Routing on Dynamic Networks: GRASP versus Genetic

Benoit Bernay
Université Blaise Pascal
LIMOS CNRS Laboratory,
LABEX IMOB3
Clermont-Ferrand 63000, France
Email: bernay@isima.fr

Samuel Deleplanque
Université Blaise Pascal
LIMOS CNRS Laboratory
LABEX IMOB3
Clermont-Ferrand 63000, France
Email: deleplanque@isima.fr

Alain Quilliot
Université Blaise Pascal
LIMOS CNRS Lab.
LABEX IMOB3
Clermont-Ferrand, France
Email: alain.quilliot@isima.fr

Abstract—We address here a large scale routing and scheduling transportation problem, through introduction of a flow model designed on a dynamic network. We deal with this model while using a master/slave decomposition scheme, and testing the behavior on this scheme of both a GRASP algorithm and a Genetic algorithm.

I. INTRODUCTION

We already introduced (see [9]), in the context of a partnership with an industrial player, a flow/multi-commodity flow model FMS which aimed at optimizing the management of an urban shuttle fleet. This model involved a dynamic network (see [2, 10]), that is a network with time indexed vertices, which made easy expressing temporal constraints. At this time we designed a GRASP algorithmic scheme, which allowed us handling a kind of large scale pre-emptive Pick Up and Delivery problem (see [10]), while using an ad hoc aggregation mechanism and performing random negative circuit cancelling.

We consider here the same model FMS, close to CFA (Capacitated Flow Assignment) models (see [1]) used in telecommunications, but we deal with it in a simpler way, while using an auxiliary cost vector as the master variable of a master/slave decomposition scheme. This scheme induces the design of resolution heuristics which mainly rely on simple shortest path procedures instead of complex negative circuit cancelling procedure, and whose generic features makes implementation easier. While next section II is devoted to a rough description of the FMS model, our main contribution is about the description in Section III of this master/slave decomposition scheme, from which we derive (sections IV and V) both a GRASP (Greedy Random Adaptive Search Procedure, see [5, 6]) algorithm, and a genetic algorithm (see [6, 7, 8]). We detail the way those algorithms are implemented, and test (Section VI) their respective behaviors.

II. THE FMS MODEL

A. Main Notations and Definitions

A network \( G \) with vertex set \( X \) and arc set \( E \), is denoted by \( G = (X, E) \). A flow vector is an arc indexed vector \( f \) with rational or integral values such that, for every vertex \( x \), we have \( \Sigma e \in \text{enter into } x . f_e = \Sigma e \in \text{comes out } x . f_e \) (Kirchhoff Law). The arc support of \( f \) is the arc subset \( \text{Arc-Supp}(f) \subseteq E \), which contains all arcs \( e \in E \) such that \( f_e \neq 0 \). A multi-commodity flow vector \( f \) is a flow vector collection \( f = \{f(k), k \in K\} \). \( \text{Sum}(f) \) is the Aggregated Flow \( \text{Sum}(f) = \Sigma_{k \in K} f(k) \).

B. The Shuttle Problem (see [16])

We consider a Urban Area network \( H = (Z, U) \) nodes of \( H \) mean either production sites \( y_1, \ldots, y_m \) \( (m = 7 \) in the original application), or residential areas, and arcs mean elementary connections. A demand \( D_k, k \in K \), is a 4-uple \((o_k, d_k, L_k, t_k)\); \( o_k \): origin/destination nodes, \( L_k \): load, \( t_k \): deadline: \( L_k \) users have to be transported from \( o_k \) and to \( d_k \) (at least one of both nodes being an industrial node) while starting (arriving) after (before) time \( s_k \) \( (a_k) \). Quality of Service (QoS) requires this trip not to last more than \( T_k \) time units. Users alternatively walk and use a shuttle system; so, every arc \( e \) of \( H \) is endowed with a walking length \( l_w(e) \) and with a vehicle length \( l_v(e) \). Vehicles start from and end into a Depot node. Our goal is to route the shuttles while meeting the demands and minimizing both the number of vehicles (Fixed Investment Cost) and their running times (Running Cost). Route preemption is allowed: several vehicles may be involved in meeting a given demand.

C. The Dynamic Network H-Dyn.

We derive it from \( H \) by associating (see 2, 8, 10), with any node \( x \) of \( Z \), \( (NP+1) \) copies of \( x \), indexed from 0 to \( NP \), which represent the states of \( x \) at the instants 0, \( \delta \), ..., \( NP \delta \);
\( \delta \) is an elementary time unit, chosen between 3 \( \text{mn} \) and 6 \( \text{mn} \) in our application; \( \text{NP} \) is a parameter which fixes the planning period (between 2 and 3 \( \text{h} \)). We add 2 fictitious vertices \( \text{DP}, \text{DP}^* \) and set \( X = \{ x, x \in Z, r \in 0, \ldots, \text{NP} \} \cup \{ \text{DP}, \text{DP}^* \} \). As for the arc set \( E \), we round modulo \( \delta \) the vehicle and walking lengths of any arc \( u \) in \( U \) by setting:

\[
I^\text{\( \text{DP} \)}(u) = \left\lceil I_u(\delta^{\text{\( \text{DP} \)}}) \right\rceil, \quad I^\text{\( \text{DP}^* \)}(u) = \left\lceil I_u(\delta^{\text{\( \text{DP}^* \)}}) \right\rceil;
\]

then we define the labeled arc family \( E \) as containing:

- wait arcs \( (x, x_{\text{wait}}), x \in Z, r \in 0, \ldots, \text{NP-1} : \) such an arc is considered twice, with walk and vehicle labels;
- arcs \( \{ \text{DP}, \text{Depot}_r \}, \{ \text{Depot}_r, \text{DP}^* \}, \ldots, \text{NP} \}, \) with vehicle labels.
- arcs \( (x, z_{\text{wait}}), u = (x, z) \in U, r \) such that \( 0 \leq r \leq \text{NP} - I^\text{\( \text{DP} \)}(u), \) with vehicle label;
- walk arcs \( (x, z_{\text{walk}}), u = (x, z) \in U, \) with walk label;
- a backward arc \( \{ \text{DP}^*, \text{DP} \}. \)

We denote by \( A \) the subset of \( E \) defined by the vehicle arcs. We provide, in a natural way, any arc \( e \) with an Economical Cost \( c_e \) and a QoS Cost \( p_e \).

**Fig. 1:** Urban Transit Network \( H = (Z, U) \)

**Fig. 2:** Dynamic Network \( H\text{-Dyn} = (X, E) \).

**Remark 1:** Size of \( H\text{-Dyn} \). We consider that \( (x, y) \) is an arc of \( H \) if a vehicle may move from \( x \) to \( y \) during time unit \( \delta \). If \( H \) contains 200 nodes, \( H\text{-Dyn} \) may contain up to \( 10^5 \) arcs.

**D. The Flow/Multi-commodity Shuttle Model (FMS)**

We want to route both vehicles and users. Aggregating vehicle routes yields, on the dynamic network \( H\text{-Dyn} = (X, E) \), some integral flow vector \( F \), and that user’s routes may be represented as a rational multi-commodity flow \( f = \{ f(k), k \in K \} \geq 0 \). Measuring \( f \) in such a way the capacity of any vehicle becomes equal to 1 yields the following **FMS: Flow/Multi-Commodity Flow Shuttle model**:

**FMS: Flow/Multi-Commodity Flow Shuttle Model.**

**Input:** The Urban Area network \( H = (Z, U) \), and the discrete time space \( \{ 0, \ldots, \text{NP} \} \).

**Output:** Compute, on the dynamic network \( H\text{-Dyn} \) an integral flow vector \( F \), and a rational multi-commodity flow \( f = \{ f(k), k \in K \} \geq 0 \), such that:

- \( F \) is null on the walk arcs;
- For any \( k \in K, f(k) \) routes \( L_k \), users from \( o_k \) to \( d_k \) between time \( s_k \) and time \( a_k \);
- \( g = \text{Sum}(f) = \sum_{k \in K} f(k) \leq F, \) for any arc of \( H\text{-Dyn} \) with vehicle label;
- Cost \( (F) + \text{QoS}(f) = c.F + p.\text{Sum}(f) = \sum_{e \in A} c_e.F_e + \sum_{e \in E} p_e.\text{Sum}(f)_e \) is minimal.

Conversely, if \( F \) and \( f \) satisfy FMS constraints and if Route Preemption is allowed, then they yield a feasible solution of the Shuttle Problem. Our model casts temporal constraints into the construction of the network \( H\text{-Dyn} \). We denote by **FMS**, the time-polynomial min cost integral flow problem which derives from **FMS** by fixing \( g = \text{Sum}(f) \).

**Remark 2:** FMS Model Size. If the number of demands is 250, then the size of \( f \) may be up to 25 \( 10^6 \); the resulting FMS model is a large scale NP-Hard MIP problem.

**III. FUNDAMENTAL TOOLS**

Before describing algorithm, we need to specify which objects and procedures they will involve.

**A.A Master/Slave Encoding of a FMS Solution**

The quality of a FMS solution relies on its ability to make users share vehicles. While the size \( H\text{-Dyn} \) may eventually be very large, the number of arcs which are going to support non null \( F \) and \( g = \text{Sum}(f) \) is comparatively small. So, a key object in our model should be the arc support set \( A = \text{Arc-Supp}(F) = \{ e \in E \text{ such that } F_e \neq 0 \} \) of \( F \). The following theoretical result, whose proof can be obtained through standard mathematical programming techniques, will help us in dealing with this arc support set.

**Dualization Theorem:** Let \( (F, f) \) an optimal solution of the FMS model. Then there exists a price vector \( \mu \geq 0 \), with indexation on the arc set of \( H\text{-Dyn} \), such that:

- \( \mu_e = 0 \) for any arc \( e \) which is a walk arc or a wait arc and which is not in \( A \);
- \( \mu_e \geq c_e \) for any arc \( e \in A; \mu_e = + \infty \) for any vehicle arc \( e \) which is not in \( A \);
- Every flow \( f(k) \) is an optimal solution of the min cost flow problem defined by: (E1)
  - for every \( k \in K, f(k) \) routes \( L_k \), from \( o_k \) to \( d_k \) between time \( s_k \) and time \( a_k \);
  - \( \sum_e (\mu_e + p_e).f(k) \) is the smallest possible.
So, the knowledge of both arc support set $A$ and cost vector $\mu$ allow us to derive, through shortest path procedures, the aggregated flow $g = \text{Sum}(f)$. Flow vector $F$ is computed as a solution of $\text{FMS}_g$. We impose every vector $f(k), k \in K$, to be routed along a single path. So, a well-fitted representation of a $\text{FMS}$ solution is given by:

- the set $\text{Arc-Supp}(F) = \{ e \in E \text{ such that } F_e \neq 0 \}$;
- the related cost vector $\mu = \mu_e, e \in A$.

Those objects define the Master part of such a solution, whose Slave part is defined by $F$ and the collection $\Gamma(k)$ of the paths followed by the flow vectors $f(k), k \in K$.

B. Dealing with the $\text{FMS}_g$ Problem

We deal with $\text{FMS}_g$ through column generation, while using an arc/path formulation of $\text{FMS}_g$:

$$\text{FMS}_g\{\Lambda \text{ denotes the set of paths from } DP \text{ to } DP^*; \text{ Compute a vector } G = (G_\gamma, \gamma \in \Lambda) \geq 0, \text{ with rational values, such that:}$$

- for any arc $e$ of $H$-Dyn with vehicle label, $\Sigma \gamma$ such that $e \in \gamma G_\gamma \geq \lceil g_e \rceil$;
- $\Sigma \gamma \text{Cost}(\gamma G_\gamma)$ is minimal

If $\Lambda_0$ is some active subset of $\Lambda$, and if $\lambda = (\lambda_e, e \in \text{arc subset of } H$-Dyn with vehicle label) $\geq 0$ is a dual solution of the restriction of $\text{FMS}_g$ to $\Lambda_0$, then the related Pricing (search for the new entering column) sub-problem is as a largest path problem, handled by Bellman algorithm. So, when dealing with the $\text{FMS}_g$ problem, we do in such a way that we are always provided with some current active path subset $\Lambda_0$ of $\Lambda$, which evolves in an incremental way.

C. Deriving the paths $\Gamma(k)$ from $A$ and $\mu$.

Dualization Theorem tells us that support set $A$ and cost vector $\mu$ should identify the arcs along which users are going to share a same vehicle. If $A$ and $\mu$ were conveniently chosen, paths $\Gamma(k), k \in K$, should be shortest paths for cost vector $(p + \mu)$. So, all throughout the execution of our processes, we derive paths $\Gamma(k), k \in K$, as shortest paths for the following cost vector $C^A,\mu$:

- If $e$ is a walk other wait arc which is not in $A$, then $C^A,\mu = p_e$;
- If $e$ is in $A$, then $C^A,\mu = \mu_e + p_e$;
- Else $C^A,\mu = p_e + \mu^*, \text{ where } \mu^* = \text{Max}_{e \in A} \mu_e/c_e$. \hspace{1cm} (E2)

As a matter of fact, for a given vehicle arc $e \notin A$, we apply (E2), which means that we want to keep paths $\Gamma(k), k \in K$ from using vehicle arcs which are not in $A$, only when no path $\Gamma(k), k \in K$ involves $e$. Else, we use $\mu^*$ defined by:

$$\mu^* = \text{Mean Value}_{e \in A} \mu_e.$$

D. A Randomized Initialization

This initialization procedure $\text{FMS-Init}$ works through successive insertions of demands $D_k, k \in K$, into a current aggregated flow vector $g$:

**FMS-INIT Procedure:**
- $g$: current aggregated flow vector; $K_0$: set of inserted demands;
- $F$ and $\lambda$: primal and dual solutions of $\text{FMS}_g$;
- $\Lambda_0$: set of active vehicle paths;
- While $K - K_0$ is not empty do
  - Randomly Pick up $k \in K - K_0$ and Insert it into $K_0$;
  - route demand $k$ according to some path $\Gamma(k)$ in $H$-Dyn, in such a way that:
    (I1) $\Gamma(k)$ connects $o_k$ to $d_k$, while satisfying related temporal constraints;
    (I2) the induced increase in the cost $\lambda, [g] + p_g$ is the smallest possible;
  - Update $F$, $\lambda$, and $\Lambda_0$.

Set $A = \text{Arc Support of } F$; For every arc $e$ in $A$, set:

$$\mu_e = \lambda_e, [g_e]. \hspace{1cm} (I2)$$

The above Insert instruction (I1) is handled by a shortest path Bellman-like Algorithm.

E. Local Transformation and Mutation Operators

The $\text{FMS-INIT}$ previous process gives rise in a generic way to a local transformation operator TRANS, which acts on a current solution $A$, $\mu$, $F$, $\Gamma = \{\Gamma(k), k \in K\}$ as follows:

**Local Operator TRANS($K_0$: $K_0$ subset of $K$)**
- Randomly select $K_0 \subseteq K$ and withdraws paths-flows $(f(k), k \in K_0)$; Update flow vector $F$;
- Reinsert demands $D_k, k \in K_0$, according to the III.D, while starting from current partial solution $(F, g)$;
- Consequently update $A$ and $\mu$.

Operator TRANS will be used here in both GRASP scheme, according to a Descent strategy and in a genetic meta-heuristic scheme, as a mutation operator.

F. Crossover Operator

Given two feasible $\text{FMS}$ solutions $A_1$, $\mu_1$, $F_1$, $\Gamma_1 = \{\Gamma_1(k), k \in K\}$ and $A_2$, $\mu_2$, $F_2$, $\Gamma_2 = \{\Gamma_2(k), k \in K\}$. $\text{SON-CREATE}$ derives children $(A, \mu)$ and $(A', \mu')$ as follows:
Crossover Operator SON-CREATE:
For every arc $e$ in $(A_1 \cap A_2)$, insert $e$ into both $A$ and $A'$ and randomly assign related value $\mu_e$ or $\mu'_e$ with one of both values $(\mu_{1,e}, \mu_{2,e})/2$ and $(\mu_{1,e}, \mu_{2,e})/2$.
For every arc $e$ in $(A_1 - A_2) \cup (A_2 - A_1)$, randomly insert $e$ into either $A$ or $A'$ and randomly assign related value $\mu_e$ or $\mu'_e$ with one of both values $(\mu_{1,e}, \mu_{2,e})/2$ or $(\mu_{1,e}, \mu_{2,e})/2$.
Compute path collections $\Gamma = \{ \Gamma(k), k \in K \}$ and $\Gamma' = \{ \Gamma'(k), k \in K \}$ as in III.C; Compute $F$ and $F'$ flow vectors as in III.B, together with dual vectors $\lambda$ and $\lambda'$; Update cost vectors $\mu$ and $\mu'$ according to (I2).

IV. A GRASP ALGORITHM FMS-GRASP FOR FMS

A GRASP: Greedy Random Adaptive Search Procedure (see [5, 6]) algorithmic scheme works by performing first a greedy randomized initialization process, and next a descent loop. It may be run according to several replications, either in a sequential or in a parallel mode. Here, we get:

\[
\text{FMS-GRASP}(R; \text{Replication Number}, Q; \text{Subset Size}, Loop; \text{Loop Length Bound});
\]
For $i = 1..R$ do
\[\text{FMS-INIT; Possible; While Possible do (I3: Descent loop) Modify } A, \mu, F, \Gamma = \{ \Gamma(k), k \in K \}; \text{ If Failure(Modify) then Not Possible; The result of FMS-GRASP is the best } A, \mu, F, \Gamma = \{ \Gamma(k), k \in K \} \text{ ever obtained.}\]

(I3) involves the TRANS operator as follows:

\[\text{Possible; While Possible do Trial-Number <- 1; Success <- False; Do Until Success or Trial-Number > Loop Generate } K_0 \subseteq K \text{ with cardinality } Q; \text{ Save current } A, \mu, F, \Gamma = \{ \Gamma(k), k \in K \}; (I4) \text{ Apply TRANS}(K_0) \text{ to } A, \mu, F, \Gamma; \text{ If } c.F + p.g \text{ is improved then Success Else Restore } A, \mu, F, \Gamma; \text{ Trial-Number <- Trial-Number + 1; Possible <- Success;}}\]

Choosing $K_0$ in the (I4) Instruction: it is defined by the paths $\{ \Gamma(k), k \in K \}$ which contain the arcs $e$ with the highest $(\mu_e + p_e)$ values.

A Random Walk Variant of FMS-GRASP: Because of the computing costs induced by Instruction (I4), we also implement a Random Walk strategy.

FMS-GRASP-1($R$: Replication Number; $RW$: Loop Length Bound; $Q$: Subset Size);
For $i = 1..R$ do Initialize $A, \mu, F, \Gamma = \{ \Gamma(k), k \in K \}$ through FMS-INIT;
For Counter = 1..$RW$ do (I4.1: Random Walk loop) Generate $K_0 \subseteq K$ with cardinality $Q$; Apply TRANS($K_0$) to $A, \mu, F, \Gamma$; The result is the best $(A, \mu, F, \Gamma)$ ever obtained.

V. A GENETIC ALGORITHM FMS-GEN FOR FMS

The main components of a Genetic algorithm are (see [6, 7, 8]): its Encoding scheme (Chromosome Representation); the Initialization Procedure which yields the initial population $\Sigma$; its Mutation operator; its Crossover operator.

Clearly, the Encoding scheme is the encoding scheme of Section III.A whose master objects are:
- the arc support set $\text{Arc-Supp}(F) = \{ e \in E \text{ such that } F_e \neq 0 \}$ of $F$;
- the related cost vector $\mu = \mu_e, e \in A$;
and the slave objects are the flow vector $F$ and the path collection $\Gamma = \{ \Gamma(k), k \in K \}$.
Initialization is performed through Card($\Sigma$) successive applications of FMS-INIT.
Mutation results from application of the operator TRANS, with parameter $K_0$ generated with a given cardinality $Q$, $Q$ becoming a parameter of the global process:

\[
\text{FMS-Mutation}(A, \mu, F, \Gamma = \{ \Gamma(k), k \in K \}, Q); \text{ Generate some subset } K_0 \subseteq K \text{ with cardinality } Q; \text{ Apply TRANS($K_0$)} \text{ to } A, \mu, F, \Gamma;\]

The FMS-Crossover crossover operator is the SON-CREATE operator of Section III.F.

What remains to be discussed here is the Fitness Criterion, and the way FMS-Crossover is applied:
- Given $(A_1, \mu_1, F_1, \Gamma_1 = \{ \Gamma_1(k), k \in K \})$ and $(A_2, \mu_2, F_2, \Gamma_2 = \{ \Gamma_2(k), k \in K \})$ in current population $\Sigma$. Fitness is related here to the cardinality of the difference set $(A_1 - A_2) \cup (A_2 - A_1)$; the smallest is it, the largest is the Fitness measurement;
- Best-fitted pairs are selected, in order to avoid cloning, with the constraint that no solution $\sigma$ belongs to more than 2 pairs. It is done in a heuristic way.

The main parameters of the deriving Genetic Algorithm FMS-GEN are the population size $P$, the number $L$ of iterations of mutation/crossover process and the size $Q$. 
VI. NUMERICAL EXPERIMENTS

Experiments are performed on a LINUX server CentOS 5.4, Processor Intel Xeon 3.6 GHZ, with help of the CPLEX 12 library.

A. Instance Generation

An instance is defined by: the Urban Area network \( H = (Z, U) \), with \( n \) vertices and \( m \) arcs; demands \( D_k, k \in K \); walking lengths \( l_w(e), e \in E \); vehicle lengths \( l_v(e), e \in U \); Vehicle cost vector \( c \) and User cost vector \( p \); the size \( N \) of the time-space; the arc number \( NA \) of \( H-Dyn \). We generate our own small and large instances: nodes of \( H \) are points of the 2D Euclidean space, with adjacency related to distance thresholds; demands \( D_k, k \in K \) are randomly generated through uniform distribution.

B. Evaluation of FMS-INIT.

We first consider small instances, for which we get an exact optimal value through the CPLEX.12 Library, and next consider larger size instances, with focus on the large scale issue. In both cases:

- \( n, m \) are respectively the node and arc numbers of \( H \), \( N \) is the period number, \( NOD \) is the number of demands; \( L \) is the mean value of loads \( L_k, k \in K \), and \( \alpha \) is the mean ratio \( p_l/c, e \in E \).

- \( R \) is the replication number of FMS-INIT.

Small instances: We focus on precision of FMS-INIT, and test packages of 10 instances. GAP-MEAN is the mean error \( GAP = (VAL – OPT)/OPT \), \( VAL \) = cost value computed by FMS-INIT, OPT = optimal value computed by CPLEX.12. GAP-VAR is the variance of GAP. We get:

<table>
<thead>
<tr>
<th>( R )</th>
<th>GAP-MEAN</th>
<th>GAP-VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22.5</td>
<td>25.8</td>
</tr>
<tr>
<td>5</td>
<td>17.3</td>
<td>20.7</td>
</tr>
<tr>
<td>10</td>
<td>13.4</td>
<td>17.9</td>
</tr>
<tr>
<td>20</td>
<td>11.9</td>
<td>15.4</td>
</tr>
<tr>
<td>50</td>
<td>10.5</td>
<td>13.6</td>
</tr>
</tbody>
</table>

Table 1: FMS-INIT Evaluation, Small Instances, 10 instances/packages; Group-Instance: \( n = 10, m = 30, N = 10; NOD = 10; \alpha = 0.5; L = 0.2; \) Impact of \( R \).

Analysis: Parameter \( R \) plays a key role. GAP-VAR is usually large: a single FMS-INIT run may yield poor solutions.

Large instances: We focus here on CPU times and on the sensitivity to parameter \( R \): \( V(R) \) is the mean value \( (Max(R) - Min(R))/Min(R) \), where \( Max(R) \) and \( Min(R) \) are respectively the worse and best values obtained through FMS-INIT(R), while \( Var-V(R) \) is the related variance. Mean-CPU is the mean running time, while \( Var-CPU \) is the related variance.

<table>
<thead>
<tr>
<th>( R )</th>
<th>( V(R) )</th>
<th>( Var-V(R) )</th>
<th>Mean-CPU (in s)</th>
<th>Var-CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>24.8</td>
<td>12.5</td>
</tr>
<tr>
<td>5</td>
<td>0.08</td>
<td>0.05</td>
<td>98.5</td>
<td>67.2</td>
</tr>
<tr>
<td>10</td>
<td>0.14</td>
<td>0.04</td>
<td>177</td>
<td>101.0</td>
</tr>
<tr>
<td>20</td>
<td>0.18</td>
<td>0.04</td>
<td>329</td>
<td>151.4</td>
</tr>
<tr>
<td>50</td>
<td>0.22</td>
<td>0.03</td>
<td>745</td>
<td>256.1</td>
</tr>
</tbody>
</table>

Table 2: FMS-INIT Evaluation, Large Instances, 10 instances/packages; Group-Instance: \( N = 66256; NOD = 100; \alpha = 0.5; L = 0.2; \) Impact of \( R \).

Analysis: The replication mechanism is crucial.

C. Evaluation of FMS-GRASP.

We focus on the respective ability of the standard Descent loop with parameter \( TH \) and of the random walk with parameter \( RW \) to improve the initial solution.

Small instances, 10 instances/packages with \( n = 10, m = 30, N = 10; NOD = 10, L = 0.2; \alpha = 0.5; \) GAP-MEAN is the mean error \( GAP = (VAL – OPT)/OPT \), where \( VAL \) is computed by FMS-GRASP, and GAP-VAR is the variance of GAP. We use \( R = 10, Q = 3 \).

<table>
<thead>
<tr>
<th>( TH )</th>
<th>GAP-MEAN</th>
<th>GAP-VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.1</td>
<td>17.8</td>
</tr>
<tr>
<td>4</td>
<td>11.3</td>
<td>11.0</td>
</tr>
<tr>
<td>8</td>
<td>9.4</td>
<td>9.5</td>
</tr>
<tr>
<td>15</td>
<td>7.2</td>
<td>8.3</td>
</tr>
</tbody>
</table>

Table 3: FMS-GRASP/Descent: Impact of \( TH \)

<table>
<thead>
<tr>
<th>( RW )</th>
<th>GAP-MEAN</th>
<th>GAP-VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.5</td>
<td>17.8</td>
</tr>
<tr>
<td>4</td>
<td>11.5</td>
<td>13.1</td>
</tr>
<tr>
<td>10</td>
<td>9.3</td>
<td>11.8</td>
</tr>
<tr>
<td>40</td>
<td>6.8</td>
<td>8.8</td>
</tr>
<tr>
<td>80</td>
<td>5.8</td>
<td>5.1</td>
</tr>
</tbody>
</table>

Table 4: FMS-GRASP/Random Walk: Impact of \( RW \)

Analysis: Random Walk is more efficient than Descent.

Large instances, 10 instances/packages, with \( N = 66256; NOD = 100; \alpha = 0.5; L = 0.2; \) IMPROVE = \( (Min(R) - Val(R, RW))/Val(R, W) \), where \( Min(R) \) is computed by FMS-INIT(R) and \( Val(R, W) \) is computed by FMS-GRASP(R, RW). IMPROVE(R, RW) is the mean IMPROVE Value.

<table>
<thead>
<tr>
<th>( R )</th>
<th>( RW )</th>
<th>IMPROVE(R, RW)</th>
<th>Mean-CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>2.3</td>
<td>39</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>9.8</td>
<td>249</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1.7</td>
<td>151</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>9.1</td>
<td>1012</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1.4</td>
<td>257</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>6.7</td>
<td>1618</td>
</tr>
</tbody>
</table>

Table 5: FMS-GRASP Evaluation, Large Instances: Impact of \( R \) and \( RW \).
Analysis: Computing times remain under control. Improvement margin induced by the Random Walk loop are close to the values obtained for small instances.

D. Evaluation of FMS-GEN.

We use the same tests as in Section VI.C. \( P \) is the size of the population, \( LG \) is the length of the main loop of the process. The population \( \Sigma \) is initialized by \( \text{FMS-INIT}(P) \).

**Small instances:** 10 instance packages with \( n = 10, m = 30, NP = 10; NOD = 10; L = 0.2; \alpha = 0.5; \text{GAP-MEAN} \) is the mean error \( \text{GAP} = (\text{VAL} - \text{OPT})/\text{OPT} \), where \( \text{VAL} \) is the cost value of the solution which is computed by \( \text{FMS-GEN} \), and \( \text{OPT} \) is the optimal result computed by CPLEX.12 \( \text{GAP-VAR} \) is the variance of \( \text{GAP} \). We focus on difficult instances, and deal with rather small populations (no more than 30) and small \( LG \) values. We use \( P = 10, Q = 3; \Pi = 1; \) \( \Sigma = 66256; \text{Val} = 0.2. \)

<table>
<thead>
<tr>
<th>( P )</th>
<th>GAP-MEAN</th>
<th>GAP-VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10.3</td>
<td>14.7</td>
</tr>
<tr>
<td>10</td>
<td>6.2</td>
<td>8.0</td>
</tr>
<tr>
<td>20</td>
<td>4.2</td>
<td>5.4</td>
</tr>
<tr>
<td>30</td>
<td>3.8</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Table 6: FMS-GEN Evaluation, Impact of \( P \).

**Large instances:** For any instance, we evaluate the improvement ratio \( \text{IMPROVE} = (\text{Min}(P) - \text{Val}(P, LG))/\text{Val}(P, LG) \), where \( \text{Min}(P) \) is the value obtained while running \( \text{FMS-INIT}(P) \) and \( \text{Val}(P, LG) \) is the value obtained while running \( \text{FMS-GEN}(P, LG) \). \( \text{IMPROVE}(P, LG) \) is the mean \( \text{IMPROVE} \) value on 5 instance package defined by parameter values: \( NA = 66256; NOD = 100; \alpha = 0.5; L = 0.2. \) We use \( Q = 15 \) and \( \Pi = 1 \).

<table>
<thead>
<tr>
<th>( LG )</th>
<th>GAP-MEAN</th>
<th>GAP-VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9.3</td>
<td>11.7</td>
</tr>
<tr>
<td>20</td>
<td>9.1</td>
<td>11.4</td>
</tr>
<tr>
<td>50</td>
<td>6.2</td>
<td>7.0</td>
</tr>
<tr>
<td>100</td>
<td>2.9</td>
<td>4.1</td>
</tr>
</tbody>
</table>

Table 7: FMS-GEN Evaluation, Impact of \( LG \).

**General comment:** The GRASP scheme is less accurate the the GA scheme, but it is more flexible and tackles more easily large scale instances. When it comes to practical applications, accuracy is not such an issue. So it comes that we may consider here that, from this point of view, GRASP performs better.

VII. CONCLUSION

Reformulating the FMS model through through implicit representations allows us to design efficient GRASP and genetic algorithms. Still, we notice that since those algorithms rely on sophisticated LP techniques, we should now study the way to efficiently involve recently emerging generic framework, like ILP software SCIP/CPLEX, in such a way development and maintenance costs be minimized.

REFERENCES