Team Orienteering Problem with Decreasing Profits

H. Murat Afsar* 1 Nacima Labadie* 2

*ICD-LOSI, STMR UMR 6279 CNRS
University of Technologie of Troyes
12, rue Marie Curie, CS 42060, 10004, Troyes, France

Abstract
Team Orienteering with Decreasing Profits (DP-TOP) extends the classical Team Orienteering problem (TOP) by considering the profit of each client as a decreasing function of time. It consists of maximizing the sum of collected profit by a fixed number $K$ of vehicles, visiting each client at most once. In this work, we present lower bounds based on a Dantzig-Wolfe decomposition and column generation as well as upper bounds obtained by an evolutionary local search approach (ELS).

Keywords: Team Orienteering problem, variable profits, column generation, ELS

1 Introduction

Orienteering problems are initially inspired by a sport game, where each competitor has to build a path between the starting and end points. They obtain a score (or profit) every time they visit a check point. The objective is to maximize the scores collected. This problem was defined by Tsiligrides [10]

1 Email: murat.afsar@utt.fr
2 Email: nacima.labadie@utt.fr
and consists in building a single tour so that the total profit collected is maximized, subject to time restriction on the tour length. When more than one tours are to be built ($K > 1$), the problem is called Team Orienteering problem and was described the first time by Chao [1]. For a survey on routing problems with profits see Vansteenwegen et al. [11] and Feillet al. [6].

This paper is dedicated to an extension of the classical Team Orienteering problem where the profit associated to each node is not constant but is a decreasing function of time. To our knowledge, this problem is not addressed before despite its importance in real life applications. As an example, in the field of repairing-maintenance, a client can be ready to pay more to be serviced earlier and can loose his interest as the time passes by. Another application can be found in search and rescue operations after an earthquake for example. For the success of such operations a quick and reliable situation assessment is crucial. There may be several positions where the survivors can be found with different probabilities decreasing with time. The aim is to visit as many positions as possible with higher probabilities.

The paper is organized as follows: a formal problem definition is given in section 2. A column generation approach and a metaheuristic method based on Evolutionary Local Search (ELS) are proposed in sections 3 and 4, respectively. The results obtained by both approaches are presented in section 5. Finally, section 6 is dedicated to concluding remarks and some perspectives.

## 2 Problem Definition

The Team Orienteering Problem with Decreasing Profits can be described on an undirected graph $G = (V, E)$ where $V = \{0, 1, ..., n + 1\}$ is the set of nodes. Nodes 1, ..., $n$ are potential customers to visit, whereas nodes 0 and $n + 1$ correspond respectively to beginning and end points of the paths to build. To each customer $i \in \{1, 2, ..., n\}$, a variable profit ($m_i$) and a fixed profit ($n_i$) are associated. The profit collected at node $i$ if it is visited at the moment $t$ is equal to $m_i * t + n_i$. As it is mentioned, the profits are decreasing by time, so $m_i < 0 \ \forall i \in \{1, ..., n\}$. $E$ is the set of undirected edges, each edge $[i, j]$ defines a connection between nodes $i$ and $j$ and is weighted by $w_{ij}$, the time required to traverse it in any direction. From now on, for the sake of simplicity, each edge is replaced by two arcs of opposite directions with the same travelling time $w_{ij} = w_{ji}$. A fleet of $K$ identical vehicles is available at node 0 and the total travel time of each vehicle is limited by a constant $T_{\text{max}}$.

The objective is to maximize the collected profit subject to the fleet size and time limitation of tours, by visiting at most once each potential customer.
Let \( x_{ij} \) be a binary variable taking the value 1 if the arc \((i, j)\) is traversed and 0 otherwise. The visiting time of the node \( i \) is noted by the real variable \( t_i \).

\[
z^* = \max \sum_{i \in V} m_i t_i + \sum_{i \in V} \sum_{j \in V} x_{ij} n_i \tag{1}
\]

subject to

\[
\sum_{j \in V \setminus \{0, n+1\}} x_{0j} \leq K \tag{2}
\]

\[
\sum_{j \in V \setminus \{0\}} x_{ij} \leq 1 \quad \forall i \in V \setminus \{n+1\} \tag{3}
\]

\[
\sum_{j \in V \setminus \{0\}} x_{ij} - \sum_{j \in V \setminus \{n+1\}} x_{ji} = 0 \quad \forall i \in V \setminus \{0, n+1\} \tag{4}
\]

\[
t_j + w_{j,n+1} - \sum_{i \in V \setminus \{n+1\}} x_{ij} T_{\text{max}} \leq 0 \quad \forall j \in V \setminus \{n+1\} \tag{5}
\]

\[
t_i + x_{ij}(w_{ij} + T_{\text{max}}) - t_j \leq T_{\text{max}} \quad \forall i \in V \setminus \{n+1\} \quad \forall j \in V \setminus \{0\} \tag{6}
\]

\[
x_{ij} \in \{0,1\} \quad \forall i \in V \setminus \{n+1\} \quad \forall j \in V \setminus \{0\} \tag{7}
\]

\[
t_i \geq 0 \quad \forall i \in V \tag{8}
\]

The objective function 1 maximizes the sum of collected profits. The constraint 2 limits the fleet size. Each node is visited at most once because of constraints 3. Constraints 4 are flow constraints. The respect of time limitation on the tours is guaranteed by the constraints 5. Constraints 6 are used to define the visiting time of each node. Finally constraints 7 and 8 fix the nature of variables \( x_{ij} \) (binary) and \( t_i \) (non negative).

### 3 Column Generation

A Dantzig-Wolfe decomposition is proposed to reformulate the model 1-8: the master problem keeps the constraints 2 and 3, the rest of the constraints are pushed to the sub-problem. This method proved its value in several works such as in [3,4,5,8].

#### 3.1 Restricted Master Problem

In the path reformulation of the DP-TOP, \( \lambda_r \) is the binary variable taking the value 1 if the route \( r \) is in the optimal solution. The total profit collected by the route \( r \) is noted as \( p_r \). As the number of variables in the path reformulation is very large, the master problem is solved over a restricted subset of variables \((\Omega' \subset \Omega)\) and the variables (columns) are generated dynamically by a column...
generation procedure. The candidate columns are found by solving the sub-
problem. In the following model, \( \gamma^i_r \) is a parameter showing either the node \( i \) is serviced by the route \( r \) or not.

\[
\begin{align*}
  z^*_{LP} &= \max \sum_{r \in \Omega'} p_r \lambda_r \\
  \text{subject to} \quad & \Pi_0 \to \sum_{r \in \Omega'} \lambda_r \leq K \\
  & \Pi_i \to \sum_{r \in \Omega'} \gamma^i_r \lambda_r \leq 1 \quad \forall i \in V \setminus \{0, n + 1\} \\
  & \lambda_r \geq 0 \quad \forall r \in \Omega' 
\end{align*}
\]

In the linear relaxation of the restricted problem (RMLP), \( \Pi_0 \) and \( \Pi_i \) are the dual variables associated with the fleet size restriction (10) and covering (11) constraints.

3.2 Sub-problem

The sub-problem seeks feasible routes (or columns) subject to the constraints (4-8) which are supposed to improve the objective function (9). A route is feasible if and only if:

- It starts at 0 and ends at \( n + 1 \),
- Its total duration does not exceed \( T_{max} \),
- It does not visit a node more than once.

The route \( r \) may improve the objective function if its reduced cost is positive (\( \bar{p}_r = p_r - \sum_{i \in V} \gamma^i_r \lambda_i \Pi_i - \Pi_0 \)). This is an elementary longest path problem with decreasing prices. We can reformulate it to have a classical elementary shortest path problem by:

\[
\bar{p}_r = \sum_{i \in V} \gamma^i_r (\Pi_i - m_i t_r - n_i) + \Pi_0
\]

We use a dynamic programming approach which is a modified version of the method developed in [7]. A route \( r \) weakly dominates \( r' \) if and only if:

- \( p \) and \( p' \) have the same last node \( i \),
- \( \bar{p}_r \leq \bar{p}_{r'} \),
• Total duration of $r$ is smaller than $r'$ ($t_i^r \leq t_i^{r'}$ where $t_i^r$ is the visiting time of node $i$ by the route $r$),
• All accessible nodes for $p'$ should also be accessible also for $p$.

4 Evolutionary Local Search

Evolutionary Local Search (ELS) metaheuristic was proposed by Merz and Wolf [12] for a peer-to-peer problem in telecommunications. The general structure of the implemented ELS (see Algorithm 1) starts by generating two initial solutions (with two constructive heuristics $H_1$ and $H_2$) which are then improved by a local search procedure ($LS$). The best incumbent solution is considered as the starting one in the ELS. Each iteration of the algorithm considers the best current solution which is perturbed several times to generate $NBC$ child solutions, these resulting child solutions are then improved by Local Search ($LS$). For the next iterations, the best child solution replaces the best solution every time this later is improved, and the parameter $p$ used to control the perturbation rate is reset to its minimal value $p_{min}$.

<table>
<thead>
<tr>
<th>Algorithm 1 : General structure of the ELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $s_1$ and $s_2$ with $H_1$ and $H_2$ resp.</td>
</tr>
<tr>
<td>$s_1 := LS(s_1); s_2 := LS(s_2); BestSol := \text{argmax}(z^<em>(s_1), z^</em>(s_2));$</td>
</tr>
<tr>
<td>$p := p_{min}$</td>
</tr>
<tr>
<td>for Iter := 1 to MaxIter do</td>
</tr>
<tr>
<td>$z^*(BestChild) := 0$</td>
</tr>
<tr>
<td>for $k := 1$ to $NBC$ do</td>
</tr>
<tr>
<td>$ChildSoln := BestSoln$</td>
</tr>
<tr>
<td>$Perturb(ChildSoln, p)$</td>
</tr>
<tr>
<td>$LS(ChildSoln)$</td>
</tr>
<tr>
<td>if $z^<em>(ChildSoln) &gt; z^</em>(BestChild)$ then</td>
</tr>
<tr>
<td>$BestChild := ChildSoln$</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>if $z^<em>(BestChild) &gt; z^</em>(BestSoln)$ then</td>
</tr>
<tr>
<td>$BestSoln := BestChild$</td>
</tr>
<tr>
<td>$p := p_{min}$</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>$p := \min(p_{max}, p + 1)$</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end for</td>
</tr>
</tbody>
</table>

4.1 Initial solutions

Two constructive heuristics (denoted $H_1$ and $H_2$) have been developed to build the initial solutions used in the ELS. They are both based on insertion mechanism. In the first heuristic ($H_1$), $K$ tours are initialized each with a customer
for which \( \tau_i = \frac{n_i}{w_{0,i} + w_{i,n+1}} - m_i \) is maximal. The tours are then extended by inserting at each iteration a node \( j \) in the best possible position after node \( i \) for which \( P\text{Var}(i,j) \) is maximal. \( P\text{Var}(i,j) \) and \( \delta_{i,j} \) stand respectively for the variation of tour profit and tour duration, if \( j \) is inserted after \( i \). As for \( \delta_{i,j} \), the computation of \( P\text{Var}(i,j) \) can be done in \( O(1) \) since for each customer \( i \) visited in a given tour at position \( k \), we keep in memory the sum of variable profits \( m_j \) of all customers \( j \) visited in previous positions \( 1, 2, \ldots, k \) in the same tour. Because of the decreasing profits, some customers may become non profitable from a particular moment \( t^* \leq T_{\text{max}} \) (for a customer \( i \), this moment is such that \( m_i + t^*n_i = 0 \)). Therefore, as for vehicle routing problems with time windows, the maximum allowable delay keeping the customer profitable is also recorded. Thanks to these precautions, the variation of both profit and duration of a tour before an insertion can be made in \( O(1) \).

In the second heuristic \( H_2 \), the \( K \) tours are initialized as in \( H_1 \). The tours are extended in the next iterations by considering only the insertion positions after the last customer of each tour. The criterion used is hence different: for each potential customer \( i \) and each tour \( k = 1, \ldots, K \) ending at customer \( j_k \), the value \( \frac{n_i}{w_{0,i} + w_{i,n+1}} - m_i \) is computed and the customer giving its maximum to this criterion is inserted at the end of the corresponding tour. Both algorithms stop when there is no further possible insertion.

### 4.2 Perturbation procedure

The perturbation procedure has the role of diversification in the ELS. It is a crucial component since if the perturbation introduces strong mutation on a solution, the quality of this last could be deteriorated considerably and when the solution is not sufficiently perturbed, the algorithm would be trapped quickly in a local optima. For this reason, we proposed a perturbation mechanism which remove some customers from the current solution and replace them by other customers not visited. It consists in considering each tour of the solution and removing \( p \) customers randomly chosen from the tour. Each emptied space is fulfilled by adding non visited customers which are selected such as to increase the tour profit as much as possible. The perturbation rate is controlled thanks to a parameter \( p \) which is controlled dynamically between two values \( p_{\text{min}} \) and \( p_{\text{max}} \) as proposed in [9].

### 4.3 Local Search

The local search contains two classical moves usually used in routing problems (\( \text{Or-Opt, Exchange} \)) and two new moves more suitable to problems with profits
These moves are tested in the order given by the list below and the search stops when no improving move can be found.

- **AddNew**: adds a customer not yet visited to the current solution
- **Or-opt**: relocates one customer already visited in the solution
- **Exchange**: swaps the positions of two customers
- **ReplaceSequence**: deletes a sequence of at most $\beta$ consecutive customers and replaces it with a new sequence of non visited customers.

5 Experimental Results

The computational experiments were driven on TOP instances proposed by Chao et al. [2] for which we only added the variable profits. There are 7 data sets with the number of vertices $n+1=$ 21, 32, 33, 64, 66, 100 and 102. The starting and the ending points are assumed to be distinct in these instances. The problems within each data set differ in the maximal duration of the tour and in the number of tours. The number of tours ranges from 2 to 4. Due to the lack of rooms and the important number of files (a total of 387), only the results of the Set 5 (with $n+1 = 100$) are presented. This set contains 60 files (20 files for each value of $K = 2, 3, 4$). The ELS was executed on a Dell Latidude E6420 equipped with 2.4 GHz intel core i7-2760 processor and 8 GB of RAM. It was coded in Delphi, a Pascal-like programming environment. The column generation approach was written in C++ and the commercial solver CPLEX was used to solve the mathematical models in an Intel Core i5-3570 processor clocked at 3.40GHz × 4.

After several tests, the ELS parameters were set to the following values: $MaxIter = 100$, $NBC = 50$, $p_{\text{min}} = 2$, $p_{\text{max}} = 4$ and a maximum of $\beta = 4$ consecutive nodes are removed in the last neighborhood of the local search. Tables 1 and 2 summarize respectively the results obtained on Set 5 for the case where $m_i$ are randomly generated in the intervals $[-2.50, 0]$ (Tab. 1) and $[-1.1, -0.1]$ (Tab.2). Both tables follow the same structure: columns 2 and 3 give respectively the average profit for each group of instances with $K = 2, 3, 4$, when the best heuristic ($H_1$) and the ELS are executed. Column 4 indicates the average running time achieved by the ELS. The three following columns are devoted to the column generation approach (CG) and indicate respectively for each group of files the average values of the upper bound (CG-UB) and lower bound (CG-LB) as well as running time (Time-CG). The last line of the tables give the mean over the three groups of instances of Set 5.

For the Set 5 and for $m_i \in [-2.50, 0]$, almost all of the 60 instances except eight are solved to optimality by CG, while the ELS is less competitive in terms of computational time and the quality of results. Furthermore, the
column generation procedure is very fast. Its average computational time is 0.11 seconds with a maximal running time of 0.36 seconds. For the case where $m_i \in ]-1.1,-0.1[$ (Tab. 2) the column generation becomes more time consuming and the results shown in the three last columns are those obtained when the running time is limited to 600 seconds for each instance. CG still however more efficient than ELS in terms of obtained results. It is interesting to mention that execution time are very uneven for instances in the same group for $m_i \in ]-1.1,-0.1[$. For example in Set 5, there are 13 files for $K = 2$, 9 for $K = 3$ and only 4 files for $K = 4$ for which the execution time is more than 200 seconds.

<table>
<thead>
<tr>
<th></th>
<th>HEUR</th>
<th>ELS</th>
<th>Time-ELS(s)</th>
<th>CG-UB</th>
<th>CG-LB</th>
<th>Time-CG (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K = 2$</td>
<td>123.747</td>
<td>130.601</td>
<td>15.429</td>
<td>159.188</td>
<td>159.156</td>
<td>0.11</td>
</tr>
<tr>
<td>$K = 3$</td>
<td>141.824</td>
<td>150.966</td>
<td>17.582</td>
<td>185.408</td>
<td>185.160</td>
<td>0.13</td>
</tr>
<tr>
<td>$K = 4$</td>
<td>160.452</td>
<td>171.320</td>
<td>25.765</td>
<td>196.423</td>
<td>196.406</td>
<td>0.10</td>
</tr>
<tr>
<td>Aver.</td>
<td>141.023</td>
<td>149.872</td>
<td>19.297</td>
<td>179.387</td>
<td>179.286</td>
<td>0.11</td>
</tr>
</tbody>
</table>

Table 1
Summary of results on Set 5, $n + 1 = 100$, $m_i \in ]-1.1,-0.1[$

<table>
<thead>
<tr>
<th></th>
<th>HEUR</th>
<th>ELS</th>
<th>Time-ELS(s)</th>
<th>CG-UB</th>
<th>CG-LB</th>
<th>Time-CG (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K = 2$</td>
<td>254.119</td>
<td>277.400</td>
<td>32.497</td>
<td>320.233</td>
<td>319.523</td>
<td>349.558</td>
</tr>
<tr>
<td>$K = 3$</td>
<td>287.192</td>
<td>313.790</td>
<td>35.122</td>
<td>370.892</td>
<td>368.872</td>
<td>264.338</td>
</tr>
<tr>
<td>$K = 4$</td>
<td>311.552</td>
<td>353.036</td>
<td>37.959</td>
<td>407.116</td>
<td>405.821</td>
<td>148.498</td>
</tr>
<tr>
<td>Aver.</td>
<td>282.775</td>
<td>312.707</td>
<td>35.045</td>
<td>363.796</td>
<td>362.364</td>
<td>259.600</td>
</tr>
</tbody>
</table>

Table 2
Summary of results on Set 5, $n + 1 = 100$, $m_i \in ]-2.5,0[$

6 Conclusion and Perspectives

This work presents the first study dealing with the Team Orienteering with variable profits. This new variant is motivated by real applications in service sector for example. A mathematical model, a lower bound and upper bound based on column generation approach and Evolutionary Local search Metaheuristic are developed. Further research must be conducted in order to improve the performance of the ELS especially its local search. An interesting issue of this work is to understand why the methods behave differently for slightly higher values of negative variable profits. Future work would also be dedicated to other variants (other kinds of profit functions, multi-criteria case).
References


