Orienteering Problem with Hotel Selection: A Variable Neighborhood Search Method

A. Divsalar[1], P. Vansteenwegen[2], and D. Cattrysse[1]

[1]K.U.Leuven, Centre for Industrial Management Traffic and Infrastructure, ali.divsalar@cib.kuleuven.be

[2]Ghent University, Department of Industrial Management,

Abstract

In this research, we developed a skewed variable neighbourhood search algorithm to solve the orienteering problem (OP) with hotel selection, a non-investigated variant of the OP. We also designed two appropriate sets of benchmark instances with known optimal solutions. Applying the proposed algorithm on these instances shows the quality of the algorithm. The algorithm is also fast enough to be implemented in a tourist application.

Keywords: orienteering problem, hotel selection, variable neighbourhood search

Imagine a tourist who is planning to visit a certain region with various attractions. The tourist wants to select the combination of attractions that maximizes his pleasure. The visit will last several days and only the initial departure and the final arrival location are fixed. The departure and arrival locations (hotels) during the visit should be selected in an optimal way based on the attractions that are selected. Obviously, the hotel in which the tourist ends a given day has to be the same as the starting hotel of the next day. The hotels can be selected from all suitable hotels available in the region. This example can be modeled as a new variant of the orienteering problem that we call the orienteering problem with hotel selection (OPHS). This problem is defined in detail in [1].

Here we focus on an algorithm to solve the OPHS. The algorithm combines different moves using a modified variable neighborhood search (VNS) framework. This is called Skewed VNS [2](SVNS) and it is a variant of VNS that is often used for problems which have several separated and possibly far apart hills containing near-optimal solutions. This is the case for all variants of the OP and SVNS was used successfully before on team OP (TOP) instances[3].

The algorithm consists of two main parts: initialization and improvement. The initialization starts by heuristically solving a sub-OP between every pairs of hotels. It is important to note that these sub-OPs are solved by a very simple local search method which makes this step very fast. Then, all the feasible combinations of hotels for the OPHS are determined. Afterwards, using the sub-OP solutions, a heuristic upper bound (HUB) is calculated for each determined combination. This HUB is considered in order to sort all the possible combinations of hotels. To make the initial solution, the feasible combination of hotels with the highest HUB is selected and between each pair of hotels an OP is solved, taking into account which vertices are already selected.

After preparing the initial solution, the improvement phase with vertices-update moves and hotel-update moves is started.

In the literature, many local search moves are applied to different variants of OP[4]. Because of the similarities between the OPHS and the TOP, the most successful local search moves for the TOP [4] are selected, modified and applied as vertices-update moves for the OPHS. These local search moves are Insert, Replacement, Two-Opt, Swap-best, swap-trips, and Remove-insert. Vertices-update moves are used to improve the solution until no more improvements are possible.

When a local optimum is reached, three different hotel-update moves are used to shake the solution. All three moves start by replacing the hotels in the current solution by another combination of hotels. This combination is selected randomly out of the 50 feasible combinations of hotels with the highest HUB. The hotel-update moves differ from each other in the way the current selection of vertices is combined with the new selection of hotels.

In the re-center part of the algorithm, when the new solution is better or even when it is slightly worse than the solution before shaking, it is still considered as the new solution to start the next iteration from. This helps to diversify the search and explore other parts of the solution space.

Two sets of benchmark instances including 75 and 50 instances respectively are made to examine the algorithm. The average gap with the known optimal solution and the CPU time for SET 1 are 1.7 % and 0.15 s; for SET2 2.31 % and 0.13 s.

References:

[1] A. Divsalar, P. Vansteenwegen, and D. Cattrysse, Orienteering Problem with Hotel Selection 25th Annual Conference of the Belgian Operations Research Society (ORBEL), (2011) February 10-11, Ghent, Belgium, p.85-86.

[2] P. Hansen and N. Mladenovi, Variable neighborhood search: Principles and applications European Journal of Operational Research, vol. 130, no. 3, pp. 449-467, May 2001

[3] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, Metaheuristics for Tourist Trip Planning Lecture Notes in Economics and Mathematical Systems, vol. 624, pp. 15-31, 2009.

[4] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, The orienteering problem: A survey European Journal of Operational Research, vol. 209, no. 1, pp. 1-10, Feb. 2011.