# An ILS Algorithm for the Team Orienteering Problem with Variable Profit

**Aldy Gunawan †**
School of Information Systems
Singapore Management University, Singapore
Tel: (+65) 6808-5227, Email: aldygunawan@smu.edu.sg

**Kien Ming Ng**
Department of Industrial and Systems Engineering
National University of Singapore, Singapore
Tel: (+65) 6516-5541, Email: isenkm@nus.edu.sg

**Graham Kendall**
School of Computer Science
The University of Nottingham, Malaysia Campus
Email: Graham.Kendall@nottingham.edu.my

**Junhan Lai**
Department of Industrial and Systems Engineering
National University of Singapore, Singapore
Email: a0097547@u.nus.edu

**Abstract.** The Orienteering Problem (OP) is a routing problem that has numerous applications in various fields such as logistics and tourism. The objective is to determine a subset of nodes to visit so that the total collected score is maximized and a given time budget is not exceeded. The extensive application of OP has led to many different variants, including the Team Orienteering Problem (TOP) and the Orienteering Problem with Time Windows (OPTW). In this paper, the TOP with Variable Profits (TOPVP) is studied. The main characteristic of the TOPVP is that the amount of score collected from a particular node depends on the duration of stay on that node. We first propose a mathematical model for the TOPVP. We use the AIMMS Outer Approximation (AOA) algorithm to solve modified benchmark instances. We then propose a simple algorithm based on Iterated Local Search in order to solve some modified benchmark instances. Finally, we conclude that ILS is able to produce results which are comparable to those solved by the AOA algorithm.

**Keywords:** Orienteering problem, variable profit, mathematical model, iterated local search

## 1. INTRODUCTION

The Orienteering Problem (OP) is a multi-level optimization problem that has numerous applications in various fields such as logistics (Golden et al., 1987) and tourism (Souffriau et al., 2008). Vansteenwegen et al. (2011) formally defines the OP as a combination of node selection and determining the shortest Hamiltonian path between the selected nodes. The main objective is to determine a path, limited by the total travel time or distance, which visits some nodes in order to maximize the collected score from visited nodes.

The Team Orienteering Problem (TOP) is an extension of the OP with multiple paths. Each path is limited by the total travel time. The objective is to maximize the total collected score from all paths. Some recent works related to the TOP can be found in Dang et al. (2013), Ferreira et al. (2014) and Ke et al. (2015). There are many different variants of the OP, such as the (Team) Orienteering Problem with Time Windows ((T)OPTW), the Time Dependent Orienteering Problem (TDOP) and so on. Vansteenwegen et al. (2011) provide a comprehensive survey about the OP and its variants up to the year 2009. Gunawan et al. (2016) extended the survey by focusing on the most recent works of the OP and its variants.

One of the recent variants of the OP is the Orienteering Problem with Variable Profits (OPVP), as presented by Erdoğan and Laporte (2013). In the OPVP, a visit at a particular node can be extended in order to collect more scores. In order to replicate such a situation, Erdoğan and Laporte (2013) introduced discrete passes, where each pass on a particular node represents a constant time incurred. The more passes the visit made, the longer the duration of stay is.

In this paper, we introduce a new variant of the OPVP, namely the Team OP with Variable Profits (TOPVP). In the context of logistic applications, a path can be referred as a vehicle that needs to visit certain number of nodes and profits from nodes are considered as collected scores. The TOPVP extends the OPVP by considering multiple paths/vehicles. Therefore the total collected scores from all paths is the main objective of the TOPVP.

We introduce a mathematical model for the TOPVP. First, we solve the mathematical model by the AIMMS Outer Approximation algorithm (AOA) solver. Due to the limitation of solving large instances, we then propose a heuristic which is based on Iterated Local Search (ILS). We conclude that the proposed algorithm performs well with short computation times.

The remainder of this paper is as follows. In Section 2, a literature review of the OP including its variants is provided. The problem description including the mathematical model is detailed in Section 3. In Section 4, the proposed algorithms are described. Section 5 reports numerical experiments that were performed on benchmark instances. Finally, in Section 6, we summarize the main achievements and future works.

## 2. LITERATURE REVIEW

Tsiligirides (1984) first defined the standard Orienteering Problem (OP). Important assumptions include perfect knowledge over the score specified for each node and the time incurred for the edges. In addition, each node can only be visited once except the start and the end nodes which commonly refer to the same node. In this paper, we regard the start and end nodes as the same node.

One characteristic of the classical OP is that it is prohibited from staying at any node during a visit; the full profit is collected upon reaching the node. In other words, the time spent in a particular node is assumed to be zero. However, in certain situations especially related to logistics problems, the time spent in a particular node for a vehicle to unload the delivery has to be considered. This problem is referred to as the OPVP (Erdoğan & Laporte, 2013).

In the OPVP, the vehicle is permitted to prolong the duration of stay. In this case, each node is assigned with a profit that can potentially be collected where the actual collected amount depends on the time spent on the node. The vehicle is not compelled to collect the full profit. Erdoğan and Laporte (2013) introduce discrete passes to represent the vehicle's duration of stay at a particular node for the discrete model of OPVP. More specifically, making a pass means staying a predefined amount of time at a node. The amount of time spent on a node is additive according to the number of passes made. The profit collected over the duration of stay is described using growth/decay functions, dictating the rate of increase/decrease of profit collected per pass made. The rest of the conditions for OPVP remain identical to the OP. Hence, making multiple passes on a node does not equate to visiting the node multiple times since each node can only be visited once.

A unified branch-and-cut algorithm for OPVP was proposed as the solution approach, using adapted inequalities from the Covering Tour Problem (CTP) formulation (Gendreau, Laporte, & Semet, 1997). Since no prior research was done, there were no benchmark instances available for OPVP. As such, Erdoğan and Laporte (2013) modified the Travelling Salesman Problem (TSP) test instances from TSPLIB. Even though optimality was achieved for most of the test instances, excessive computation times were required for the larger instances.

Considering the limited literature available for the OPVP, reviewing solution approaches to TOP may provide deeper insights into the development of heuristics for TOPVP. This is because TOP and TOPVP share largely similar characteristics with the exception of the variable profits component.

Chao et al.'s (1996a) proposed a heuristic that involved neither searching techniques nor acceptance of infeasible intermediate solutions. Instead, Chao et al.'s heuristic involves two phases: initialization and improvement. In the initialization phase, a feasible solution is constructed using vertices that are furthest from the depot. Additional paths involving vertices not in the initial feasible solution are constructed in this phase as well. The improvement phase consists of iterating the sequence of two-point exchange, one-point movement and 2-Opt until terminating conditions are met.

Boussier, Feillet and Gendreau (2007) proposed a Branch & Price algorithm using column generation to solve the relaxed master problem and then using the branch-and-bound method to obtain an integer solution. Archetti, Hertz and Speranza (2007) proposed four comparable metaheuristics that are variants of the tabu search and variable neighbourhood search heuristics. The metaheuristics first generate an initial feasible solution using the initialization phase from Chao et al.'s (1996a) heuristic.

According to Vansteenwegen et al.'s survey (2011), Archetti et al.'s proposed metaheuristics are one of the leading algorithms for TOP in terms of achieving best known solution, average gap to best known solution and average computation time. In addition, Vansteenwegen et al. noted that high-performing algorithms are inclined to construct feasible paths for non-included vertices, allow infeasible solutions during the

search procedure as well as alternate between objective value increasing and travel time decreasing operators.

Considering that the solution approaches to TOP are well researched as well as able to produce high-performing and credible results, it is reasonable to adapt solution approaches to TOP for TOPVP. Notably, both Archetti et al.'s and Chao et al.'s heuristics involve elements of iterated local search (ILS) method, a method that can be easily implemented. Thus, it is possible to use the ILS method as a solution approach to TOPVP if the operators can be adapted to accommodate the variable profit component.

## 3. Team Orienteering Problem with Variable Profit

### 3.1 Problem Description

The TOPVP can be described on an undirected graph G = ($N$, $E$), where $N = \{0, 1, \ldots, n\}$ is the set of nodes and $E$ is the set of edges. Nodes 1 to $n$ are potential nodes to visit, whereas node 0 corresponds to the start and end nodes of the paths. Each node $i \in N$ is designed with a score $S_i$ as well as an associated collection parameter $\alpha_i \in [0,1]$. The amount of score collected at each node $i$ depends on the duration of stay at that node and its collection parameter $\alpha_i$. The duration of stay at nodes is represented by discrete passes. Each pass made at node $i$ incurs a constant time cost $r_i$. The collection parameter is used to model the decay of the collected score where each pass made at a node allows collecting 100 $\alpha_i$ percent of the remaining score.

A travel time $t_{ij}$ is associated with every edge $(i,j) \in E$. Thus, the total travel time on a particular path is contributed by the travel time across edges as well as the number of passes made at visited nodes. The objective of the TOPVP is to determine a set of paths $P$ such that the collected score by all paths is maximized. The amount of time required to traverse between two nodes $(i,j)$ is assumed to be symmetrical ($t_{ij} = t_{ji}$). In addition, the travel time associated with every edge satisfies the triangle inequality. Standard constraints applied to the OP (Vansteenwegen et al., 2011) are also applied in the TOPVP, such as each node can only be visited at most once except the start node which is the same with the end node, each path has to be started and ended at the start and end nodes, respectively, and each path is limited by the time budget $T$.

### 3.2 Mathematical Model

The formulation for the TOPVP is extended from the OPVP discrete model (Erdoğan and Laporte, 2013). The theoretical maximum number of passes at node $i$ is denoted as $m_i$ ($\leq \lfloor (L - 2t_{0i})/r_i \rfloor$). Below is the list of decision variables for the mathematical model.

Decision Variables:

$x_{ijp}$ = 1, if the vehicle traverses edge $(i,j) \in E$ on path $p$; 0, otherwise

$y_{ilp}$ = 1, if $l$ or more passes are performed on node $i$ on path $p$; 0, otherwise

$N_p$ = number of nodes visited by path $p$; including node 0

$$Maximize \quad \sum_{i \in V \setminus \{0\}} S_i \sum_{l \in \{1,\ldots,m_i\}} \alpha_i (1-\alpha_i)^{l-1} \; y_{ilp} \tag{1}$$

Subject to:

$$\sum_{j:(i,j) \in E, p \in P} x_{0jp} = \sum_{i:(i,j) \in E, p \in P} x_{i0p} = |P| \tag{2}$$

$$\sum_{p \in P} y_{i1p} \leq 1, \; (i \in V \setminus \{0\}) \tag{3}$$

$$\sum_{(i,k) \in E, k \neq 0} x_{ikp} = \sum_{(k,j) \in E, k \neq 0} x_{kjp} = y_{k1p}, \; (p \in P, i \neq j) \tag{4}$$

$$y_{ilp} \leq \; y_{i,l-1,p}, \; (i \in V \setminus \{0\}, l \in \{2, \ldots, m_i\}, p \in P) \tag{5}$$

$$y_{i(m_i+1)p} = 0, \; (i \in V \setminus \{0\}, p \in P) \tag{6}$$

$$\sum_{(i,j) \in E, i \neq j} t_{ij} x_{ijp} + \sum_{i \in V} r_i \sum_{l \in \{1,\ldots,m_i\}} y_{ilp} \leq T, \; (p \in P) \tag{7}$$

$$2 \leq u_{ip} \leq N_p, \; (i \in V \setminus \{0\}, p \in P) \tag{8}$$

$$u_{ip} - u_{jp} + 1 \leq (N_p - 1)(1 - x_{ijp}), \; (i,j \in V \setminus \{0\}, p \in P) \tag{9}$$

$$y_{01p} = 0, \; (p \in P) \tag{10}$$

$$y_{ilp} = 0 \; or \; 1, \; (i \in V \setminus \{0\}, l \in \{1, \ldots, m_i\}, p \in P) \tag{11}$$

$$x_{ijp} = 0 \; or \; 1, \; ((i,j) \in E, p \in P) \tag{12}$$

The objective function (1) is to maximize the total collected scores from visited nodes from all paths. It is worth noting that the total profit collected from each node will then be $S_i \sum_{l \in \{1,\ldots,m_i\}} \alpha_i (1-\alpha_i)^{l-1} \; y_{ilp}$ resembling a finite geometric series. Constraints (2) designate node 0 as the start and end nodes for each path. Constraints (3) ensure that across all paths, each node can only be visited at most once with the exception of node 0. Constraints (4) ensure the connectivity between the edges and the node. In other words, each node that is visited must be the origin and the destination for a pair of edges.

Constraints (5) ensure that in order to make further passes at a particular node, the preceding pass must be made. Constraints (6) ensure that the paths do not exceed the maximum allowable passes of the visited nodes. If the maximum allowable passes of all nodes are not limited by exogenous reasons, then constraints (6) can be relaxed. Constraints (7) ensure that the total time allocated does not exceed the time budget $T$ for every path. Since both edge costs and time incurred from making passes at nodes are subtracted from the total time allocated, costs and time are treated as synonymous in this paper. Constraints (8) and (9) prevent the forming of subtours. Constraints (10) ensure that no passes are made at node 0. Constraints (11) and (12) are integer constraints.

Erdoğan and Laporte (2013) noted that in the case where $a_i = 1$ for all $i \in V \setminus \{0\}$, the OPVP is reduced into a Selective Travelling Salesman Problem (STSP) (Laporte &

Martello, 1990). Since STSP is NP-Hard and is a special case of OPVP, by extension, TOPVP is NP-Hard. This implies that an exact solution algorithm might be beyond computational reach and that attempting to obtain a sub-optimal solution through heuristics will be more appropriate.

## 4. ALGORITHMS

We propose two different approaches to solve the TOPVP: 1) the AIMMS Outer Approximation algorithm (AOA) and 2) an Iterated Local Search algorithm. Each of these approaches will be explained in the following sub-sections.

### 4.1 The Outer Approximation algorithm

In order to solve the proposed mathematical model in Section 3.2, we first use a commercial solver, AIMMS. Since the mathematical model is considered as a Mixed Integer Non Linear Problem (MINLP) model, we therefore use the AIMMS Outer Approximation algorithm (AOA) which solves an alternating sequence of Non Linear Programming (NLP) and Mixed Integer Programming (MIP) models (Hunting, 2011).

The flow diagram for the AOA algorithm is given in Figure 1. The MINLP is first solved as a relaxed NLP and the linearization is then performed around the optimal solution. Next, the resulting linear constraints are added to the model.

We refer the new linear model as the master MIP problem. Subsequently, the master MIP problem is solved and the integer part of the resulting solution is temporarily fixed. The fixed integer part is then translated back to the relaxed NLP. The linearization and integer fixing process is repeated until the termination condition (e.g. iteration limit) is met. The in-built solvers that AIMMS used to solve the TOPVP are CONOPT 3.14V and CPLEX 12.6.2.

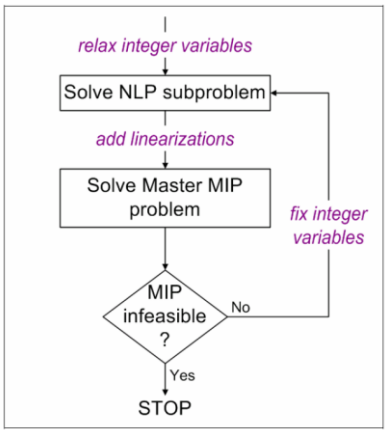


Figure 1: The Outer Approximation algorithm (Hunting, 2011).

### 4.2 Iterated Local Search

The limitation of the first algorithm is the capability to solve large instances within short computation times. We then propose the second algorithm which is mainly based on ILS, as shown in Figure 2 (Chao et al. 1996a). In this sub-section, an overview of the algorithm structure will first be presented followed by more detailed elaboration of the different operators used in the algorithm.

Step 1. Initialization
  Perform initialization
  Set *record* = objective value of the initial solution
  Set *deviation* = 5% × *record*
Step 2. 1st Improvement Phase
  For $k$ = 1, 2, 3, … ($K$ loop)
    For $l$ = 1, 2, 3, … ($L$ loop)
      Perform two-point exchange
      Perform one-point movement
      Perform 2-opt
      If no adjustment has been made, end $L$ loop
      If a solution with higher objective value is obtained, then
        Set *record* = objective value of new solution
        Set *deviation* = 5% × *record*
    End $L$ loop
    If *record* is not updated for 5 consecutive iterations, then
      Go to Step 3
    Else
      Perform re-initialization 1 (free $k$ vertices)
  End $K$ loop
Step 3. Re-initialization 2
  Perform re-initialization 2 (free $k$ vertices, $k$ is the stopping value in $K$ loop)
  Set *deviation* = 2.5% × *record*
Step 4. 2nd Improvement Phase
  Same sequence as Step 2 except *deviation* is now set at 2.5% × *record*

Figure 2: Iterated Local Search.

#### 4.2.1 Overview of the Algorithm Structure

In short, the proposed algorithm for TOPVP follows the basic principle of an iterated local search (ILS) method. The ILS is defined as a local search method that iteratively applies local search to perturbations of the current locally optimal solution (Stützle, 2006). The four basic requirements of the ILS are 1) an initial solution, 2) a perturbation guideline to deconstruct the locally optimal solution, 3) a local search to seek improvements in the solution and 4) an acceptance criterion to determine from which solution the local search is continued.

Similar to Chao et al.'s (1996a) algorithm, the proposed ILS consists of an initialization, two improvement and two re-initialization steps. The initialization step constructs the initial feasible solution. The two improvement steps represent the local search method, seeking possible improvements in the current solution. The two re-initialization steps perturb the locally optimal solution for the next iteration of improvement. The acceptance criterion used in ILS is based on an optimization algorithm called Record-to-Record Travel (RRT) (Dueck, 1993).

The RRT is used in ILS to determine if a new configuration of the solution should be accepted. In the RRT,

the best solution obtained thus far is set as the *record.* Any configuration found that is an improvement over *record* is set as the new *record*. The proposed ILS attempts to seek further improvement using the new configuration. In addition, a constant percentage of the *record* is set as the acceptance threshold called *deviation*. Whenever the algorithm fails to find a configuration that is an improvement, the best configuration that deteriorates the solution within the *deviation* will be chosen to be worked on.

The sequence of events to be executed for ILS is as follows:

**Step 1 (Initialization Phase)**: An initial solution is constructed using the initialization process which is discussed in Section 4.2.2. The objective value of the initial solution is set as the *record* while the *deviation* is set at 5% of the *record*.

**Step 2 (Improvement Phase):** The improvement phase consists of two loops; the inner loop will be referred to as *I* loop while the outer loop will be referred to as *K* loop. In *I* loop, a local search for improvement is conducted. The local search is a sequence of operators consisting of two-point exchange, one-point movement followed by 2-opt. All operators would be discussed in Section 4.2.3. At the end of the sequence, if a solution with higher objective value is obtained, then *record* and *deviation* are updated. The local search is iterated until no exchanges or movement of vertices are performed, ending *I* loop.
Note that it is possible for exchanges or movement that result in no improvement in objective value. In *K* loop, re-initialization 1 (Section 4.2.4) is performed to perturb the solution obtained from *I* loop. The new solution will then be iterated through *I* loop again. Subsequent re-initialization 1 perturbs the solution according to how many times re-initialization 1 was performed. A global variable *k* is used to track the number of re-initialization 1 steps performed. The terminating condition for *K* loop is when no new *record* is achieved for 5 consecutive iterations.

**Step 3 (Re-initialization 2):** Perform re-initialization 2 (Section 4.2.4) using the last *k* value of Step 2. The *deviation* is also set to 2.5% of the current *record.*

**Step 4 (Improvement Phase 2):** The 2nd improvement phase follows the same sequence of events as Step 2 (1st improvement phase) with the exception of the *deviation* used in RRT exchanges and movements.

### 4.2.2 Initialization Phase

In the initialization phase, we first remove nodes which cannot be theoretically visited given the time budget. Any vertex $i$ for which $2t_{i0} + r_i > T$ is removed. This is because the vehicles have to return to the depot and any vehicle that visits these nodes will always violate the time budget allocated. The remaining nodes are classified as feasible nodes.

The next process is to construct $|P|$ paths. The main idea is to assign as many nodes as possible without violating the time budget (Chao et al., 1996 and Archetti et al., 2007). Due to the discrete passes, it is required to decide between increasing the passes made on nodes currently in a particular path of interest or inserting an additional vertex that is not in that path.

The $|P|$ paths with the highest profit collected constitute the solution and will be referred to as the set of paths $P_{TOPVP}$. Thus, the sum of the profit collected from each path in $P_{TOPVP}$ is the objective value. The set of remaining paths will be referred to as $P_{NTOPVP}$.

### 4.2.3 Improvement Phase

Using the initial solution generated in the initialization phase, we then improve the solution by performing three different operators of Iterated Local Search.

*Two-point exchange*
The objective of the two-point exchange is to seek possible improvement in the solution by exchanging nodes from the paths in $P_{TOPVP}$ with nodes from the paths in $P_{NTOPVP}$.

*One-point movement*
One-point movement is the operator used after two-point exchange in the local search component. One-point movement attempts to improve the solution by relocating nodes from one path to another. In particular, every feasible node is checked for possible movement one at a time. The candidate node is inserted into the designated path using the cheapest insertion heuristic.

*2-opt*
The 2-opt technique is used to reduce the total edge cost incurred by the paths in $P_{TOPVP}$ and $P_{NTOPVP}$. By doing so, there may be opportunities for more exchanges and movements in later iterations. There should be no improvement in the objective value due to 2-opt.

Note that as mentioned earlier in the overview of the structure of our proposed algorithm, two parameters, *record* and *deviation,* are updated only after a sequence of two-point exchange, one-point movement and 2-opt is completed. If no new *record* is found after 5 consecutive iterations, then the terminating condition for the corresponding improvement phase has been achieved. Otherwise, the solution obtained will be perturbed using re-initialization 1.

### 4.2.4 Re-initialization Phase

*Re-initialization 1*

To avoid being stuck in a local optimum, re-initialization 1 is used to prepare the solution for the next iteration of local search. In this phase, nodes with the lowest collected scores are removed from all paths in $P_{TOPVP}$. The number of nodes removed from each path is determined by a variable $k$. As the iteration count for the local search increases, the value of $k$ is increased. In other words, more vertices are removed from each path in $P_{TOPVP}$ in subsequent re-initialization 1 steps.

*Re-initialization 2*

In re-initialization 2, the nodes are removed differently from re-initialization 1. Instead of removing nodes with the lowest collected score, nodes with the smallest ratio of collected score to insertion cost are removed from each path in $P_{TOPVP}$. Note that throughout the TOPVP heuristic, re-initialization 2 will only be performed once.

## 5. COMPUTATIONAL RESULTS

Since there are no benchmark instances for the TOPVP, we adopt the same scheme proposed by Erdoğan and Laporte (2013) to generate benchmark OPVP instances using selected TSP test instances. Those instances are kroA100, kroB100, kroC100, kroA200 and kroB200 which are obtained from the TSPLIB. The OPVP test instances, generated from TSP test instances, are then used to generate the TOPVP test instances. This is done by taking the 1-vehicle OPVP and dividing the time budget by the number of vehicles/paths in TOPVP (Chao et al., 1996).

In order to verify and justify the performance of ILS, we compare the results obtained by the AIMMS AOA and ILS for solving small benchmark instances. The parameters varied are number of nodes $|N|$, number of paths/vehicles $|P|$ and time budget for each path $T$. Since the test instances from TSPLIB are in sets of 100 vertices (kroA100, kroB100 and kroC100) and 200 vertices (kroA200 and kroB200), solving for optimality using the entire set of benchmark instances will be beyond computational reach. As such, to reduce the size of the experiment, only the initial 15 vertices from each instance were used, and experiments for 5, 10 and 15 vertices were conducted. In addition, the number of paths $|P|$ tested for each test instance was 1, 2 and 3.

Table 1 summarizes some results obtained. Due to limited space, we are not able to show the entire set of results. The gaps between ILS and AIMMS solutions for all instances range from 0% to 3.32%. As such, further experimentation to evaluate the TOPVP heuristic for larger test instances is a plausible notion.

With regards to the time budget designated for each path, arbitrary values were picked for the verification and validation experiments. To illustrate, the edge costs calculated from the benchmark instances can range up to around 3000 time units. For experiments with only 5 nodes or 5000 time units allocated for each path, the optimality gap is likely to be 0%. This can be attributed to the elimination of nodes that require more than 2500 time units when visiting from the start node. As such, the reduction in the number of nodes considered resulted in the TOPVP heuristic to more likely converge to the optimal solution.

On the contrary, for larger experiments with more than 7000 time units allocated, every node can be visited. In this case, the performance of ILS drops slightly, being able to achieve optimality in some experiments only. This is because for a given larger time budget, the paths in the solution will be longer and the maximum allowable passes made at each path will be higher. As observed, the computation times required for some larger instances have already exceeded 300000 seconds. Thus, setting larger time budget is likely to be beyond computational reach.

We continue conducting experiments for the full range of nodes using the benchmark instances, kroA100, kroB100, kroC100, kroA200 and kroB200. Since AIMMS is unable to solve the problem optimally after 2 hours of computational time, we decide to relax the integer requirements of the decision variables and treat them as the upper bound values of the solutions.

Table 1: KroA100 experimental results

| Instance | $\|P\|$ | $\|V\|$ | $T$ | AIMMS Obj Value | ILS Obj Value | Optimality Gap (%) | AIMMS CPU Time (s) |
|---|---|---|---|---|---|---|---|
| kroA100 | 1 | 5 | 5000 | 81.00 | 81.00 | 0.000 | 0.26 |
| | 1 | 5 | 7500 | 173.07 | 169.13 | 2.277 | 150.84 |
| | 1 | 10 | 5000 | 251.99 | 250.50 | 0.591 | 3.84 |
| | 1 | 10 | 7500 | 294.19 | 293.07 | 0.381 | 13.66 |
| | 1 | 15 | 5000 | 249.90 | 249.35 | 0.220 | 54.91 |
| | 1 | 15 | 7500 | 334.64 | 330.11 | 1.354 | 592.26 |
| | 2 | 5 | 9000 | 279.82 | 279.77 | 0.018 | 2.86 |
| | 2 | 10 | 9000 | 541.10 | 538.50 | 0.481 | 796.95 |
| | 2 | 15 | 9000 | 715.09 | 691.35 | 3.320 | 19634.19 |
| | 3 | 5 | 9000 | 280.00 | 280.00 | 0.000 | 2.70 |
| | 3 | 10 | 9000 | 553.80 | 553.17 | 0.114 | 2604.34 |
| | 3 | 15 | 9000 | 788.15 | 779.49 | 1.099 | 346925.58 |

Table 2: KroA100 and KroA200 experimental results

| Instance | $\|P\|$ | $\|V\|$ | $T$ | AIMMS Relaxed Solution | ILS Obj Value | TOPVP CPU Time (s) | Upper Bound-TOPVP Gap (%) |
|---|---|---|---|---|---|---|---|
| kroA100 | 2 | 25 | 7020 | 889.06 | 736.49 | 0.66 | 23.24 |
| | 2 | 50 | 7186 | 1842.05 | 1205.71 | 1.50 | 38.73 |
| | 2 | 75 | 7240 | 2453.64 | 1372.54 | 1.75 | 49.82 |
| | 2 | 100 | 7351 | 2873.86 | 1539.49 | 4.51 | 50.98 |
| | 3 | 25 | 4680 | 914.01 | 667.23 | 0.86 | 29.99 |
| | 3 | 50 | 4791 | 1486.12 | 967.35 | 1.12 | 42.14 |
| | 3 | 75 | 4827 | 2060.14 | 1174.92 | 2.07 | 47.30 |
| | 3 | 100 | 4901 | 2671.92 | 1318.98 | 2.17 | 58.43 |
| | 4 | 25 | 3510 | 566.24 | 580.05 | 0.74 | 0.03 |
| | 4 | 50 | 3593 | 1148.19 | 757.50 | 0.64 | 41.47 |
| | 4 | 75 | 3620 | 1648.31 | 956.83 | 1.77 | 50.40 |
| | 4 | 100 | 3673 | 2063.15 | 1083.80 | 2.64 | 53.83 |
| kroA200 | 2 | 125 | 7345 | 3332.03 | 1755.72 | 3.90 | 53.92 |
| | 2 | 150 | 7380 | 3634.92 | 1914.66 | 7.17 | 57.49 |
| | 2 | 175 | 7380 | 4133.91 | 2036.53 | 5.21 | 61.01 |
| | 2 | 200 | 7380 | 4144.51 | 2111.68 | 8.94 | 60.62 |
| | 3 | 125 | 4897 | 2864.98 | 1687.18 | 3.34 | 50.22 |
| | 3 | 150 | 4920 | 3160.60 | 1722.68 | 4.77 | 55.39 |
| | 3 | 175 | 4920 | 3927.29 | 1860.15 | 4.89 | 61.20 |
| | 3 | 200 | 4920 | 3376.31 | 1917.95 | 4.97 | 59.64 |
| | 4 | 125 | 3673 | 2422.35 | 1405.30 | 2.17 | 50.46 |
| | 4 | 150 | 3690 | 2564.64 | 1498.14 | 2.94 | 50.01 |
| | 4 | 175 | 3690 | 2778.86 | 1595.99 | 3.18 | 51.90 |
| | 4 | 200 | 3690 | 3498.26 | 1630.20 | 3.70 | 53.40 |

Some results for the kroA100 instance are presented in Table 1. The number of nodes were varied at 25, 50, 75 and 100. The results for a larger instance, kroA200, are also presented in Table 2. Experiments were conducted for 2-path, 3-path and 4-path TOPVPs.

As observed from Table 2, ILS is able to obtain a solution using considerably small computation time, even for the large instances kroA200 and kroB200. The maximum computation time recorded was 11.95 seconds for 2-path 175 nodes TOPVP using test instance kroB200.

Although the gap between the solution obtained from ILS and the upper bound is large, it can be reasonably justified. Given the termination time being set at 7200 seconds, AIMMS is unable to achieve the terminating condition for the AOA algorithm. Thus, the upper bounds as well as the solutions obtained from AIMMS are still relaxed and may not be feasible. Hence, the actual optimality gap is expected to be smaller than the gap reported in Table 2.

ILS managed to produce near-optimal solution for a few of the experiments. More specifically, for kroA100 4-path as well as kroB100 3-path and 4-path, the gaps for the experiments using only 25 nodes are considerably small, ranging from 0.03% to 5.59%. The reason for this is twofold; 1) the number of nodes is only 25 and 2) the time budgets allocated for the 3-path and 4-path experiments are relatively low. As such, the number of feasible nodes in the time budget constraint is small, enabling ILS to reach a near-optimal solution quickly.

On the contrary, for experiments with more nodes, ILS is unable to achieve similar small gaps. By the same argument, the number of feasible nodes for ILS to consider is large which explains the much larger gap reported.

## 6. CONCLUSION

We introduce a variant of the Orienteering Problem, namely the Team Orienteering Problem with Variable Profit (TOPVP). In TOPVP, multiple paths are involved in collecting scores which are dependent on the time spent at the nodes. In this paper, the TOPVP is formulated as a Mixed Integer Non-Linear Programming mathematical model. We also developed an Iterated Local Search algorithm for solving the TOPVP.

We then compare the results obtained from solving some modified benchmark instances by ILS with that obtained by the AIMMS Outer Approximation algorithm. The AOA is only able to solve small instances. For the small benchmark instances ranging up to 15 nodes, ILS is able to achieve optimality in several experiments using considerably short computation time. ILS is then applied to larger instances ranging up to 200 nodes. While the computation time for ILS is low, the gap between the ILS and the upper bound obtained from AIMMS after 2 hours is still considerably large. This could be due to the simplicity of the operators used in ILS. As such, the development of a more effective heuristic incorporating more advanced operators to achieve a smaller optimality gap is a possible direction of future research. Finally, we plan to implement the algorithm to solve some applications of the OP, such as the vehicle routing problem and the tourist trip design problem.

## REFERENCES


Archetti, C., Hertz, A., and Speranza, M.G. (2007) Metaheuristics for the team orienteering problem. *Journal of Heuristics*, **13**, 49-76.

Boussier, S., Feillet, D., and Gendreau, M.L. (2007) An exact algorithm for team orienteering problems. *4OR*, **5**, 211-230.

Chao, I., Golden, B., and Wasil, E. (1996a) A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, **88**, 475-489.

Chao, I., Golden, B., and Wasil, E. (1996b). The team orienteering problem. *European Journal of Operational Research*, **88**, 464-474.

Dueck, G. (1993) New optimization heuristics; the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, **104**, 86-92.

Erdoğan, G., and Laporte, G. (2013) The orienteering problem with variable profits. *Networks*, **61**, 104-116.

Gunawan, A., Lau, H.C., and Vansteenwegen, P. (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, **255**, 315-332.

Gendreau, M., Laporte, G., and Semet, F. (1997). The covering tour problem. *Operations Research*, **45**, 568-576.

Harvey, W.D., and Ginsberg, M.L. (1995) Limited discrepancy search. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 607-615.

Heidelberg University. (n.d.). *TSPLIB*. Retrieved from Discrete and Combinatorial Optimization: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html

Hunting, M. (2011) *The AIMMS Outer Aproximation Algorithm for MINLP*. Advanced Interactive Multidimensional Modeling System.

Laporte, G., and Martello, S. (1990) The selective travelling salesman problem. *Discrete Applied Mathematics*, **26**, 193-207.