GATeS: A Hybrid Algorithm Based on Genetic Algorithm and Tabu Search for the Direct Marketing Problem

MÜLLER F.M.[1], SCHNEIDER V. A.[2], BONILHA I. S.[2], SOUZA V. B. S.[2], CRUZ G. R. S. and MAVROVOUNIOTIS M.[3]

[1]Department of Applied Computing, Federal University of Santa Maria, Building 07, office 322, Santa Maria, 97105-900, RS, Brazil PPGEP, Federal University of Santa Maria, Building 07, Santa Maria, 97105-900, RS, Brazil PPCC, Federal University of Santa Maria, Building 07, Santa Maria, 97105-900, RS, Brazil

[2]Department of Postgraduate Production Engineering, University of Santa Maria, Building 07, Santa Maria, 97105-900, RS, Brazil

[3]ERATOSTHENES Centre of Excellence, Limassol, Cyprus

Algorithms

Algorithm 1 Constructive Algorithm adapted from TS

1: $val \leftarrow 0,\ exp \leftarrow 0,\ V \leftarrow \{1,\dots,m\},\ S \leftarrow \{\ \}$
2: **for** $each\ j = 1,\dots,n$ and **for** $each\ i \in V$, **compute :** $NPP_{ij} = (p_{ij} - c_{ij})/c_{ij}$
3: **for** $all\ products\ j \notin S$ **do**
4: $VNPP_j \leftarrow$ clients in V sorted in non-increasing order of their NPP_{ij}
5: $C_j \leftarrow$ the cost of offers to the first O_j clients in $VNPP_j$
6: $P_j \leftarrow$ the revenue of offers to the first O_j clients in $VNPP_j$
7: $PR_j \leftarrow P_j - C_j - f_j$
8: **end for**
9: select $j^* \notin S$ with the highest PR_{j^*} and satisfying $C_{j^*} \leq B_{j^*}$, and $val + P_{j^*} \geq (1+H)(exp + C_{j^*} + f_{j^*})$, and $\{\nexists(x \in S) \mid (x, j^*) \in CAN\ or\ (j^*, x) \in CAN\}$
10: **if** $j^*\ exists$ and $PR_{j^*} > 0$ **then**
11: $S \leftarrow S \cup \{j^*\},\ y_{j^*} \leftarrow 1,\ val \leftarrow val + P_{j^*},\ exp \leftarrow exp + C_{j^*} + f_{j^*}$
12: **for** $each\ client\ i\ amongst\ the\ first\ O_{j^*}\ in\ VNPP_{j^*}$ **do**
13: $x_{ij^*} \leftarrow 1,\ M_i \leftarrow M_i - 1$
14: **if** $M_i = 0$ **then**
15: $V \leftarrow V \setminus \{i\}$
16: **end if**
17: **end for**
18: return to Step 3
19: **end if**
20: **for** $all\ products\ j \notin S$ **execute Algorithm 2**
21: **for** $each\ active\ client\ i$ **do**
22: **for** $each\ j \in S$ **do**
23: **if** $p_{ij} > c_{ij}$ and the offer of product j to client i leads to a feasible solution **then** make that offer and update the current solution
24: **end for**
25: **end for**

Algorithm 2 Algorithm for additional product selection: (*build_minimum_offer_set* function)

$VC_j \leftarrow$ clients in V sorted in non-decreasing order of their c_{ij}
$C_j \leftarrow$ the cost of offers to the first O_j clients in VC_j
$P_j \leftarrow$ the revenue of offers to the first O_j clients in VC_j
$PR_j \leftarrow P_j - C_j - f_j$
if $(C_j \leq B_j)$ and $\{\nexists(x \in S) \mid (x, j^*) \in CAN \text{ } or \text{ } (j^*, x) \in CAN\}$ **then**
  $client\_i_{j^*} \leftarrow$ first O_{j^*} entries of VC_{j^*}
  repeat
    select a client i from $client\_i_{j^*}$, with the minimum NPP_{ij^*}
    select a client k from $V \setminus client\_i_{j^*}$, with the maximum NPP_{ij^*}
    if $c_{kj} - c_{ij} \leq B_j - C_{j^*}$ **then**
      $client\_i_{j^*} \leftarrow client\_i_{j^*} \cup \{k\} \setminus \{i\}$
      $C_{j^*} \leftarrow C_{j^*} + c_{kj} - c_{ij}$
      $P_{j^*} \leftarrow P_{j^*} + p_{kj} - p_{ij}$
    end if
  until $c_{kj} - c_{ij} \leq B_j - C_{j^*}$
  if $val + P_{j^*} \geq (1 + H)(exp + C_{j^*} + f_{j^*})$ **then**
    $S \leftarrow S \cup \{j^*\},\ val \leftarrow val + P_{j^*},\ exp \leftarrow exp + C_{j^*} + f_{j^*}$
    for each client i from set $client\_i_{j^*}$ **do**
      $x_{ij^*} \leftarrow 1,\ M_i \leftarrow M_i - 1$
      if $M_i = 0$ **then**
        $V \leftarrow V \setminus \{i\}$
      end if
    end for
  end if
end if

Algorithm 3 Constructive Greedy Randomized Adaptive Procedure (Modifications in Algorithm 1)

Replace Lines 3-7 from Algorithm 1 with:

for *all products* $j \notin S$ **do**
  $client\_i_j \leftarrow \{\ \},\ C_j \leftarrow 0,\ P_j \leftarrow 0$
  $VNPP_j \leftarrow$ clients in V sorted in non-increasing order of their NPP_{ij}
  while $|client\_i_j| \leq O_j$ **do**
    select i randomly from the first 10% entries of $\{VNPP_j \setminus client\_i_j\}$, and $C_j + c_{ij} \leq B_j$
    $client\_i_j \leftarrow client\_i_j \cup \{i\}$
    $C_j \leftarrow C_j + c_{ij}$
    $P_j \leftarrow P_j + p_{ij}$
  end while
  $PR_j \leftarrow P_j - C_j - f_j$
end for

Replace Line 11 from Algorithm 1 with:

for *each client* i *from set* $client\_i_{j^*}$ **do**

Algorithm 4 Variant of the Constructive Greedy Randomized Adaptive Procedure (Modifications in Algorithm 1)

Replace Lines 3-7 from Algorithm 1 with:

for *all products* $j \notin S$ **do**
  $client\_i_j \leftarrow \{\ \},\ C_j \leftarrow 0,\ P_j \leftarrow 0$
  $VNPP_j \leftarrow$ clients in V sorted in non-increasing order of their NPP_{ij}
  repeat
    select i randomly from the first 10% entries of $\{VNPP_j \setminus client\_i_j\}$
    if $C_j + c_{ij} \leq B_j$ **then**
      $client\_i_j \leftarrow client\_i_j \cup \{i\}$
      $C_j \leftarrow C_j + c_{ij}$
      $P_j \leftarrow P_j + p_{ij}$
    end if
  until $C_j + c_{ij} > B_j$
  $PR_j \leftarrow P_j - C_j - f_j$
end for

Replace Line 11 from Algorithm 1 with:

for $each\ client\ i\ from\ set\ client\_i_{j^*}$ **do**

Algorithm 5 Repair Based Constructive Algorithm

$val \leftarrow 0,\ exp \leftarrow 0,\ V \leftarrow \{1, \ldots, m\},\ S \leftarrow \{\ \}$
for *each* $j = 1, \ldots, n$ and **for** *each* $i \in V$, **compute :** $NPP_{ij} \leftarrow (p_{ij} - c_{ij})/c_{ij}$
for *all products* $j \notin S$ **do**
  $client\_i_j \leftarrow \{\ \},\ sclient\_i_j \leftarrow \{\ \},\ C_j \leftarrow 0,\ P_j \leftarrow 0,\ PR_j \leftarrow 0,\ sPR_j \leftarrow 0$
  $VNPP_j \leftarrow$ clients in V sorted in non-increasing order of their NPP_{ij}
  while $(C_j < 5B_j)$ *or* $(|client\_i_j| = m)$ **do**
    pick i as the first entry of the set $VNPP_j \setminus client\_i_j \setminus sclient\_i_j$
    if $C_j + c_{ij} < B_j$ **then**
      $client\_i_j \leftarrow client\_i_j \cup \{i\}$
      $C_j \leftarrow C_j + c_{ij}$
      $P_j \leftarrow P_j + p_{ij}$
      $PR_j \leftarrow PR_j + p_{ij} - c_{ij}$
    else
      $sclient\_i_j \leftarrow sclient\_i_j \cup \{i\}$
      $C_j \leftarrow C_j + c_{ij}$
      $sPR_j \leftarrow sPR_j + p_{ij} - c_{ij}$
    end if
  end while
end for
for *each product* $j \notin S$ *ordered in non−decreasing order of* $(PR_j + 0.2sPR_j)$ **execute Algorithm 6** (Conflict Management)
for *each active client* i **do**
  for *each active client* i **do**
    if $p_{ij} > c_{ij}$ and the offer of product j to client i leads to a feasible solution **then** make that offer and update the current solution
  end for
end for

Algorithm 6 Function for Managing Offer Conflicts

```
if $(|client\_i_j = O_j|)$ and $\{\nexists(x \in S) \mid (x,j) \in CAN \text{ or } (j,x) \in CAN\}$ then
  for each client $i \in client\_i_j$ do
    if $i \in V$ then
      $x_{ij} \leftarrow 1,\ M_i = M_i - 1$
      if $M_i = 0$ then $V \leftarrow V \setminus \{i\}$
    else
      for each product $j^* \in S \setminus \{j\}$ do
        if $|client\_i_{j^*}| > 0$ then find the minimum cost feasible replacement of a client $i$ offered in $j$ by a client $k \in sclient\_i_{j^*}$
      end for
      if $k$ exists then
        if $M_i = 0$ then $V \leftarrow V \cup \{i\}$
        $x_{ij} \leftarrow 0,\ M_i \leftarrow M_i - 1,\ x_{kj} \leftarrow 1,\ M_k \leftarrow M_k - 1$
        $client\_i_j \leftarrow client\_i_j \setminus \{i\},\ client\_i_j \leftarrow client\_i_j \cup \{k\},\ sclient\_i_j \leftarrow sclient\_i_j \cup \{i\}$
        if $M_k = 0$ then $V \leftarrow V \setminus \{k\}$
        $C_{j^*} \leftarrow C_{j^*} + c_{kj^*} - c_{ij^*},\ P_{j^*} \leftarrow P_{j^*} + p_{kj^*} - p_{ij^*},\ val \leftarrow val + p_{kj^*} - p_{ij^*},\ exp \leftarrow exp + c_{kj^*} - c_{ij^*}$
      else
        if $|sclient\_i_j| > 0$ then find the minimum cost feasible replacement of a client $i$ offered in $j$ by a client $h \in sclient\_i_j$
        if $h$ exists then
          $x_{ij} \leftarrow 0,\ x_{hj} \leftarrow 1,\ M_h \leftarrow M_h - 1$
          $client\_i_j \leftarrow client\_i_j \setminus \{i\},\ client\_i_j \leftarrow client\_i_j \cup \{h\},\ sclient\_i_j \leftarrow sclient\_i_j \setminus \{h\}$
          if $M_h = 0$ then $V \leftarrow V \setminus \{h\}$
          $C_j \leftarrow C_j + c_{hj} - c_{ij},\ P_j \leftarrow P_j + p_{hj} - p_{ij}$
        else
          if $|sclient\_i_j| > O_j$ then
            $x_{ij} \leftarrow 0,\ client\_i_j \leftarrow client\_i_j \setminus \{i\},\ C_j \leftarrow C_j - c_{ij},\ P_j \leftarrow P_j - p_{ij}$
          else
            the product $j$ can not be offered, reverse all actions taking so far
            $y_j \leftarrow 0,\ S \leftarrow S \setminus \{j\}$
            return to Algorithm 5
          end if
        end if
      end if
    end if
  end for
  if $val + P_j \geq (1+H)(exp + C_j + f_j)$ then
    $S \leftarrow S \cup \{j\}, y_j \leftarrow 1,\ val \leftarrow val + P_j,\ exp \leftarrow exp + C_j + f_j$
  else
    the product $j$ can not be offered, reverse all actions taking so far
    $y_j \leftarrow 0,\ S \leftarrow S \setminus \{j\}$
  end if
end if
```

Algorithm 7 Neighborhood 3

```
Initialize the sets to store movement's information, IC_1 ← { }, IC_2 ← { }
SC ← clients in V sorted in non-increasing order of their variance of p_ij for all products
for each client i ∈ SC do
    current_tuple ← { }
    for each j ∈ S and x_ij = 1 do
        for each l ∈ S and x_il = 0 do
            while current_tuple = { } do
                if p_ij < p_il then
                    current_tuple ← {(j, l, i)}
                    if ∃h|(l, j, h) ∈ IC_1 then
                        if (l, j, h) makes current_tuple feasible then
                            x_ij ← 0, x_il ← 1
                            x_hj ← 1, x_hl ← 0
                            IC_1 ← IC_1 \ (l, j, h)
                            if ∃m|(l, j, m) ∈ IC_2 then
                                IC_1 ← IC_1 ∪ (l, j, m)
                                IC_2 ← IC_2 \ (l, j, m)
                            end if
                        else
                            if (∃m|(l, j, m) ∈ IC_2) and (l, j, m) makes current_tuple feasible then
                                x_ij ← 0, x_il ← 1
                                x_mj ← 1, x_ml ← 0
                                IC_2 ← IC_2 \ (l, j, m)
                            else
                                if current_tuple is a feasible movement then
                                    x_ij ← 0, x_il ← 1
                                else
                                    if ∀x ∄(j, l, x) ∈ IC_2 then
                                        IC_2 ← IC_2 ∪ {current_tuple}
                                    else
                                        if current_tuple is more profitable than any x|(j, l, x) ∈ IC_1 then
                                            IC_1 ← IC_1 ∪ {current_tuple}
                                        else
                                            if current_tuple is more profitable than any x|(j, l, x) ∈ IC_2 then
                                                IC_2 ← IC_2 ∪ {current_tuple}
                                            end if
                                        end if
                                    end if
                                end if
                            end if
                        end if
                    end if
                end if
            end while
        end for
    end for
end for
for each j ∈ S do
    for each active client i examined in non-increasing order of their NPP_ij with x_ij = 0 do
        if offer product j to client i is feasible then
            x_ij ← 1
        end if
    end for
end for
```

Algorithm 8 Tabu Search Procedure, where $of(x, y)$ is the objective function value of solution (x, y)

$Iter \leftarrow 0, tabu\_list \leftarrow \{\ \}$
choose an initial solution (x, y)
$x^* \leftarrow x' \leftarrow x,\ y^* \leftarrow y' \leftarrow y$
while $Iter < 30$ && $time < 300s$ **do**
  $(x_{prev}, y_{prev}) \leftarrow (x', y')$
  $(x_{Iter}, y_{Iter}) \leftarrow explore\ N_1(x', y'),$ considering tabu_list
  if $of(x_{Iter}, y_{Iter} > of(x', y')$ **then**
    $x' \leftarrow x_{Iter},\ y' \leftarrow y_{Iter},$ update tabu_list
  end if
  $(x_{Iter}, y_{Iter}) \leftarrow explore\ N_2(x', y'),$ considering tabu_list
  if $of(x_{Iter}, y_{Iter} > of(x', y')$ **then**
    $x' \leftarrow x_{Iter},\ y' \leftarrow y_{Iter},$ update tabu_list
  end if
  $(x_{Iter}, y_{Iter}) \leftarrow explore\ N_3(x', y'),$ considering tabu_list**if** of(x$_{Iter}, y_{Iter} > of(x', y')$ **then**
    $x' \leftarrow x_{Iter},\ y' \leftarrow y_{Iter},$ *update tabu_list*
  end if
  if $of(x_{prev}, y_{prev}) \leq of(x', y')$ **then**
    $Iter \leftarrow Iter + 1$
    if $(Iter\ mod\ 10) = 0$ **then**
      Update GA Population
      Expand TS Neighborhood Size
    end if
    if $Iter > 1$ **then**
      $\alpha = \alpha + 0.03$
      $(x', y') \leftarrow$ *perform Genetic Regression for* (x', y') *and* α
    else
      $(x', y') \leftarrow$ *perform Genetic Optimization for* (x', y')
    end if
  end if
  if $of(x^*, y^*) < of(x', y')$ **then**
    $x^* \leftarrow x',\ y^* \leftarrow y'$
    $Iter \leftarrow 0,\ \alpha \leftarrow 0.0$
    Reset TS Neighborhood Size
  end if
end while

Algorithm 9 Genetic Algorithm Procedure

Initialize the *Population* performing 100 times the Algorithm 3
$BS \leftarrow$ *individual with highest fitness on Population*
$Generation \leftarrow 0$, *offer_list* $\leftarrow \{ \}$
while $Generation < 4$ **do**
    $New\_Population \leftarrow \{ \}$
    $Elite \leftarrow BS$
    while $|New\_Population| < 99$ **do**
        Parent1 $\leftarrow$ *fitness_weighted_roulette_wheel(Population)*
        Parent2 $\leftarrow$ *fitness_weighted_roulette_wheel(Population* $\setminus$ $\{$*Parent1*$\}$*)*
        Dominant $\leftarrow$ *individual with highest fitness between Parent1 and Parent2*
        Parent2 $\leftarrow$ *individual with lowest fitness between Parent1 and Parent2*
        offspring $\leftarrow$ *apply Algorithm 10 (Crossover) for Dominant and Parent2*
        if *fitness(offspring)* $<$ *fitness(Elite* **then**
            if $\exists(k, j)|k \in Parent1$ && $j \in Parent2$ && $(k, j) \in CAN$ **then**
                Dominant $\leftarrow$ *individual with lowest fitness between Parent1 and Parent2*
                Parent2 $\leftarrow$ *individual with highest fitness between Parent1 and Parent2*
                offspring $\leftarrow$ *apply Algorithm 10 (Crossover) for Dominant and Parent2*
            end if
        end if
        if *(fitness(offspring)* $<$ *fitness(Elite)) && (Generation* $>$ *2)* **then**
            if $random(0, 1) \leq 0.75$ **then**
                Apply Mutation to offspring using offer_list
            end if
        end if
        $New\_Population \leftarrow New\_Population \cup \{$ *offspring* $\}$
        if *fitness(offspring)* $>$ *fitness(Elite)* **then**
            Elite $\leftarrow$ *offspring*
        end if
    end while
    Population $\leftarrow$ *New_Population* $\cup$ $\{$*Elite*$\}$
    if *fitness(Elite)* $>$ *fitness(BS)* **then**
        BS $\leftarrow$ *Elite*
    end if
    offer_list $\leftarrow$ *buil similarity list from New_Population*
    Generation $\leftarrow$ *Generation* $+1$
end while

Algorithm 10 Crossover Procedure

offspring $\leftarrow \{\ \}$, *Losing_Genes* $\leftarrow \{\ \}$
$\{\forall(x,j) \in CAN \mid (x \neq j)$ *and* $[(y_x = 1, y_j = 1) \in (Dominant \cup Parent2)]\}$ *set the products in CAN from Dominant as infeasible and eliminate all offers allocated to them*
$G \leftarrow$ *set of genes sorted in non-increasing order of the expected profit of the client*
for each $i \in G$ **do**
  if *viable offers of* $i \in Dominant$ *are more profitable than offers of* $i \in Parent2$ **then**
    if $\exists[($*a product* $j)$ *in* $i \in Dominant] \mid$ *number of clients in* j *does not reach* O_j **then**
      set j *as infeasible and eliminate all offers allocated to it in offspring*
    end if
    transfer all feasible offers $i \in Dominant$ *to the same gene in offspring*
    Losing_Genes $\leftarrow$ *Losing_Genes* $\cup\{i \in Parent2\}$
  else
    if $\exists[($*a product* $j)$ *in* $i \in Parent2] \mid$ *number of clients in* j *does not reach* O_j **then**
      set j *as infeasible and eliminate all offers allocated to it in offspring*
    end if
    transfer all feasible offers $i \in Parent2$ *to the same gene in offspring*
    Losing_Genes $\leftarrow$ *Losing_Genes* $\cup\{i \in Dominant\}$
  end if
end for
for *each (product* $j \in$ *offspring)* $\mid$ *number of clients in* j *does not reach* O_j **do**
  insert clients, sorted in non-decreasing order of NPP_{ij} *into product* j *until reach* O_j
  $C_j \leftarrow$ *sum of the client's cost allocated to* j
  if $C_j > B_j$ **then**
    remove all offers allocated to product j *from offspring*
  end if
end for
for *each gene* $g \in$ *Losing_Genes* **do**
  transfer all feasible offers from g *to offspring*
end for
for *each (product* $j \in$ *offspring)* $\mid C_j < B_j$ **do**
  insert clients, sorted in non-decreasing order of NPP_{ij} *into product* j *until there is budget*
end for