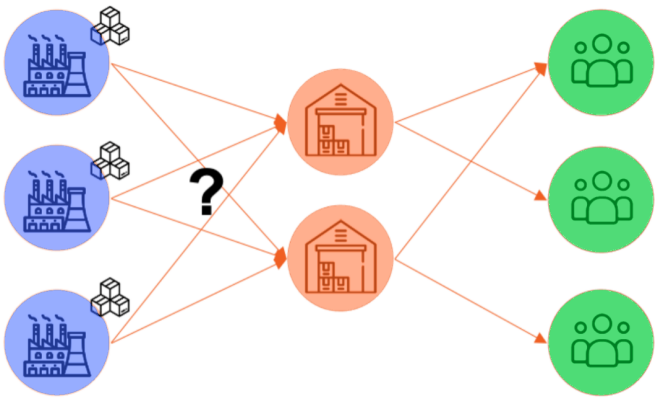

Problème de transport
Optimisation en langage AMPL

Reda Mdair

2025

Table des matières

1 Introduction

Dans un contexte de mondialisation où la compétitivité des entreprises repose sur l'efficacité de leur logistique, l'optimisation des coûts de transport est un enjeu stratégique majeur. Cet enjeu est d'autant plus critique pour les entreprises évoluant sur des marchés de grande envergure, où chaque décision relative à la chaîne d'approvisionnement impacte directement la rentabilité et la réactivité face à la demande.

Ce projet s'inscrit dans une démarche d'optimisation logistique appliquée à une entreprise allemande qui doit approvisionner six clients à partir de deux usines, avec la possibilité de transiter par plusieurs dépôts stratégiquement répartis sur le territoire. L'objectif principal est de minimiser les coûts de transport tout en respectant les contraintes opérationnelles telles que les capacités de production des usines, les limites de stockage des dépôts et les besoins spécifiques des clients.

Nous avons ainsi mis en place une modélisation sous AMPL, intégrant diverses stratégies d'optimisation :

i) une première analyse basée sur le réseau de distribution initial,

ii) une évaluation des coûts et des flux en tenant compte des préférences des clients envers certains fournisseurs,

iii) une réflexion plus avancée sur la restructuration des dépôts, avec l'étude de nouvelles ouvertures, extensions ou fermetures afin d'optimiser les coûts fixes et opérationnels.

Grâce à une approche rigoureuse basée sur la modélisation et l'utilisation du solveur CPLEX, cette étude propose des solutions concrètes et applicables pour améliorer la gestion logistique. Les résultats obtenus montrent comment des choix stratégiques bien pensés peuvent réduire les coûts, optimiser les ressources et améliorer l'efficacité globale du réseau de distribution.

2 Présentation du problème

On considère une entreprise allemande se préparant à approvisionner un marché de grande envergure à six clients sur une période fixée. Pour livrer ces derniers, elle dispose de deux usines (les sources de fabrication) ainsi que de quatre dépôts (espaces de stockage).

— **Usines** : Berlin, Munich.

— **Dépôts** : Hambourg, Francfort, Düsseldorf, Leipzig.

— **Clients** : C1, C2, C3, C4, C5, C6.

Voici les emplacements des usines et des dépôts sur la carte de l'Allemagne.

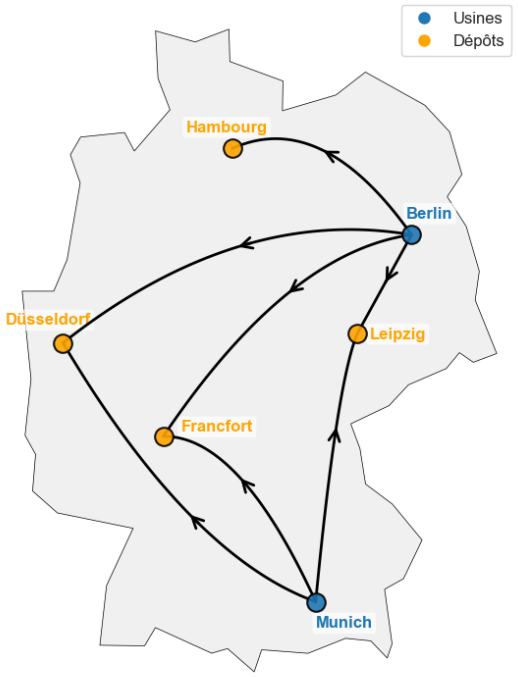

FIGURE 1 – Réseau de transport de l'entreprise entre usines et dépôts

Les livraisons peuvent se faire d'une usine vers un client sans passer par un dépôt ou bien par l'intermédiaire d'un dépôt. Néanmoins, les combinaisons ne sont pas toutes disponibles. On regroupe l'ensemble des coûts de distribution via le tableau qui suit, en marquant d'une case vide les livraisons indisponibles.

Destinataire / *Livreur*	**Hambourg**	**Francfort**	**Düsseldorf**	**Leipzig**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Berlin	47	62	56	38		83	82		85	
Munich		48	60	57	84			78		93
Hambourg					22	35	40	30		31
Francfort					26	31	37	24	29	
Düsseldorf					38		26		33	36
Leipzig						34		28	30	40

TABLE 1 – Tableau des coûts de transport (en €/tonne)

L'ensemble des usines disposent d'une limite de capacité de production. Les dépôts, quant à eux, ont une capacité d'accueil limitée. Nous pouvons regrouper l'ensemble de ces données à travers un seul paramètre, donné par le graphique ci-dessous :

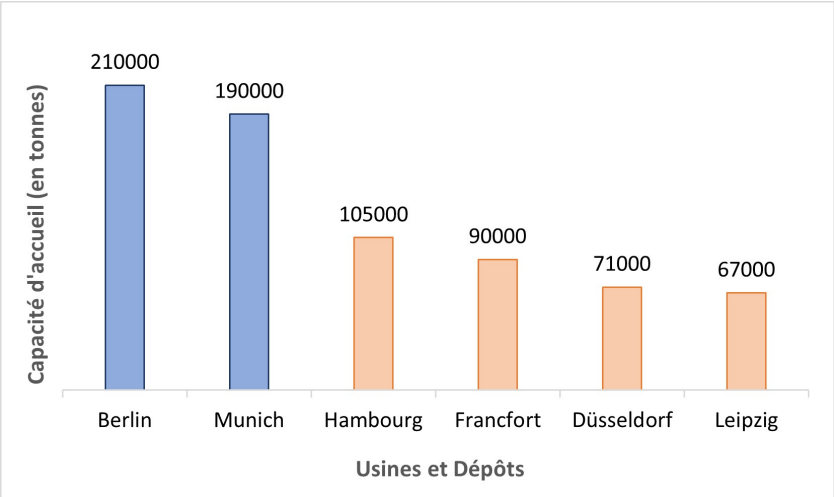

FIGURE 2 – Capacités d'accueil des usines et des dépôts (en tonnes)

Les stocks initiaux des usines et dépôts sont supposés vides. On exige que les produits acheminés vers les dépôts soient entièrement expédiés par la suite (chaque quantité entrante est égale à la quantité sortante pour chaque dépôt).

Concernant les clients, ils disposent de besoins en marchandise à satisfaire. Nous présentons ces données comme suit :

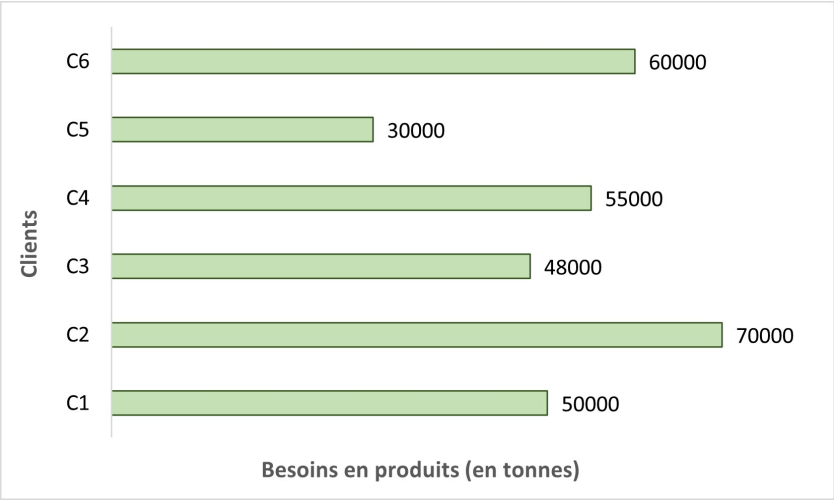

FIGURE 3 – Besoins des clients (en tonnes)

L'objectif est de minimiser les coûts de transport tout en répondant aux demandes des clients et aux limites d'exportation des usines et dépôts.

3 Résolution avec les dépôts initiaux

3.1 Modélisation sous AMPL

Nous présentons la formulation des données initiales dans le fichier .mod. Celui-ci contient notamment les ensembles indiqués par les commandes **set**, les paramètres désignés par **param** et les variables décrites par **var**.

Ensembles

Dans ce projet, nos ensembles regroupent les usines, les dépôts ainsi que les clients.

```
set USINES;
set DEPOTS;
set CLIENTS;
```

Paramètres

Les paramètres sont les bases numériques du modèle. Ils nous permettront de traduire les caractéristiques opérationnelles et économiques en données exploitables. La liste de nos paramètres se forme ainsi :

— **prix_route** : Coût de transport selon la route de livraison (avec un 0 en cas de chemin indisponible).

```
param prix_route{USINES union DEPOTS , CLIENTS union DEPOTS} >= 0;
```

— **capacite_accueil** : Maximum d'importation (et d'exportation) des produits pour les usines et dépôts (en tonnes).

```
param capacite_accueil{USINES union DEPOTS} >= 0;
```

— **besoin_client** : Quantité de produits demandés par un client (en tonnes).

```
param besoin_client{CLIENTS} >= 0;
```

Variables

Par la suite, on peut définir nos variables (inconnues) que le solveur va calculer dans le but d'optimiser les coûts de l'entreprise. Elles se composent de :

— **quantite_UC** : Quantité acheminée d'une usine vers un client (en tonnes).

— **quantite_UD** : Quantité acheminée d'une usine vers un dépôt (en tonnes).

— **quantite_DC** : Quantité acheminée d'un dépôt vers un client (en tonnes).

```
var quantite_UC{u in USINES, c in CLIENTS} >= 0;
var quantite_UD{u in USINES, d in DEPOTS} >= 0;
var quantite_DC{d in DEPOTS, c in CLIENTS} >= 0;
```

Fonction objectif

L'objectif est de **minimiser** les dépenses de l'entreprise, définies comme la somme des quantités acheminées affectées par les coûts de transport des routes choisies :

```
minimize cout_total:
sum{u in USINES, c in CLIENTS} prix_route[u,c] * quantite_UC[u,c]
+ sum{u in USINES, d in DEPOTS} prix_route[u,d] * quantite_UD[u,d]
+ sum{d in DEPOTS, c in CLIENTS} prix_route[d,c] * quantite_DC[d,c];
```

Contraintes

Les contraintes garantissent que les solutions respectent les capacités et les restrictions du système. On les regroupe en 4 classes :

— **Respect des capacités des usines et des dépôts** : Les quantités sortantes des usines et entrantes aux dépôts ne doivent pas dépasser les limites fixées.

```
subject to Capacite_Usines{u in USINES}:
sum{c in CLIENTS} quantite_UC[u,c] + sum{d in DEPOTS} quantite_UD[u,d] <=
    capacite_accueil[u];
subject to Capacite_Depots{d in DEPOTS}:
sum{u in USINES} quantite_UD[u,d] <= capacite_accueil[d];
```

— **Satisfaction des besoins des clients** : Chaque client doit se voir recevoir la quantité de produit demandée.

```
subject to Besoin_Clients{c in CLIENTS}:
sum{u in USINES} quantite_UC[u,c] + sum{d in DEPOTS} quantite_DC[d,c] =
    besoin_client[c];
```

— **Exclusion des routes indisponibles** : Chaque chemin marqué d'un coût de 0 dans le paramètre *prix_route* doit être ignoré dans l'optimisation.

```
subject to Route_Indisponible_Usine_Client{u in USINES, c in CLIENTS:
    prix_route[u,c] == 0}:
quantite_UC[u,c] = 0;
subject to Route_Indisponible_Usine_Depot{u in USINES, d in DEPOTS :
    prix_route[u,d] == 0}:
quantite_UD[u,d] = 0;
subject to Route_Indisponible_Depot_Client{d in DEPOTS, c in CLIENTS:
    prix_route[d,c] == 0}:
quantite_DC[d,c] = 0;
```

— **Liquidation des stocks des dépôts** : La quantité entrante pour un dépôt doit être totalement livrée par la suite.

```
subject to Liquidation_Stock{d in DEPOTS}:
sum{u in USINES} quantite_UD[u,d] = sum{c in CLIENTS} quantite_DC[d,c];
```

3.2 Interprétation des résultats

Grâce au module **solve cplex**, nous pouvons désormais résoudre le problème. En sortie, on s'intéresse au coût optimal donné par l'appellation **cout_total**, ainsi que l'ensemble des quantités de marchandises reliant les usines aux dépôts et clients, et les dépôts aux clients. Voici la commande à exécuter dans le fichier **.run** :

```
option solver cplex;
solve;

display quantite_UC; display quantite_UD; display quantite_DC;
display cout_total;
```

Nous obtenons alors un coût optimal de **23 420 000 €** avec un schéma de distribution donné par les tableaux suivants :

Usine	Client	Quantité
Berlin	C1	0
Berlin	C2	0
Berlin	C3	38000
Berlin	C4	0
Berlin	C5	0
Berlin	C6	0
Munich	C1	0
Munich	C2	0
Munich	C3	0
Munich	C4	3000
Munich	C5	0
Munich	C6	0

TABLE 2 – Quantités par livraison optimales entre usines et clients (en tonnes)

Usine	Dépôt	Quantité
Berlin	Düsseldorf	0
Berlin	Francfort	0
Berlin	Hambourg	105000
Berlin	Leipzig	67000
Munich	Düsseldorf	10000
Munich	Francfort	90000
Munich	Hambourg	0
Munich	Leipzig	0

TABLE 3 – Quantités par livraison optimales entre usines et dépôts (en tonnes)

Dépôt \ Client	C1	C2	C3	C4	C5	C6
Hambourg	45 000	0	0	0	0	60 000
Francfort	5 000	33000	0	52 000	0	0
Düsseldorf	0	0	10 000	0	0	0
Leipzig	0	37 000	0	0	30 000	0

TABLE 4 – Quantités par livraison optimales entre dépôts et clients (en tonnes)

Pour une meilleure visualisation, nous pouvons résumer les tableaux précédents via le diagramme de flux de transport suivant :

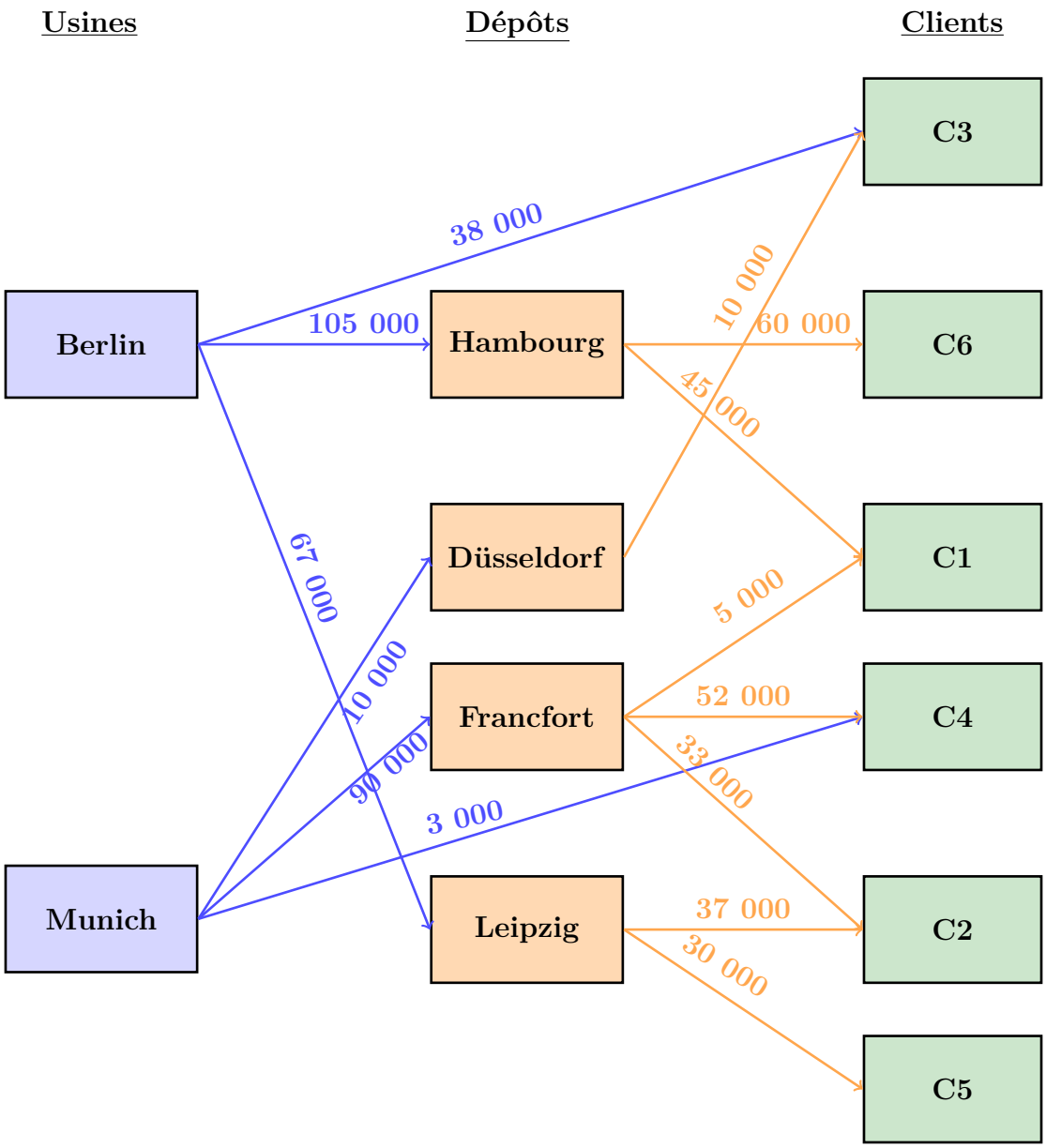

FIGURE 4 – Schéma de distribution minimisant les coûts de transport

On constate que l'usine la plus active est celle de Berlin, expédiant le total de sa limite de production vers les dépôts et clients. Ceci s'explique notamment par des tarifs avantageux (comme ses routes avec Leipzig et Hambourg). L'usine de Munich, quant à elle, achemine un total de 100 000 tonnes par rapport à son plafond de 190 000 tonnes. Les 90% de ses livraisons se font vers Francfort, soit une exploitation logique vue la proximité entre les deux villes.
Concernant les dépôts, ils sont tous pleinement exploités à l'exception de celui de Düsseldorf, recevant uniquement 10 000 tonnes de marchandise sur les 71 000 autorisés. La raison est que cette ville est simultanément éloignée des deux usines, et ne bénéficie donc pas d'un prix de transport bas avec l'une des usines.
Du point de vue des clients, il est intéressant de noter que C3 et C4 reçoivent leurs produits à la fois d'une usine et d'un dépôt. Cela montre qu'une solution optimale à ce problème nécessite d'évaluer tout type de combinaison de transport possible.

3.3 Ajout des préférences des clients en contrainte

Supposons désormais que les clients possèdent plus ou moins des préférences en termes de fournisseurs, données par :

- C1 : le dépôt de Hambourg.
- C2 : les dépôts de Francfort et Leipzig.
- C3 : l'usine de Berlin et le dépôt de Francfort.
- C4 : l'usine de Munich et le dépôt de Francfort.
- C5 : pas de préférence.
- C6 : l'usine de Munich et le dépôt de Düsseldorf.

On doit alors ajouter un paramètre sur ces préférences, pouvant être caractérisé de manière binaire. Le fichier .mod se met à jour avec :

```
param preference_client{CLIENTS, USINES union DEPOTS} binary;
```

Sachant que le paramètre **preferences** est une variable binaire, l'algorithme va forcer un maximum de préférences à être actives.
Dans le fichier .dat, ce paramètre s'adapte à nos données comme suit :

```
param preference_client :
            Berlin  Munich  Hambourg  Francfort  Dusseldorf  Leipzig :=
    C1        0       0        1         0          0          0
    C2        0       0        0         1          0          1
    C3        1       0        0         1          0          0
    C4        0       1        0         1          0          0
    C5        1       1        1         1          1          1
    C6        0       1        0         0          1          0 ;
```

La contrainte à ajouter dans le fichier .mod peut alors se formuler ainsi :

```
subject to Respect_Preference_Client{c in CLIENTS, s in USINES union DEPOTS}:
if preference_client[c,s] == 0 then
    (sum{u in USINES: u == s} quantite_UC[u,c] + sum{d in DEPOTS: d == s}
    quantite_DC[d,c]) = 0;
```

En incluant les préférences des clients, nous obtenons un coût optimaml de **24 192 000 €** avec le schéma de distribution qui suit :

Usine	**Client**	**Quantité**
Berlin	C1	0
Berlin	C2	0
Berlin	C3	48000
Berlin	C4	0
Berlin	C5	0
Berlin	C6	0
Munich	C1	0
Munich	C2	0
Munich	C3	0
Munich	C4	0
Munich	C5	0
Munich	C6	15000

TABLE 5 – Quantités optimales entre usines et clients en incluant les préférences des clients

Usine	**Dépôt**	**Quantité**
Berlin	Düsseldorf	45000
Berlin	Francfort	0
Berlin	Hambourg	50000
Berlin	Leipzig	67000
Munich	Düsseldorf	0
Munich	Francfort	88000
Munich	Hambourg	0
Munich	Leipzig	0

TABLE 6 – Quantités optimales entre usines et dépôts en incluant les préférences des clients

Dépôt \ **Client**	C1	C2	C3	C4	C5	C6
Hambourg	50 000	0	0	0	0	0
Francfort	0	33000	0	55000	0	0
Düsseldorf	0	0	0	0	0	45 000
Leipzig	0	37000	0	0	30000	0

TABLE 7 – Quantités optimales entre dépôts et clients en incluant les préférences des clients

En voici une représentation en diagramme de flux de transport :

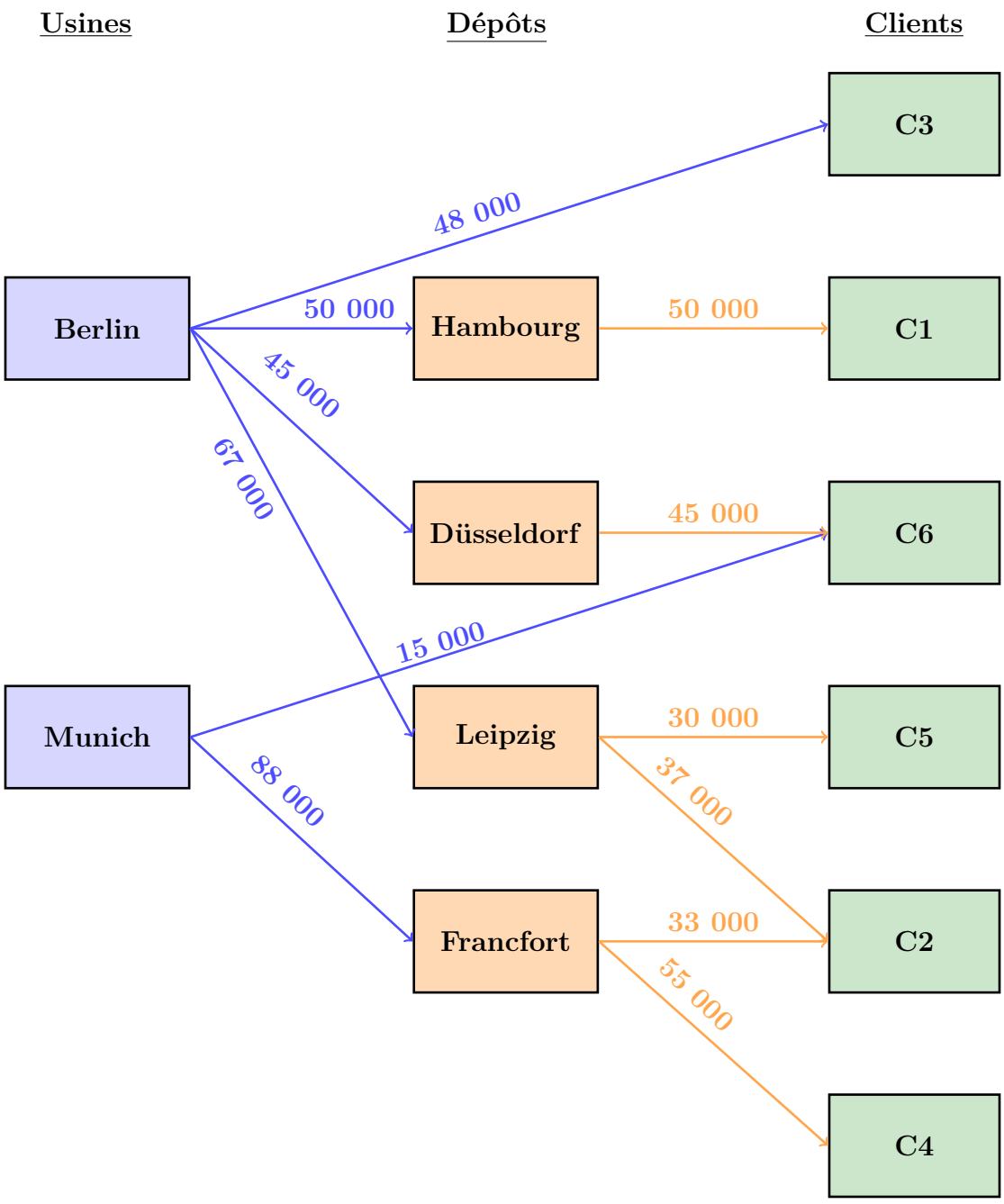

FIGURE 5 – Schéma de distribution optimal incluant les préférences des clients

Le coût total, de 24 192 000 €, est évidemment plus élevé que le coût précédent (23 420 000 €) puisqu'on a ajouté une contrainte en plus.
La principale différence avec le schéma sans les préférences est la meilleure exploitation du dépôt de Düsseldorf. Précédemment, ce dernier n'a que très peu livré. Cette fois-ci, le client C6 a forcé son utilisation en l'incluant dans ses préférences. On peut également noter que les usines produisent les mêmes quantités que précédemment, à savoir Berlin avec ses 210 000 tonnes et Munich avec ses 103 000

tonnes. Néanmoins, les chemins de livraison entre usine et dépôt/client ont changé.

Remarque :

Pour la contrainte de préférences des clients, il aurait tout à fait été possible qu'elle soit incompatible. Par exemple, si le client C2 (qui demande 70 000 tonnes) avait exigé uniquement le dépôt de Leipzig (capacité d'accueil de 67 000 tonnes), on n'aurait pas pu satisfaire la préférence de ce client.
Tout de même, il aurait été intéressant d'étudier comment maximiser au mieux les préférences des clients (c'est-à-dire satisfaire un maximum de préférences) tout en minimisant le coût. Pour cela, on aurait rajouté une seconde fonction objective qui précède celle associée à la minimisation du coût :

```
maximize nombre_preferences_clients:
    sum{u in Usines, c in Clients} preference_client[c,u] * quantite_UC[u,c]
   +sum{d in Depots, c in Clients} preference_client[c,d] * quantite_DC[d,c];
```

4 Nouvelles options de dépôts

Dans cette partie, on réfléchit à un éventuel réarrangement des dépôts afin d'effectuer plus d'économies sur le coût total. Voici les pistes étudiées de l'entreprise :

— ouvrir éventuellement deux nouveaux dépôts à Stuttgart et à Dortmund.

— agrandir le dépôt de Leipzig.

— ne pas maintenir plus de quatre dépôts ouverts. Si ceux de Stuttgart et Dortmund ouvrent, alors ce sont ceux de Düsseldorf et Francfort qui doivent fermer.

Les coûts de livraisons incluant les éventuels nouveaux dépôts sont donnés par le tableau suivant :

Destinataire / *Livreur*	**Stuttgart**	**Dortmund**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Berlin	68	53						
Munich	41	61						
Stuttgart			30	26	29	38	23	34
Dortmund			28	27	24	32	31	25

TABLE 8 – Tableau des coûts de transport (en €/tonne)

On dispose également des tarifs et des débits suivants :

— l'ouverture d'un dépôt à Stuttgart coûte 285 000 € pour une capacité d'accueil de 60 000 tonnes.

— l'ouverture d'un dépôt à Dortmund coûte 175 000 € pour une capacité d'accueil de 38 000 tonnes.

— l'extension du dépôt de Leipzig coûte 75 000 € pour une capacité additionnelle de 25 000 tonnes.

— les fermetures des dépôts de Francfort et Düsseldorf feraient économiser respectivement 94 000 € et 79 000 €.

4.1 Modélisation sous AMPL

Ensembles

Dans cette nouvelle partie, on reprend les mêmes ensembles que précédemment avec une mise à jour de l'ensemble des dépôts :

```
# Fichier .dat
set USINES := Berlin Munich;
set DEPOTS := Hambourg Francfort Dusseldorf Leipzig Stuttgart Dortmund;
set CLIENTS := C1 C2 C3 C4 C5 C6;
```

Paramètres

Concernant les paramètres, on met à jour ceux des prix de livraisons (via **prix_route**), des capacités de départ des dépôts/usines (via **capacite_depart**). Celui des besoins des clients (nommé **besoin_client**) reste inchangé. On vient ensuite ajouter trois paramètres :

— **cout_ouverture** : intègre les coûts d'ouverture des nouveaux dépôts.

— **cout_extension** : contient le coût pour l'extension du dépôt de Leipzig.

— **economie_fermeture** : intègre les prix d'économies pour la fermeture des dépôts de Francfort et de Düsseldorf.

```
param cout_ouverture{DEPOTS} >= 0;
param economie_fermeture{DEPOTS} >= 0;
param cout_extension{DEPOTS} >= 0;
```

Variables

Comme il est toujours question de chercher les quantités de produits optimales à acheminer par chemin, nous conservons nos trois variables précédentes.
Afin de prendre des décisions sur l'ouverture ou la fermeture de dépôt, ainsi que sur l'extension du dépôt de Leipzig, nous définissons cinq variables binaires supplémentaires.

```
var quantite_UC{u in USINES, c in CLIENTS} >= 0;
var quantite_UD{u in USINES , d in DEPOTS} >= 0;
var quantite_DC{d in DEPOTS, c in CLIENTS} >= 0;

var decision_Stutt binary := 0;
var decision_Dort binary := 0;
var decision_Leip binary := 0;
var decision_Duss binary := 1;
var decision_Franc binary := 1;
```

Ces variables binaires valent initialement 0 ou 1 selon l'état initial du dépôt. Pour ceux de Stuttgart et Dortmund, ils sont pour le moment (avant application de l'optimisation) fermés donc les variables associées valent 0. Concernant Leipzig, il est actuellement non agrandi (variable initialement à 0). Les dépôts de Düsseldorf et Francfort sont supposés ouverts donc leurs variables valent actuellement 1.
Suite à l'algorithme d'optimisation, ces variables vont soit rester au même état, soit basculer d'état. Les nouvelles valeurs des variables nous permettront d'en conclure quant aux décisions d'ouverture, de fermeture et d'extension des dépôts concernés.

Fonction objectif

Notre fonction d'intérêt va chercher à minimiser le coût total tout en prenant en compte des éventuels coûts d'ouverture/extension des dépôts ainsi que des éventuelles économies de fermeture.

```
minimize cout_total_avec_changement_depots:
    decision_Stutt * cout_ouverture["Stuttgart"]
    + decision_Dort * cout_ouverture["Dortmund"]
    + decision_Leip * cout_extension["Leipzig"]
    - (1 - decision_Duss) * economie_fermeture["Dusseldorf"]
    - (1 - decision_Franc) * economie_fermeture["Francfort"]
    + sum{u in USINES, c in CLIENTS} prix_route[u,c] * quantite_UC[u,c]
    + sum{u in USINES, d in DEPOTS} prix_route[u,d] * quantite_UD[u,d]
    + sum{d in DEPOTS, c in CLIENTS} prix_route[d,c] * quantite_DC[d,c];
```

La particularité très intéressante de cette fonction est qu'elle effectue la minimisation du coût tout en prenant une décision sur chacune des variables binaires.

Contraintes

Les contraintes à ajouter aux précédentes sont les suivantes :

— capacité de 20 000 tonnes additionnelles au dépôt de Leipzig **en cas d'extension**.

— maximum de 4 dépôts ouverts.

— quantités sortantes des dépôts de Francfort, Düsseldorf, Stuttgart et Dortmund inférieures à leur capacité **en cas d'ouverture/maintien de ces dépôts** (sinon égales à 0).

```
subject to Capacite_Depots{d in (DEPOTS diff {"Leipzig"})}:
sum{c in CLIENTS} quantite_DC[d,c] <= capacite_accueil[d];

subject to Capacite_Leipzig:
sum{c in CLIENTS} quantite_DC["Leipzig",c]
<= capacite_accueil["Leipzig"] + decision_Leip * 25000;

subject to Nombre_Max_Depots:
decision_Stutt + decision_Dort + decision_Franc + decision_Duss <= 2;

subject to Capacite_Francfort:
sum{c in CLIENTS} quantite_DC["Francfort", c] <= decision_Franc *
    capacite_accueil["Francfort"];
```

```
subject to Capacite_Dusseldorf:
sum{c in CLIENTS} quantite_DC["Dusseldorf", c] <= decision_Duss *
    capacite_accueil["Dusseldorf"];

subject to Capacite_Stuttgart:
sum{c in CLIENTS} quantite_DC["Stuttgart", c] <= decision_Stutt *
    capacite_accueil["Stuttgart"];

subject to Capacite_Dortmund:
sum{c in CLIENTS} quantite_DC["Dortmund", c] <= decision_Dort *
    capacite_accueil["Dortmund"];
```

4.2 Interprétation des résultats

Après exécution du code par l'intermédiaire du solver **cplex**, on obtient les décisions suivantes sur les variables binaires :

- *decision_Stutt* = 1
- *decision_Dort* = 0
- *decision_Franc* = 1
- *decision_Duss* = 0
- *decision_Leip* = 1

La solution optimale induit donc une ouverture du dépôt de Stuttgart, une non prise en compte de celui de Dortmund, une extension du dépôt de Leipzig, la conservation du dépôt de Francfort et la fermeture de celui de Düsseldorf. Le coût optimal est alors de **22 752 000 €**, soit une économie de 668 000 € par rapport au schéma avec les dépôts initiaux.
Ci-dessous le schéma de distribution détaillé :

Usine	Client	Quantité
Berlin	C1	0
Berlin	C2	0
(...)	(...)	0
Berlin	C6	0
Munich	C1	0
Munich	C2	0
(...)	(...)	0
Munich	C6	0

TABLE 9 – Quantités optimales toutes nulles entre usines et clients (nouvelles options de dépôts)

Usine	Dépôt	Quantité
Berlin	Francfort	0
Berlin	Hambourg	105000
Berlin	Leipzig	92000
Berlin	Stuttgart	0
Munich	Francfort	56000
Munich	Hambourg	0
Munich	Leipzig	0
Munich	Stuttgart	60000

TABLE 10 – Quantités optimales entre usines et dépôts (nouvelles options de dépôts)

Client / Dépôt	C1	C2	C3	C4	C5	C6
Francfort	5000	0	0	51000	0	0
Hambourg	45000	0	0	0	0	60000
Leipzig	0	58000	0	4000	30000	0
Stuttgart	0	12000	48000	0	0	0

TABLE 11 – Quantités optimales entre dépôts et clients (nouvelles options dépôts)

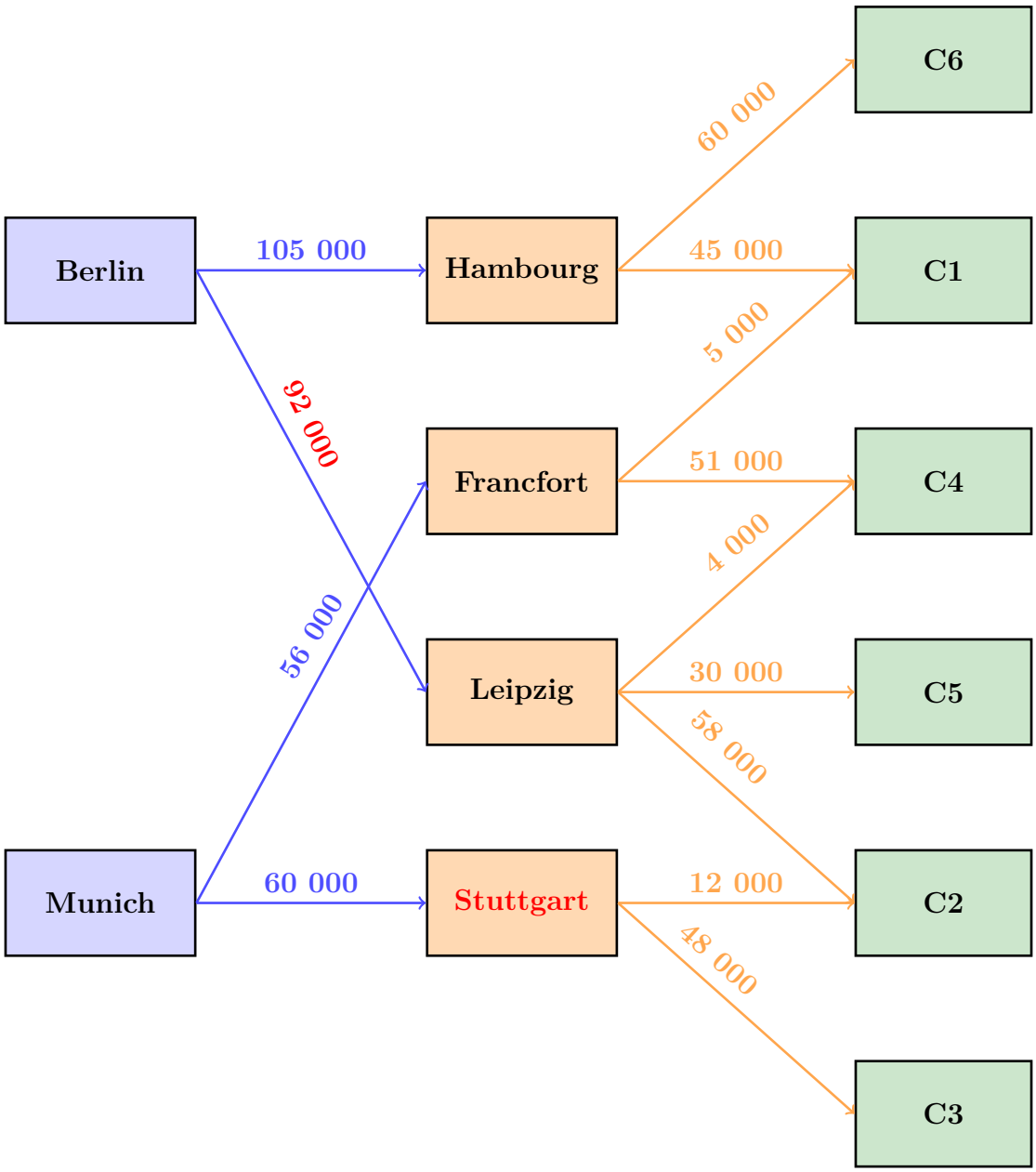

FIGURE 6 – Diagramme associé au schéma optimal avec nouvelles options de dépôts (ouverture Stuttgart, fermeture Düsseldorf et agrandissement Leipzig)

Le solveur a trouvé une rentabilité en celui de Stuttgart, ayant un coût de liaison attractif avec l'usine de Munich. Afin de conserver exactement 4 dépôts actifs, l'optimisation a été simultanément compensée avec la fermeture de celui de Düsseldorf. Ce dernier était le seul non pleinement exploité et ce choix de fermeture semble cohérent avec les attentes.
Concernant le dépôt de Leipzig, il a été jugé utile d'investir dans son extension, permettant une nouvelle capacité d'accueil de 92 000 tonnes. Ceci a permis notamment d'alléger la charge du dépôt de Francfort (90 000 tonnes auparavant contre 56 000 tonnes désormais).
Pour finir, on peut noter que ces changements ont garanti des livraisons 100% usine-dépôt-client, rejetant celles des usines directement vers les clients (assez coûteuses).

4.3 Ajout des préférences des clients en contrainte

Reprenons l'optimisation sous nouvelles options de dépôts mais en incluant désormais les préférences des clients envers les fournisseurs. On commence par mettre à jour les valeurs du paramètre **preference_client** dans le fichier .dat. Le seul client à n'avoir pas exprimé de préférences est C5. On lui attribue alors des valeurs de 1 par défaut envers les potentiels nouveaux dépôts de Stuttgart et Dortmund.

```
# Fichier .dat
param preference_client :
        Berlin  Munich  Hambourg  Francfort  Dusseldorf  Leipzig  Stuttgart
    Dortmund :=
    C1        0        0         1          0           0        0        0
    C2        0        0         0          1           0        1        0      0
    C3        1        0         0          1           0        0        0      0
    C4        0        1         0          1           0        0        0      0
    C5        1        1         1          1           1        1        1      1
    C6        0        1         0          0           1        0        0      0
    ;
```

Les contraintes à ajouter sont données par :

```
subject to Respect_Preferences{c in CLIENTS, s in USINES union DEPOTS}:
if preference_client[c,s] == 0 then
    (sum{u in USINES: u == s} quantite_UC[u,c] + sum{d in DEPOTS: d == s}
    quantite_DC[d,c]) = 0;
```

L'exécution du programme fournit alors un coût optimal de **24 058 000 €**, soit une économie de 134 000 € par rapport au schéma avec dépôts initiaux et préférences des clients.

Les variables binaires pour ce cas de figure deviennent :

- $decision\_Stutt = 0$
- $decision\_Dort = 0$
- $decision\_Franc = 1$
- $decision\_Duss = 0$
- $decision\_Leip = 1$

L'optimisation exige donc de passer à 3 dépôts actifs, avec la fermeture de celui de Düsseldorf et l'absence d'investissement dans ceux de Stuttgart et Dortmund. L'extension du dépôt de Leipzig a de nouveau été mise en œuvre. La disposition du système est donnée ci-dessous :

Usine	**Client**	**Quantité**
Berlin	C1	0
Berlin	C2	0
Berlin	C3	48000
Berlin	C4	0
Berlin	C5	0
Berlin	C6	0
Munich	C1	0
Munich	C2	0
Munich	C3	0
Munich	C4	0
Munich	C5	0
Munich	C6	60000

TABLE 12 – Quantités optimales entre usines et clients (nouvelles options de dépôts avec préférence des clients)

Usine	**Dépôt**	**Quantité**
Berlin	Francfort	0
Berlin	Hambourg	50000
Berlin	Leipzig	92000
Munich	Francfort	63000
Munich	Hambourg	0
Munich	Leipzig	0

TABLE 13 – Quantités optimales entre usines et dépôts (nouvelles options de dépôts avec préférence des clients)

Client / **Dépôt**	C1	C2	C3	C4	C5	C6
Francfort	0	8000	0	55000	0	0
Hambourg	50000	0	0	0	0	0
Leipzig	0	62000	0	0	30000	0

TABLE 14 – Quantités optimales entre dépôts et clients (nouvelles options de dépôts avec préférence des clients)

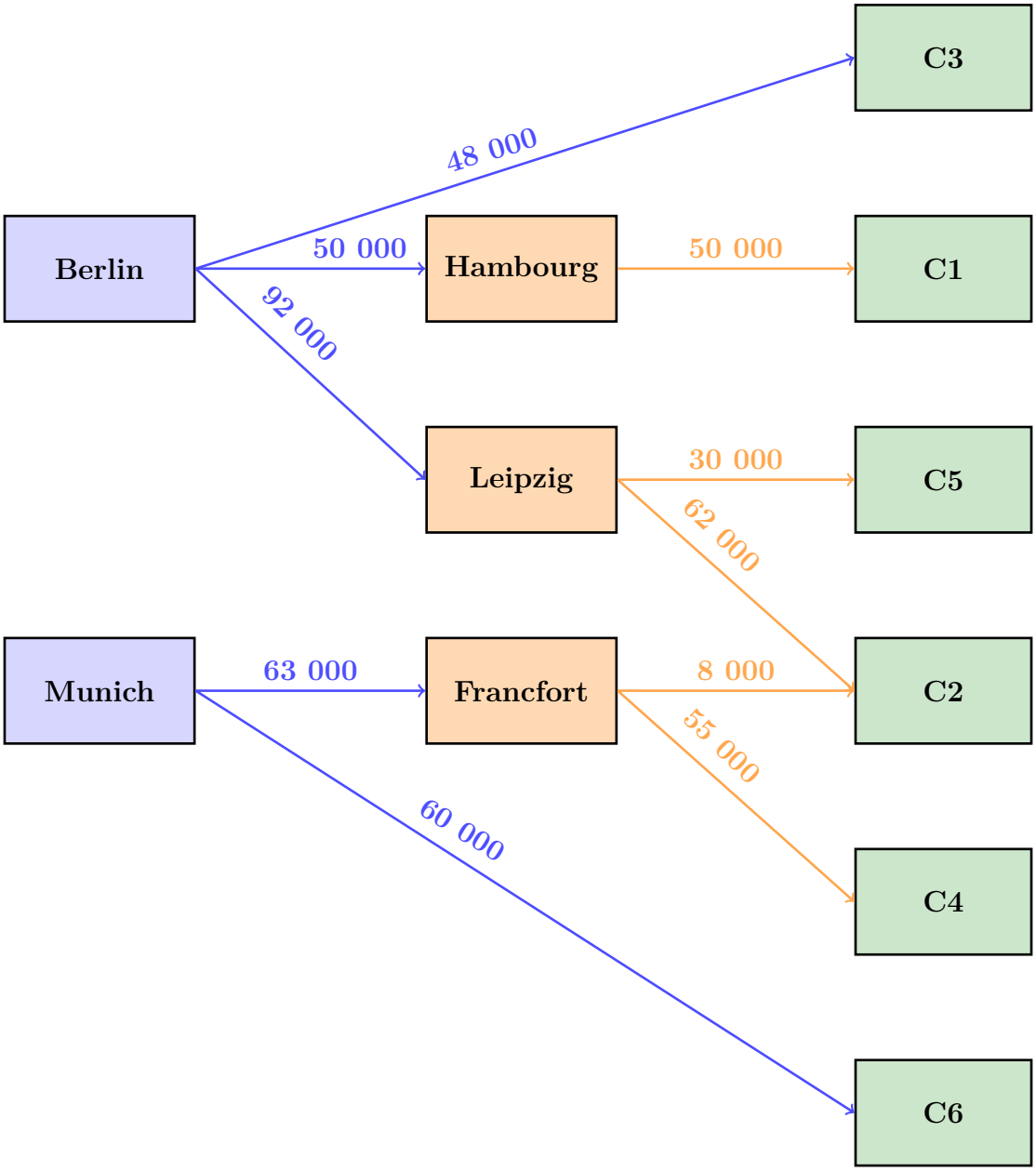

FIGURE 7 – Schéma de distribution optimal avec nouvelles options de dépôts incluant les préférences des clients (fermeture Düsseldorf et agrandissement Leipzig)

La stratégie de schéma est basée sur une plus grande sollicitation du dépôt de Leipzig (agrandi) aux dépens de celui de Düsseldorf (fermé). La possibilité d'ouverture du dépôt de Stuttgart a été jugée non rentable, et cela s'explique par les contraintes de préférences des clients, qui restreignent le solveur dans l'exploration de cette piste.

5 Synthèse des résultats et perspectives

L'analyse menée a permis de quantifier et d'optimiser les flux logistiques de l'entreprise tout en proposant des ajustements stratégiques en matière de gestion des dépôts. Les principales conclusions sont les suivantes :

- **Optimisation du réseau initial** : En l'absence de modifications structurelles, la meilleure allocation des ressources a permis de réduire le coût total de transport à **23 420 000 €** tout en garantissant la satisfaction des besoins clients.
- **Prise en compte des préférences clients** : En intégrant les contraintes de préférences sur certains fournisseurs, une légère augmentation du coût total (**+3,3%**, soit **24 192 000 €**) a été observée, démontrant l'impact des choix individuels sur la rentabilité globale du réseau.
- **Réorganisation des dépôts** : L'étude de nouvelles options a révélé qu'en ouvrant un dépôt supplémentaire à Stuttgart, en fermant celui de Düsseldorf et en agrandissant celui de Leipzig, une économie de **668 000 €** pouvait être réalisée, ramenant le coût total à **22 752 000 €**.
- **Optimisation combinée avec préférences clients** : En tenant compte simultanément des préférences clients et des nouvelles options de dépôts, une économie plus modérée (**134 000 €**) a été obtenue, confirmant qu'une flexibilité stratégique est nécessaire pour concilier rentabilité et satisfaction des clients.

Ces résultats montrent que la gestion des flux logistiques ne se limite pas à la réduction des coûts, mais implique aussi une adaptation efficace des infrastructures. En ajustant le réseau de dépôts, l'entreprise a pu équilibrer ses dépenses et améliorer l'efficacité des livraisons. L'intégration des préférences clients a également permis de mieux aligner l'offre logistique avec la demande, tout en limitant l'impact financier.

Cette étude démontre qu'un réseau logistique optimisé repose sur des décisions stratégiques bien calibrées, adaptées aux contraintes du marché et aux attentes des clients.

6 Annexes

6.1 Partie avec les dépôts initiaux

```
#### Fichier probleme_transport_partie1.dat ####
set USINES := Berlin Munich;
set DEPOTS := Hambourg Francfort Dusseldorf Leipzig;
set CLIENTS := C1 C2 C3 C4 C5 C6;

param prix_route :
               Hambourg  Francfort  Dusseldorf  Leipzig  C1  C2  C3  C4  C5  C6
     :=
    Berlin      47       62        56          38        0   83   82    0   85    0
    Munich       0       48        60          57       84    0    0   78    0   93
    Hambourg     0        0         0           0       22   35   40   30    0   31
    Francfort    0        0         0           0       26   31   37   24   29    0
    Dusseldorf   0        0         0           0       38    0   26    0   33   36
    Leipzig      0        0         0           0        0   34    0   28   30   40
    ;

param capacite_accueil :=
    Berlin  210000
    Munich  190000
    Hambourg  105000
    Francfort  90000
    Dusseldorf  71000
    Leipzig  67000
    ;

param besoin_client :=
    C1  50000
    C2  70000
    C3  48000
    C4  55000
    C5  30000
    C6  60000
    ;

param preference_client :
            Berlin  Munich  Hambourg  Francfort  Dusseldorf  Leipzig :=
    C1        0       0        1          0          0          0
    C2        0       0        0          1          0          1
    C3        1       0        0          1          0          0
    C4        0       1        0          1          0          0
    C5        1       1        1          1          1          1
    C6        0       1        0          0          1          0
    ;
```

```
#### Fichier probleme_transport_partie1.mod ####
# Ensembles
set USINES;
set DEPOTS;
set CLIENTS;

# Paramètres
param prix_route{USINES union DEPOTS , CLIENTS union DEPOTS} >= 0;
param capacite_accueil{USINES union DEPOTS} >= 0;
param besoin_client{CLIENTS} >= 0;
param preference_client{CLIENTS, USINES union DEPOTS} binary;

# Variables
var quantite_UC{u in USINES, c in CLIENTS} >= 0;
var quantite_UD{u in USINES, d in DEPOTS} >= 0;
var quantite_DC{d in DEPOTS, c in CLIENTS} >= 0;

# Fonction objectif
minimize cout_total:
sum{u in USINES, c in CLIENTS} prix_route[u,c] * quantite_UC[u,c]
+ sum{u in USINES, d in DEPOTS} prix_route[u,d] * quantite_UD[u,d]
+ sum{d in DEPOTS, c in CLIENTS} prix_route[d,c] * quantite_DC[d,c];

# Contrainte des capacités des usines et des dépôts
subject to Capacite_Usines{u in USINES}:
sum{c in CLIENTS} quantite_UC[u,c] + sum{d in DEPOTS} quantite_UD[u,d] <=
    capacite_accueil[u];
subject to Capacite_Depots{d in DEPOTS}:
sum{u in USINES} quantite_UD[u,d] <= capacite_accueil[d];

# Contrainte des besoins des clients
subject to Besoin_Clients{c in CLIENTS}:
sum{u in USINES} quantite_UC[u,c] + sum{d in DEPOTS} quantite_DC[d,c] =
    besoin_client[c];

# Contrainte des routes indisponibles
subject to Route_Indisponible_Usine_Client{u in USINES, c in CLIENTS:
    prix_route[u,c] == 0}:
quantite_UC[u,c] = 0;
subject to Route_Indisponible_Usine_Depot{u in USINES, d in DEPOTS : prix_route
    [u,d] == 0}:
quantite_UD[u,d] = 0;
subject to Route_Indisponible_Depot_Client{d in DEPOTS, c in CLIENTS:
    prix_route[d,c] == 0}:
quantite_DC[d,c] = 0;

# Contrainte de liquidation des stocks des dépôts
subject to Liquidation_Stock{d in DEPOTS}:
sum{u in USINES} quantite_UD[u,d] = sum{c in CLIENTS} quantite_DC[d,c];

# Veuillez retirer les commentaires pour inclure les préférences des clients
#subject to Respect_Preferences{c in CLIENTS, s in USINES union DEPOTS}:
#if preference_client[c,s] == 0 then (sum{u in USINES: u == s} quantite_UC[u,c]
    # + sum{d in DEPOTS: d == s} quantite_DC[d,c]) = 0;
```

```
#### Fichier probleme_transport_partie1.run ####
reset;
model ../nomrepertoire/probleme_transport_partie1.mod;
data ../nomrepertoire/probleme_transport_partie1.dat;
option solver cplex;
solve;

display quantite_UC; display quantite_UD; display quantite_DC;
display cout_total;
```

6.1.1 Sorties sans prise en compte des préférences des clients

```
Presolve eliminates 14 constraints and 13 variables.
Adjusted problem:
31 variables, all linear
15 constraints, all linear; 68 nonzeros
1 linear objective; 31 nonzeros.

solve
optimal solution; objective 23420000
16 dual simplex iterations (0 in phase I)
display
quantite_UC :=
Berlin C1       0
Berlin C2       0
Berlin C3   38000
Berlin C4       0
Berlin C5       0
Berlin C6       0
Munich C1       0
Munich C2       0
Munich C3       0
Munich C4    3000
Munich C5       0
Munich C6       0
;

display
quantite_UD :=
Berlin Dusseldorf        0
Berlin Francfort         0
Berlin Hambourg     105000
Berlin Leipzig       67000
Munich Dusseldorf    10000
Munich Francfort     90000
Munich Hambourg          0
Munich Leipzig           0
;

display
quantite_DC [*,*] (tr)
:  Dusseldorf Francfort Hambourg Leipzig    :=
C1         0       5000    45000        0
C2         0      33000        0    37000
C3     10000          0        0        0
C4         0      52000        0        0
C5         0          0        0    30000
C6         0          0    60000        0
;

display
cout_total = 23420000
```

6.1.2 Sorties incluant les préférences des clients

```
Presolve eliminates 52 constraints and 26 variables.
Adjusted problem:
18 variables, all linear
13 constraints, all linear; 42 nonzeros
1 linear objective; 18 nonzeros.

solve
optimal, objective 24192000
12 simplex iterations
display
quantite_UC :=
Berlin C1       0
Berlin C2       0
Berlin C3   48000
Berlin C4       0
Berlin C5       0
Berlin C6       0
Munich C1       0
Munich C2       0
Munich C3       0
Munich C4       0
Munich C5       0
Munich C6   15000
;

display
quantite_UD :=
Berlin Dusseldorf   45000
Berlin Francfort        0
Berlin Hambourg     50000
Berlin Leipzig      67000
Munich Dusseldorf       0
Munich Francfort    88000
Munich Hambourg         0
Munich Leipzig          0
;

display
quantite_DC [*,*] (tr)
:  Dusseldorf Francfort Hambourg Leipzig    :=
C1         0          0    50000        0
C2         0      33000        0    37000
C3         0          0        0        0
C4         0      55000        0        0
C5         0          0        0    30000
C6     45000          0        0        0
;

display
cout_total = 24192000
```

6.2 Partie avec les nouvelles options de dépôts

```
#### Fichier probleme_transport_partie2.dat ####
set CLIENTS := C1 C2 C3 C4 C5 C6;
set DEPOTS := Hambourg Francfort Dusseldorf Leipzig Stuttgart Dortmund;
set USINES := Berlin Munich;

param prix_route :
              Hambourg  Francfort  Dusseldorf  Leipzig  Stuttgart  Dortmund
    C1  C2  C3  C4  C5  C6 :=
    Berlin      47    62    56    38    68    53     0   83   82    0   85    0
    Munich       0    48    60    57    41    61    84    0    0   78    0   93
    Hambourg     0     0     0     0     0     0    22   35   40   30    0   31
    Francfort    0     0     0     0     0     0    26   31   37   24   29    0
    Dusseldorf   0     0     0     0     0     0    38    0   26    0   33   36
    Leipzig      0     0     0     0     0     0     0   34    0   28   30   40
    Stuttgart    0     0     0     0     0     0    30   26   29   38   23   34
    Dortmund     0     0     0     0     0     0    28   27   24   32   31   25
    ;


param capacite_accueil :=
    Berlin    210000
    Munich  190000
    Hambourg 105000
    Francfort   90000
    Dusseldorf  71000
    Leipzig 67000
    Stuttgart 60000
    Dortmund 38000
    ;

param besoin_client :=
    C1  50000
    C2  70000
    C3  48000
    C4  55000
    C5  30000
    C6  60000
    ;

param cout_ouverture := Stuttgart 285000 Dortmund 175000;
param cout_extension := Leipzig 75000;
param economie_fermeture := Francfort 94000 Dusseldorf 79000;

param preference_client :
           Berlin  Munich  Hambourg  Francfort  Dusseldorf  Leipzig  Stuttgart
     Dortmund :=
    C1        0        0        1          0          0         0        0      0
    C2        0        0        0          1          0         1        0      0
    C3        1        0        0          1          0         0        0      0
    C4        0        1        0          1          0         0        0      0
    C5        1        1        1          1          1         1        1      1
    C6        0        1        0          0          1         0        0      0;
```

```
#### Fichier probleme_transport_partie2.mod ####
# Ensembles
set CLIENTS;
set DEPOTS;
set USINES;

# Paramètres de la partie 1
param prix_route{USINES union DEPOTS , CLIENTS union DEPOTS} >= 0;
param capacite_accueil{USINES union DEPOTS} >= 0;
param besoin_client{CLIENTS} >= 0;
param preference_client{CLIENTS, USINES union DEPOTS} >= 0;

# Nouveaux paramètres entrants
param cout_ouverture{DEPOTS} >= 0;
param economie_fermeture{DEPOTS} >= 0;
param cout_extension{DEPOTS} >= 0;

# Variables
var quantite_UC{u in USINES, c in CLIENTS} >= 0;
var quantite_UD{u in USINES , d in DEPOTS} >= 0;
var quantite_DC{d in DEPOTS, c in CLIENTS} >= 0;

# Variables de décision binaires
var decision_Stutt binary := 0; # devient 1 si ouverture d'un dépôt a Stuttgart
var decision_Dort binary := 0; # devient 1 si ouverture d'un dépôt a Dortmund
var decision_Leip binary := 0; # devient 1 si extension du dépôt de Leipzig
var decision_Duss binary := 1; # devient 0 si fermeture d'un dépôt a Düsseldorf
var decision_Franc binary := 1; # devient 0 si fermeture d'un dépôt a Francfort

# Nouvelle fonction objectif : Minimiser les coûts en variant les options des
    dépôts
minimize cout_total_avec_changement_depots:
decision_Stutt * cout_ouverture["Stuttgart"]
+ decision_Dort * cout_ouverture["Dortmund"]
+ decision_Leip * cout_extension["Leipzig"]
- (1 - decision_Duss) * economie_fermeture["Dusseldorf"]
- (1 - decision_Franc) * economie_fermeture["Francfort"]
+ sum{u in USINES, c in CLIENTS} prix_route[u,c] * quantite_UC[u,c]
+ sum{u in USINES, d in DEPOTS} prix_route[u,d] * quantite_UD[u,d]
+ sum{d in DEPOTS, c in CLIENTS} prix_route[d,c] * quantite_DC[d,c];

# Contrainte des besoins des clients
subject to Besoin_Clients{c in CLIENTS}:
sum{u in USINES} quantite_UC[u,c]
+ sum{d in DEPOTS} quantite_DC[d,c] = besoin_client[c];

# Contraintes des capacités des usines et des dépôts
subject to Capacite_Usines{u in USINES}:
sum{c in CLIENTS} quantite_UC[u,c]
+ sum{d in DEPOTS} quantite_UD[u,d] <= capacite_accueil[u];

subject to Capacite_Depots{d in (DEPOTS diff {"Leipzig"})}:
sum{c in CLIENTS} quantite_DC[d,c] <= capacite_accueil[d];
```

```
# Contrainte pour le dépôt de Leipzig
subject to Debit_Leipzig:
sum{c in CLIENTS} quantite_DC["Leipzig",c]
<= capacite_accueil["Leipzig"] + decision_Leip * 25000;

# Contrainte pour les routes indisponibles entre Usines et Clients
subject to Route_Usine_Client_Zero{u in USINES, c in CLIENTS: prix_route[u,c]
    == 0}:
quantite_UC[u,c] = 0;

# Contrainte pour les routes indisponibles entre Usines et Dépôts
subject to Route_Usine_Depot_Zero{u in USINES, d in DEPOTS : prix_route[u,d] ==
     0}:
quantite_UD[u,d] = 0;

# Contrainte pour les routes indisponibles entre Dépôts et Clients
subject to Route_Depot_Client_Zero{d in DEPOTS, c in CLIENTS: prix_route[d,c]
    == 0}:
quantite_DC[d,c] = 0;

# Contrainte de liquidation des stocks des dépôts
subject to Liquidation_Stock{d in DEPOTS}:
sum{u in USINES} quantite_UD[u,d] = sum{c in CLIENTS} quantite_DC[d,c];

# Contrainte pour garantir un maximum de 4 dépôts ouverts
subject to Nombre_Max_Depots:
decision_Stutt + decision_Dort + decision_Franc + decision_Duss <= 2;

# Contraintes des quantités livrées selon l'ouverture ou la fermeture des
    dépôts
subject to Quantites_Francfort:
sum{c in CLIENTS} quantite_DC["Francfort", c] <= decision_Franc *
    capacite_accueil["Francfort"];

subject to Quantites_Dusseldorf:
sum{c in CLIENTS} quantite_DC["Dusseldorf", c] <= decision_Duss *
    capacite_accueil["Dusseldorf"];

subject to Quantites_Stuttgart:
sum{c in CLIENTS} quantite_DC["Stuttgart", c] <= decision_Stutt *
    capacite_accueil["Stuttgart"];

subject to Quantites_Dortmund:
sum{c in CLIENTS} quantite_DC["Dortmund", c] <= decision_Dort *
    capacite_accueil["Dortmund"];

# Contrainte des préférences des clients
# Veuillez retirer les commentaires ci-dessous pour traiter ces hypothèses
#subject to Respect_Preferences{c in CLIENTS, s in USINES union DEPOTS}:
#if preference_client[c,s] == 0 then (sum{u in USINES: u == s} quantite_UC[u,c]
    # + sum{d in DEPOTS: d == s} quantite_DC[d,c]) = 0;
```

```
#### Fichier probleme_transport_partie2.run ####
reset;
model ../nomrepertoire/probleme_transport_partie2.mod;
data ../nomrepertoire/probleme_transport_partie2.dat;
option solver cplex;
solve;

display decision_Stutt, decision_Dort, decision_Leip, decision_Duss,
    decision_Franc;
display quantite_UC; display quantite_UD; display quantite_DC;
display cout_total_avec_changement_depots;
```

6.2.1 Sorties sans prise en compte des préférences des clients

```
Presolve eliminates 13 constraints and 13 variables.
Adjusted problem:
52 variables:
  5 binary variables
  47 linear variables
25 constraints, all linear; 154 nonzeros
1 linear objective; 52 nonzeros.

solve
optimal integer solution; objective 22752000
40 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 3.72529e-09, relmipgap = 1.63735e-16
display
decision_Stutt = 1
decision_Dort = 0
decision_Leip = 1
decision_Duss = 0
decision_Franc = 1

display
quantite_UC :=
Berlin C1   0
Berlin C2   0
Berlin C3   0
Berlin C4   0
Berlin C5   0
Berlin C6   0
Munich C1   0
Munich C2   0
Munich C3   0
Munich C4   0
Munich C5   0
Munich C6   0
;

display
quantite_UD :=
Berlin Dortmund          0
Berlin Dusseldorf        0
Berlin Francfort         0
Berlin Hambourg     105000
Berlin Leipzig       92000
Berlin Stuttgart         0
Munich Dortmund          0
Munich Dusseldorf        0
Munich Francfort     56000
Munich Hambourg          0
Munich Leipzig           0
Munich Stuttgart     60000
;
```

```
display
quantite_DC [*,*]
:              C1      C2      C3      C4      C5      C6      :=
Dortmund        0       0       0       0       0       0
Dusseldorf      0       0       0       0       0       0
Francfort    5000       0       0   51000       0       0
Hambourg    45000       0       0       0       0   60000
Leipzig         0   58000       0    4000   30000       0
Stuttgart       0   12000   48000       0       0       0
;

display
cout_total_avec_changement_depots = 22752000
```

6.2.2 Sorties incluant les préférences des clients

```
Presolve eliminates 66 constraints and 36 variables.
Adjusted problem:
29 variables:
  5 binary variables
  24 linear variables
20 constraints, all linear; 73 nonzeros
1 linear objective; 29 nonzeros.

solve
optimal integer solution; objective 24058000
10 MIP simplex iterations
0 branch-and-bound nodes
display
decision_Stutt = 0
decision_Dort = 0
decision_Leip = 1
decision_Duss = 0
decision_Franc = 1

display
quantite_UC :=
Berlin C1       0
Berlin C2       0
Berlin C3   48000
Berlin C4       0
Berlin C5       0
Berlin C6       0
Munich C1       0
Munich C2       0
Munich C3       0
Munich C4       0
Munich C5       0
Munich C6   60000
```

```
;

display
quantite_UD :=
Berlin Dortmund         0
Berlin Dusseldorf       0
Berlin Francfort        0
Berlin Hambourg     50000
Berlin Leipzig      92000
Berlin Stuttgart        0
Munich Dortmund         0
Munich Dusseldorf       0
Munich Francfort    63000
Munich Hambourg         0
Munich Leipzig          0
Munich Stuttgart        0
;

display
quantite_DC [*,*]
:              C1      C2    C3    C4      C5    C6    :=
Dortmund          0       0   0       0       0   0
Dusseldorf        0       0   0       0       0   0
Francfort         0    8000   0   55000       0   0
Hambourg      50000       0   0       0       0   0
Leipzig           0   62000   0       0   30000   0
Stuttgart         0       0   0       0       0   0
;

display
cout_total_avec_changement_depots = 24058000
```