_{i \in R} v_i}{C_{\text{vol}}} \right\rceil, \left\lceil \frac{\sum_{i \in R} q_i}{C_{\text{qty}}} \right\rceil \right\} \tag{1}$$

Les hypothèses sont résumées comme suit:

- Les positions de stockage des commandes dans un batch peuvent être visitées dans n'importe quel ordre. Autrement dit, nous pouvons mélanger les positions de stockage de

différentes commandes dans le même batch car elles ne nécessitent pas d'être ramassées séquentiellement. En termes pratiques, cela signifierait que le préparateur de commandes est capable de manipuler les supports pour placer les articles dans leurs boîtes respectives.

- Chaque item de chaque commande d'un batch, regroupés dans ce même batch, doivent être visitées pendant la même collecte. En d'autres termes, une commande ne peut pas être scindée dans plusieurs batches et un batch est collecté par un unique préparateur, et ce lors de la même tournée de collecte.
- Toutes les contraintes de mobilité sont prises en compte dans la matrice de distance. Ainsi, la distance entre deux positions est le plus court chemin entre elles, en tenant compte de la disposition de l'entrepôt et des contraintes de mobilité des véhicules.
- Les dates d'échéance et les coûts de stockage ne sont pas pris en compte dans le problème. Il n'est pas intéressant de collecter les commandes le plus tôt possible, mais de minimiser la distance totale de picking.
- La quantité de véhicules est suffisante pour couvrir tous les batches formés (c'est-à-dire qu'au moins les $\underline{K}$ véhicules sont disponibles). Il n'est pas intéressant de minimiser le nombre total de batches, mais la distance totale de picking à la place.
- Les préparateurs de commandes commencent et terminent le picking aux positions 0 et $n-1$, respectivement.
- Les véhicules que les préparateurs de commandes utilisent ont les mêmes propriétés (flotte homogène).
- Seule la distance entre les positions est prise en compte. Tous les temps de service (temps de manipulation) sont négligeables.

1.1 Formulation jointe: problème de routage de véhicules groupés

Étant donné que les articles sont regroupés en commandes, le problème de picking peut être modélisé comme un problème de routage de véhicules groupés (clustered vehicle routing problem, CluVRP) comme le montre [1]. Dans le CluVRP, les articles appartenant à une commande doivent être visités par le même préparateur de commandes, ce qui permet au préparateur de visiter leurs positions dans n'importe quel ordre au sein du batch (c'est-à-dire que les positions de ramassage sont mélangées entre les commandes dans le même batch). Ce dernier est une considération importante car la plupart de la littérature CluVRP suppose des contraintes de cluster rigides, de sorte que le préparateur doit terminer tous les articles d'une commande avant de commencer la suivante, même s'il existe un article plus proche d'une autre commande dans le même batch.

On définit la variable d'affectation $y_{ik} \in \{0,1\}$ comme 1 si l'article $i \in V$ est sélectionné pour le batch $k \in K$, 0 sinon. On définit la variable de séquence $x_{ijk} \in \{0,1\}$ comme 1 si l'article $i \in V$ est visité avant l'article $j \in V$ dans le batch $k \in K$, 0 sinon. Le modèle présenté par [1] peut être adapté comme suit.

$$\min \sum_{k\in K}\sum_{(i,j)\in E} d_{ij}x_{ijk} \tag{2}$$

$$\text{s.t.} \quad \sum_{k\in K} y_{ik} = 1 \quad \forall i \in V \setminus \{0, n-1\} \tag{3}$$

$$\sum_{k\in K} y_{0k} = \sum_{j\in V\setminus\{0,n-1\}}\sum_{k\in K} x_{0jk} \leq \underline{K} \tag{4}$$

$$\sum_{j\in V} x_{ijk} = \sum_{j\in V} x_{jik} = y_{ik} \quad \forall i \in V, \forall k \in K \tag{5}$$

$$\sum_{j\in V\setminus\{0,n-1\}} v_j y_{jk} \leq C_{\text{vol}} \quad \forall k \in K \tag{6}$$

$$\sum_{j\in V\setminus\{0,n-1\}} q_j y_{jk} \leq C_{\text{qty}} \quad \forall k \in K \tag{7}$$

$$\sum_{i\in S}\sum_{j\in V\setminus S} x_{ijk} \geq y_{pk} \quad \forall S \subset V \setminus \{0, n-1\}, \forall p \in S, \forall k \in K \tag{8}$$

$$\sum_{(i,j)\in\delta^+(P(r))}\sum_{k\in K} x_{ijk} = \sum_{(i,j)\in\delta^-(P(r))}\sum_{k\in K} x_{ijk} \geq 1 \quad \forall r \in R \tag{9}$$

$$y_{ik} = y_{jk} \quad \forall i, j \in P(r), \forall r \in R, \forall k \in K \tag{10}$$

$$x_{ijk} \in \{0, 1\} \quad \forall (i, j) \in E, \forall k \in K \tag{11}$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in V, \forall k \in K \tag{12}$$

La fonction objectif (2) minimise la distance totale parcourue par les préparateurs de commandes. Les contraintes (3) garantissent que chaque article est affecté à exactement un batch. Les contraintes (4) assurent que tous les batches commencent au dépôt. Les contraintes (5) garantissent que la conservation du flux est satisfaite pour l'emplacement de stockage de chaque article. Les contraintes (6) et (7) sont les contraintes de capacité de volume et de quantité pour chaque batch, respectivement. Les contraintes (8) éliminent les sous-tours en forçant le flux à être nul pour tous les sous-ensembles d'articles $S \subset V$. Les contraintes (9) forcent les articles du même batch à être visités par le même préparateur de commandes, ce qui permet de visiter leurs emplacements dans n'importe quel ordre. Dernière contrainte (10) se réfère à la condition d'intégrité de la commande pour s'assurer que tous les articles d'une commande sont affectés au même batch.

1.2 Formulation séquentielle: partitionnement d'abord, routage ensuite

La formulation séquentielle est composée d'un problème de partitionnement pour déterminer l'ensemble des batches, et d'un problème de voyageur de commerce pour chaque batch pour déterminer la séquence de positions à visiter. Les avantages de cette approche sont résumés comme suit:

- Le problème de partitionnement peut être simplifié en considérant la distance de Hausdorff entre les commandes au lieu d'évaluer la plus courte séquence de positions de ramassage entre chaque paire de commandes.

- Étant donné une solution pour le problème de batching, le problème de picking peut être résolu indépendamment pour chaque batch.
- Les solutions optimales pour les deux problèmes peuvent être améliorées en utilisant une procédure de recherche locale.

1.2.1 Problème de batching: partitionnement avec contraintes de capacité

Le problème de batching est formulé comme un problème de partitionnement avec contraintes de capacité, dans lequel nous cherchons à minimiser la proximité entre les commandes dans le même batch, tout en maximisant la proximité entre les commandes dans des batches différents. La proximité entre deux commandes est définie comme la distance de Hausdorff entre leurs ensembles respectifs de positions de ramassage [10]. La distance de Hausdorff dirigée $H(i,j)$ entre deux commandes $i,j \in R$ est la distance maximale entre un article de la commande i et son article le plus proche de la commande j (13).

$$H(i,j) = \max_{p \in P(i)} \min_{q \in P(j)} d_{pq} \tag{13}$$

Étant donné que cette distance est asymétrique, c'est-à-dire $H(i,j) \neq H(j,i) \forall i,j \in R$, la proximité t_{ij} entre les commandes $i,j \in R$ est définie comme le maximum entre les deux distances dirigées (14).

$$t_{ij} = \max\{H(i,j), H(j,i)\} \tag{14}$$

Par exemple, supposons une seule allée avec 7 positions de stockage. La commande i a des articles dans les positions $P(i) = \{1,6\}$ et la commande j a des articles dans les positions $P(j) = \{2,3\}$ (18).

$$H(i,j) = \max\{\min\{d_{12}, d_{13}\}, \min\{d_{62}, d_{63}\}\} = \max\{1,3\} = 3 \tag{15}$$
$$H(j,i) = \max\{\min\{d_{21}, d_{26}\}, \min\{d_{31}, d_{36}\}\} = \max\{1,2\} = 2 \tag{16}$$
$$t_{ij} = \max\{H(i,j), H(j,i)\} = \max\{3,2\} = 3 \tag{17}$$
$$\tag{18}$$

Cette métrique est beaucoup moins chronophage à calculer que le plus court chemin entre chaque paire de commandes.

Donc, pour formuler le problème, on définit la variable d'affectation $y_{ik} \in \{0,1\}$ comme 1 si la commande $i \in R$ est affectée au batch $k \in K$, 0 sinon. La variable $x_{ij} \in \{0,1\}$ est 1 si les commandes $i,j \in R$ sont affectées au même batch, 0 sinon. Le problème de batching est formulé comme suit [5].

$$\min \sum_{(i,j)\in R\times R} t_{ij}x_{ij} \tag{19}$$

$$\text{s.t.} \quad \sum_{k\in K} y_{ik} = 1 \quad \forall i \in R \tag{20}$$

$$y_{ik} + y_{jk} \leq 1 + x_{ij} \quad \forall (i,j) \in R \times R, \forall k \in K \tag{21}$$

$$\sum_{i\in R} v_i y_{ik} \leq C_{\text{vol}} \quad \forall k \in K \tag{22}$$

$$\sum_{i\in R} q_i y_{ik} \leq C_{\text{qty}} \quad \forall k \in K \tag{23}$$

$$x_{ij} = x_{ji} \quad \forall (i,j) \in R \times R \tag{24}$$

$$x_{ii} = 0 \quad \forall i \in R \tag{25}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in R \times R \tag{26}$$

$$y_{ik} \in \{0,1\} \quad \forall i \in R, \forall k \in K \tag{27}$$

$$\tag{28}$$

La fonction objectif (19) minimise la proximité totale entre les commandes dans le même batch. Les contraintes (20) garantissent que chaque commande est affectée à exactement un batch. Les contraintes (21) stipulent que si deux commandes sont affectées au même batch, alors l'arc entre elles est un arc intra-batch. Les contraintes (22) et (23) sont les contraintes de capacité de volume et de quantité pour chaque batch, respectivement. Les contraintes (24) et (25) suppriment les solutions symétriques et les boucles, respectivement. Ce problème de partitionnement est connu pour être NP-difficile lorsque le nombre de batches est supérieur à 2 [5].

Attention : d'autres alternatives sont en cours d'évaluation, comme DBSCAN et la contrainte k-means, dans le but de regrouper les commandes.

1.2.2 Problème de picking: voyageur de commerce

Étant donné que les batches sont formés, le problème de déterminer la séquence de positions de stockage à visiter pour chaque batch est équivalent à résoudre un ensemble de problèmes de routage de préparateurs de commandes indépendants (single-picker routing problems, SPRP). Ce problème peut être formulé comme un problème de voyageur de commerce (traveling salesman problem, TSP) comme le montre [9], dans le but de minimiser la distance totale parcourue par le préparateur de commandes.

Nous utilisons la formulation de flux multi-marchandises proposée par [4], dans laquelle les sous-tours sont éliminés en utilisant les contraintes de conservation du flux (c'est-à-dire que le préparateur commence le trajet avec tous les articles et "livre" une unité à chaque position visitée).

La représentation du graphe G_k est similaire à celle présentée dans la formulation jointe, mais l'ensemble de nœuds est défini comme tous les articles du batch donné $\forall k \in K : V_k = \{P(i) : i \in R, y_{ik} = 1\}$, et l'ensemble d'arêtes est défini comme $E_k = \{(i,j) \in V_k \times V_k : i \neq j\}$. On définit la variable $x_{ij} \in \{0,1\}$ comme 1 si l'arc $(i,j) \in E_k$ est sélectionné, 0 sinon. La variable z_{ij}^l représente le nombre d'unités de l'article $l \in V_k \setminus \{0, n-1\}$ qui passent par l'arc $(i,j) \in E_k$. Le TSP associé au batch $k \in K$ peut être adapté de [4] comme suit.

$$\min \sum_{(i,j)\in E_k} d_{ij}x_{ij} \tag{29}$$

$$\text{s.t.} \quad \sum_{j\in V_k} x_{ij} = 1 \quad \forall i \in V_k \tag{30}$$

$$\sum_{i\in V_k} x_{ij} = 1 \quad \forall j \in V_k \tag{31}$$

$$\sum_{j\in V_k\setminus\{0,n-1\}} z^l_{0j} - \sum_{j\in V_k\setminus\{0,n-1\}} z^l_{j0} = -1 \quad \forall l \in V_k \setminus \{0, n-1\} \tag{32}$$

$$\sum_{j\in V_k} z^i_{ij} - \sum_{j\in V_k} z^i_{ji} = 1 \quad \forall i \in V_k \setminus \{0, n-1\} \tag{33}$$

$$\sum_{j\in V_k} z^l_{ij} - \sum_{j\in V_k} z^l_{ji} = 0 \quad \forall i \in V_k \setminus \{0, n-1\}, \forall l \in V_k \setminus \{0, n-1\} : l \neq i \tag{34}$$

$$z^l_{ij} \leq x_{ij} \quad \forall (i,j) \in E_k, \forall l \in V_k \setminus \{0, n-1\} \tag{35}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i,j) \in E_k \tag{36}$$

$$z^l_{ij} \geq 0 \quad \forall (i,j) \in E_k, \forall l \in V_k \setminus \{0, n-1\} \tag{37}$$

$$\tag{38}$$

La fonction objectif (29) minimise la distance totale parcourue par le préparateur de commandes dans le batch $k \in K$. Les contraintes (30) et (31) garantissent que chaque article est visité exactement une fois. Les contraintes (32) garantissent que tous les articles sont collectés à 0 et livrés à un sommet de $V_k \setminus \{0, n-1\}$. Les contraintes (33) garantissent que chaque sommet de $V_k \setminus \{0, n-1\}$ reçoit exactement une unité d'article. Les contraintes (34) indiquent que la conservation du flux est satisfaite à chaque sommet de $V_k \setminus \{0, n-1\}$. La contrainte (35) évite le flux d'articles à travers les arcs qui ne sont pas sélectionnés.

Comme le montre [8], cette formulation nécessite $O(n^3)$ variables et contraintes, et elle fournit une relaxation LP plus forte que celle obtenue par la formulation MTZ (Miller-Tucker-Zemlin), qui est la plus courante pour le TSP.

2 Approche heuristique

L'approche heuristique proposée est composée de deux phases: l'heuristique de construction et la procédure de recherche locale. L'heuristique de construction implémente la formulation séquentielle pour obtenir une solution initiale, et la procédure de recherche locale est effectuée pour améliorer la qualité de la solution. Ces opérateurs cherchent à évaluer les nouveaux chemins entre les commandes qui sont dans les différents batches résultant du problème de partitionnement, de sorte que la distance totale parcourue par les préparateurs de commandes est minimisée. Pour garantir la connexité du voisinage, nous considérons trois opérateurs: swap, insert et destroy-repair. Une recherche de voisinage variable [7] (variable neighborhood search, VNS) sélectionne le meilleur mouvement à chaque itération de l'algorithme en fonction de la stratégie de recherche, soit pour intensifier, soit pour diversifier.

La stratégie de routage combinée [3] (c'est-à-dire S-shape et Largest gap combinés) nous permet d'obtenir la nouvelle séquence de picking étant donné qu'un opérateur est appliqué à

une solution. La solution de première amélioration est prise à chaque itération de l'algorithme, qui arrête la recherche en utilisant un critère de Metropolis pour accepter les mouvements de détérioration [2]. Enfin, une mémoire adaptative de recherche tabou [6] est implémentée pour éviter le cyclisme dans l'espace de recherche et pour diversifier la recherche en forçant l'algorithme à explorer de nouvelles solutions.

2.1 Stratégies de routage

Pour évaluer les mouvements, nous pouvons explorer plusieurs stratégies de routage connues pour ce problème, telles que la S-shape, le plus grand écart ou la stratégie de routage combinée [3].

2.2 Détermination d'une solution initiale

Comme expliqué dans la description du problème, nous choisissons de construire une solution initiale par formulation séquentielle, ce qui nous permettra de partir d'une bonne solution initiale pour notre métaheuristique.

2.3 Méthode de descente de gradient

2.3.1 Définition du voisinage

Dans le but de réaliser une amélioration itérative de notre solution, nous allons effectuer une recherche locale, tout d'abord autour de notre solution initialle, puis autour de notre solution actuelle. Pour cela, il nous faut définir le voisinage d'une solution : le voisinage d'une solution actuelle est l'ensemble de toutes les solutions n'étant qu'à une permutation ou un déplacement près de notre solution actuelle.

2.3.2 Definition des opérateurs

Afin de créer ces solutions voisines, nous utiliserons deux types d'opérateurs :
- "swap", tel que swap(commande1, commande2) échange commande1 et commande2 de leur batch respectif.
- "insert", tel que insert(commande1, k) supprime commande1 de son batch et l'attribue au batch k.
- "Destroy-Repair", tel que destroy-repair(batch1, vect-répartition) supprime toutes les commandes du batch1 et les réaffecte ensuite à d'autres batches selon une répartition définie.

2.3.3 Recherche de Voisinage Variable (VNS) :

On peut ici appliquer la "VNS" pour sélectionner le meilleur mouvement au niveau de chaque itération, l'alternance entre l'intensification (qui est l'exploration approfondie d'une telle zone de l'espace de solution) et ainsi la diversification (qui est l' exploration de nouvelles zones).

Critères de Diversification et d'Intensification

Diversification :

Son application intervient lorsque les améliorations deviennent marginales ou que la recherche stagne. On peut utiliser des opérateurs comme ceux de "destroy-repair" pour la réorganisation de manière significative des batches. On peut changer la méthode de routage et/ou les paramètres de l'algorithme pour qu'on puisse explorer de nouvelles zones de notre espace de solutions.

Intensification :

L'intensification est appliquée après avoir trouvé une solution prometteuse ou une telle amélioration significative. On peut aussi utiliser des opérateurs comme "insert" et "swap" pour faire les ajustements fins sur des batches existants. Ainsi, concentrer la recherche sur les différentes zones voisines de l'actuelle solution.
****

Algorithme VNS pour le Problème de Batch et de Picking

L'algorithme VNS opte à trouver la solution optimale ou quasi-optimale pour le problème de batching et de picking et ceci en explorant systématiquement divers voisinages de l'actuelle solution. L'objectif est la minimisation de la distance totale du picking et ça tout en s'assurant que les substitutions du batch en respectent les contraintes des capacités des véhicules.

Étapes de l'Algorithme :

1. Initialisation :
- On détermine la solution initiale qui est basée sur le processus du batch qui respecte les différentes contraintes de capacité des véhicules.
- On construit un itinéraire de picking initial pour une solution du batch actuel.
- On définit la valeur de la fonction objective qui est basée sur la distance totale du picking de la solution initiale.
- On initialise le nombre d'itérations $t = 0$ et $count = 0$.

2. Amélioration du Batch (Mutation et Échange) :
-On incrémente le nombre d'itérations : $t = t + 1$.
- On applique une mutation à la solution du batch actuel pour la consolidation rapidement les commandes en des batchs, en vérifiant la conformité avec la capacité de véhicules. Si la nouvelle solution est bien valide et ainsi améliore la valeure objective, on l'adopte comme une nouvelle solution actuelle.
- Si la mutation n'apporte pas d'améliorations ou après la mutation, on applique l'échange du batch pour l'exploration d'autres améliorations. L'échange du batch est moins efficace que la mutation mais il peut encore contribuer à trouver des meilleures solutions.

3. Amélioration du Picking (Insertion et Échange) :
- On applique l'insertion du picking pour explorer une telle gamme plus large de solutions, et ceci en recalculant la distance totale du picking pour chaque nouvelle solution.
- Si aucune amélioration n'est trouvée, on procède à l'échange du picking pour peaufiner la

solution, et ceci en recalculant de nouveau la distance totale du picking pour d'éventuelles améliorations.

4. Changement de Voisinage et Perturbation :

- Si la solution n'est pas améliorée (count = 0), on effectue une perturbation sur l'actuelle solution pour échapper aux optima locaux. Cela implique un nombre bien déterminé de mutations basé sur un tel ratio spécifié du nombre de commandes. - Après la perturbation, on retourne à l'étape 2 pour l'exploration de nouveaux voisinages avec des solutions de meilleures potentiels.

5. Fin :

- On vérifie si le critère d'arrêt (par exemple, un nombre maximal d'itérations ou une qualité de solution satisfaisante) est atteinte. Si oui, on termine l'algorithme et on retourne la meilleure solution trouvée.

-Si le critère d'arrêt n'est pas encore atteint, on retourne à l'étape numéro 2 pour continuer le processus de recherche.

Pseudo-code :

Algorithm 1 Algorithme "VNS" pour le problème de batching et picking

0: Initialiser : On construit la solution initiale du "batching" et du "picking", et on calcule la valeur objective initiale.

0: Définir $t = 0$, count $= 0$.

1. Répéter :

   - $t = t + 1$.
   - On applique la mutation et/ou l'échange de batch,on vérifie la capacité du véhicule, pour enfin mettre à jour la solution si améliorée.
   - On applique l'insertion et/ou l'échange de picking,
     on vérifie la capacité du véhicule. On met à jour la solution si améliorée.
     On incrémente count si améliorée.
   - Si count $= 0$, on applique une perturbation à la solution actuelle.

   Jusqu'à ce que le critère d'arrêt soit atteint.

0: Retourner la meilleure solution trouvée.

Cette approche assure que l'algorithme "VNS" explore une large gamme des solutions en alternant entre des différentes structures de voisinage (mutation, insertion et échange) et aussi en appliquant des perturbations pour l'échappement aux optima locaux.

L'algorithme se concentre à la fois sur les differentes phases de batching et de picking, avec des recalculs de distance totale du picking après chacun des changements, dans le but de la minimisation de la distance parcourue et c'est tout en adhérant aux différentes contraintes de capacité des véhicules.

2.3.4 Stratégies de routage :

Pour déterminer les mouvements et chemins empruntés, plusieurs stratégies de routage peuvent être considérées pour ce problème, dans le but de réduire la distance totale de picking.

Des stratégies telles que S-Shape, Largest gap et même une stratégie combinée de ces deux premières sont envisagées. Cette dernière étant une combinaison des deux où, pour chaque allée dans un batch donné, la meilleure des deux stratégies sera utilisée, c'est celle que nous allons utiliser puisque meilleure ou égale au deux stratégies seules.
Nous allons maintenant expliquer exactement en quoi consistera la stratégie combinée ainsi que les stratégies S-Shape et Largest gap appliquées à une allée. Le S-Shape implique de traverser tous les rayons successivement, ignorant ceux sans items à récupérer. Largest gap consiste à parcourir les extrémités de chaque rayon et à faire demi-tour pour maximiser la distance non-parcourue dans chaque rayon.

Détail des stratégies de routage

S-Shape :

La stratégie S-Shaped consiste à traverser chaque allée qui contient des items à récupérer, en ignorant donc celles "vide" où il n'est pas nécessaire de passer.
Bien que cette stratégie ne soit pas très complexe, elle est particulièrement simple dans sa compréhension, son implémentation et son application pratique.

Largest gap :

La stratégie Largest gap quant à elle consiste à repérer la plus grande distance sans item dans chaque allée et à l'éviter en faisant demi-tour à la collecte du dernier item avant celle-ci ; seules les allées aux deux extrémités sont parcourues entièrement. Cela peut par ailleurs impliquer de ne pas entrer dans une allée dans le sens courant si cette plus grande distance se trouve en début d'allée (dans le sens courant, et en fin dans l'autre sens) et donc à en récupérer tous les items en entrant par l'autre sens de l'allée. Bien entendu, si une allée est vide on passe directement à la suivante également.
Bien qu'un peu plus complexe dans son implémentation et son application pratique, cette stratégie s'avère particulièrement utile lorsqu'il y a très peu d'items à récupérer dans une allée.

Stratégie Combinée :

Dans cette stratégie donc, le choix de la stratégie à utiliser pour chaque allée est fonction de la distribution des commandes dans celle-ci, mais également de la distribution des commandes dans la suivante. En effet, selon la disposition des commandes de l'allée suivante non-vide vers une ou l'autre de ses extrémités, il peut être plus intéressant de parcourir l'allée courante dans son entièreté (S-Shape) ou de faire demi-tour (Largest gap). Ainsi, pour chaque allées, selon les dispositions des commandes sur l'allée suivante non-vide ainsi que sur l'allée courante, en prenant en compte la plus grande distance sans item, nous déciderons d'appliquer la stratégie S-Shape ou Largest gap de manière à minimiser la distance totale parcourue et donc se rapprocher d'une solution optimale.

2.3.5 L'algorithme de descente

Algorithm 2 Descente de gradient

```
O = ["swap", "insert", "destroy-repair"], le vecteur des différents opérateurs.
i = 0
repeat
  for Chaque solution du voisinage par opérateur O[i] do
    if La solution est réalisable then
      Appliquer la stratégie de routage combinée et évaluer la solution.
      if La solution est la meilleure solution améliorante then
        La solution est gardée en mémoire, remplaçant la précédente.
      end if
    end if
  end for
  if Il existe une solution voisine améliorante réalisable then
    Cette solution devient la nouvelle solution améliorante.
    i = 0
  end if
else
  i += 1
until Il n'existe aucune solution voisine améliorante réalisable et i==3
```

References

[1] Babiche Aerts, Trijntje Cornelissens, and Kenneth Sörensen. The joint order batching and picker routing problem: Modelled and solved as a clustered vehicle routing problem. *Computers & Operations Research*, 129:105168, 2021.

[2] Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[3] Hadi Charkhgard and Martin Savelsbergh. Efficient algorithms for travelling salesman problems arising in warehouse order picking. *The ANZIAM Journal*, 57(2):166–174, 2015.

[4] A Claus. A new formulation for the travelling salesman problem. *SIAM Journal on Algebraic Discrete Methods*, 5(1):21–25, 1984.

[5] Carlos Eduardo Ferreira, Alexander Martin, C Carvalho de Souza, Robert Weismantel, and Laurence A Wolsey. The node capacitated graph partitioning problem: a computational study. *Mathematical programming*, 81:229–256, 1998.

[6] Fred Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[7] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A Moreno Pérez. *Variable neighborhood search*. Springer, 2019.

[8] Manfred Padberg and Ting-Yi Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52(1-3):315–357, 1991.

[9] André Scholz, Sebastian Henn, Meike Stuhlmann, and Gerhard Wäscher. A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, 253(1):68–84, 2016.

[10] Abdel Aziz Taha and Allan Hanbury. An efficient algorithm for calculating the exact hausdorff distance. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2153–2163, 2015.