2
Inspection Route Optimization
Problem proposed by The Co-operators

Bernard Gendron, Thibaut Vidal, Steven Lamontagne, Belhal Karimi, David Hagenimana, Jalal Ahammad, Monuara Gagum, Dena Kazerani, Ilya Chugunov, Maikel Geagea, Mickael Albertus, Bora Yongacoglu, Mathieu Giguère, and Jean-Michel Plante

2.1 Introduction

This report is a summary of the work carried out during the Eighth Montreal Industrial Problem Solving Workshop, held at the Centre de Recherches Mathématiques. Our team was supervised by Bernard Gendron and Thibaut Vidal. The student participants for this group were Steven Lamontagne, Belhal Karimi, David Hagenimana, Jalal Ahammad, Monuara Gagum, Dena Kazerani, Ilya Chugunov, Maikel Geagea, Mickael Albertus, and Bora Yongacoglu. We were assisted by the representatives from the company, Mathieu Giguère and Jean-Michel Plante.

The Co-operators is a Canadian insurance and financial services company providing personal and casualty, life, and investment insurance to individuals as well as to companies. As part of their commercial insurance services, they send inspectors to visit buildings in order to evaluate or reevaluate the risk associated with the business. The Co- operators insure over 5000 buildings, and with comparatively few inspectors they must choose which buildings to visit during a given year. Each location is assigned a risk score between 0 and 100

Bernard Gendron
CIRRELT & Université de Montréal

Thibaut Vidal
Pontificia Universidade Catòlica do Rio de Janeiro

Steven Lamontagne
Université de Montréal

Belhal Karimi · Dena Kazerani
École Polytechnique (France)

David Hagenimana
University of Nairobi

Jalal Ahammad · Monuara Gagum
Memorial University of Newfoundland

Ilya Chugunov
UC Berkeley

Maikel Geagea
Polytechnique Montréal

Mickael Albertus
Université Paul Sabatier Toulouse

Bora Yongacoglu
Queen's University

Mathieu Giguère · Jean-Michel Plante
The Co-operators

(where 100 stands for the highest risk): these scores are used for helping choose the buildings that will be visited. Finally the following constraints must be respected when constructing the inspections schedule.

(1) Each inspector works a specified number of weeks during the year, which depends upon the inspector.
(2) Inspectors must return home for weekends.
(3) There are mandatory buildings (not necessarily high-risk ones), i.e., buildings that must be included in the inspections schedule.

In order to solve the problem, the analysts at The Co-operators use two methods. The first method consists of assigning each inspector to a territory (subdivided into provinces) based on the location of the inspector's home. Within each territory, the buildings are then chosen in descending order of risk score.

The second approach used by the company is based on a decomposition similar to that used in the first method. Instead of assigning *provinces* to specific inspectors, the second approach assigns *cities* to inspectors. The locations (i.e., buildings) are then assigned to cities by minimizing the distance between building and city. The company then solves separate optimization problems for each of the cities, with the objectives being to maximize the total risk score in the selected schedule and to minimize the total distance travelled by the inspector. The resulting minimization problem is a kind of modified traveling salesman problem, in which a single person completes a *tour* of the locations (i.e., the inspector visits each location once and only once) and the total distance travelled is minimal.

Since there are two objective functions, we can picture the set of "best" solutions as a curve of combinations of total risk score and distance travelled. Hence bounding one of the two objective functions allows us to optimize the other objective. While the results of this approach are better than those obtained with the previous approach, it has two flaws. First, the decomposition into cities is carried out arbitrarily and restricts the locations to a single region. Second, while there is a constraint on the *distance*, the real constraint should involve the *time*. The distance between the locations can be used to estimate the travel time, but it does not take into account the time it takes to inspect the locations.

The first step undertaken by the team during the workshop was to provide a mathematical formulation of the problem. The objective is simple: to maximize the total risk score of the buildings visited during a given year. Rather than having two objective functions, the objective function corresponding to the distance travelled is replaced by time constraints:

- Daily constraints: every inspector makes a tour starting and ending at a *hotel* (here, and elsewhere in the document, "hotel" refers to any place where the inspector may spend the night, be it a hotel, a motel, his home, etc.). The total duration of the tour (travel time + inspection time) may not exceed 7 hours;
- Weekly constraints: every inspector works up to 5 days (i.e., makes up to 5 tours) before going back to his home;
- Yearly constraints: every inspector works (about) 46 weeks in any given year.

The natural decomposition of constraints leads to a decomposition of the problem. First, we can construct a separate optimization problem for each week. The solutions of the weekly problems are then "merged" into a solution for the annual problem. To achieve these tasks, the team of students was split into three sub-teams, whose tasks were (respectively):

1. To extract and analyze the data provided by the company;
2. To develop and implement models for the annual problem and for the weekly problem;
3. To develop and implement heuristic methods for the weekly problem.

2.2 Data Extraction

The representatives from The Co-operators provided the team with data from the company. A data file included, for each of the 5000 prospective locations to be visited, the distance to each of the other locations as well as the distance to each of the possible hotels (this information was stored in a matrix where the

(i, j) component indicates the distance from building i to building j). The data file also included the time necessary to inspect a particular building, the information that it was mandatory to visit the building (or not), and the risk score associated with it. In order to implement the time-related constraints for the weekly problem, the first task of the data extraction team was to convert the distances stored in the data file into values for elapsed time. Because of privacy concerns, the team could not access the exact location of each building: therefore the distances were converted uniformly assuming a travelling speed of 60 km/h. The next task of the data extraction team was to reduce the size of the problem by eliminating some buildings and assigning the locations to potential cities (each "city" being a potential hotel). For instance, if the time to travel from location i to hotel j was at least 3.5 hours, then it could not possibly be advantageous to include location i within the city defined by hotel j.

2.3 Annual Problem

2.3.1 *Annual Problem: Assumptions and Data*

We use the following notation for the annual problem.

Notation	
Symbol	**Meaning**
I	Set of weekly solutions
J	Set of locations
K	Set of inspectors
M	Set of mandatory locations
δ_{ij}	equals 1 if location $j \in J$ belongs to weekly solution $i \in I$, 0 otherwise
θ_{ik}	equals 1 if inspector $k \in K$ is assigned to weekly solution $i \in I$, 0 otherwise
p_i	Profit (total risk score) of weekly solution $i \in I$
W_k	Annual number of weeks worked by inspector $k \in K$

J, K, M, and W are provided by the company, whereas I, δ, θ, and p are to be constructed as part of the solution method.

2.3.2 *Annual Problem: Model*

We define the variable x_i as follows: $x_i = 1$ if weekly plan $i \in I$ is selected and $x_i = 0$ otherwise.

$$\max \sum_{i \in I} p_i x_i \tag{2.1}$$

$$\sum_{i \in I} \delta_{ij} x_i \leq 1 \quad j \in J \tag{2.2}$$

$$\sum_{i \in I} \theta_{ik} x_i \leq W_k \quad k \in K \tag{2.3}$$

$$\sum_{i \in I} \delta_{ij} x_i \geq 1 \quad j \in M \tag{2.4}$$

$$x_i \in \{0, 1\} \quad i \in I \tag{2.5}$$

The objective (2.1) is to maximize the total risk score of buildings visited during the year. Inequalities (2.2) enforce the constraint that each building may be visited once. Constraints (2.3) enforce a limit on the number of weeks during which an inspector works (in the year). Constraints (2.4) ensure that all mandatory buildings are visited. Since the mandatory buildings do not necessarily have high risk scores, there is no guarantee that they would be visited if these constraints were not included into the model.

The objective (2.1), together with constraints (2.2) and (2.5), is a well-known NP-hard problem, referred to as the *set packing problem* (see for instance [2]). If additional constraints (such as (2.3) and (2.4)) are added, the problem is referred to as a *constrained set packing problem.*

2.3.3 *Annual Problem: Solution Methods*

There are exact methods, called *branch-and-price* methods, for solving this kind of problem: they involve a combination of *column generation* and *branch-and-bound* methods. An efficient implementation of these methods, however, requires a significant programming effort. Heuristic methods generating high-quality (though not always optimal) results can be developed fairly quickly.

The method used to obtain a solution for the annual problem is based on a simple principle: assume that we have a set with every possible *weekly schedule* (a weekly schedule being a set of 5 daily tours assigned to a single inspector). The task of finding the optimal yearly schedule is then to select the optimal subset from our collection. In practice, finding the set of all possible weekly schedules and selecting the best combination of weekly schedules is computationally intractable. During the workshop the team goal was to construct a set of weekly schedules (see Sections 4 and 5) and select an optimal subset among them (a similar approach is described in [3], p. 330).

From the team working on the weekly solutions, the team working on the annual problem would receive a file containing a set of weekly schedules. Each weekly schedule i in that file is a list that includes:

- $k \in K$ with $\theta_{ik} = 1$ (i.e., the inspector assigned to that weekly schedule);
- p_i, i.e., the total profit for that weekly schedule;
- the set of $j \in J$ with $\delta_{ij} = 1$ (i.e., the set of locations comprised in that weekly schedule).

The list of weekly schedules is denoted by I. The constrained set packing problem (2.1)–(2.5) was implemented in Python; the *branch-and-bound* method in the Gurobi solver was used to find the optimal set of weekly solutions. Over the course of the workshop, the model was tested using schedules randomly generated by a Matlab script. This was carried out in order to validate the model, as well as to observe computational times for varying numbers of weekly schedules. Because of feasibility errors with randomly generated data, a second script was developed that weakened constraints (2.2) at the cost of an increase in computing time. Thanks to the tests, it was found that the computing times for models involving more than 10,000 weekly schedules were very large (indeed some models could not be solved). The final version of the script was thus based on that number of weekly schedules, with the possibility of developing later a script that could solve instances with more than 10,000 weekly schedules by subdividing the list of schedules into subgroups of 10,000 of fewer.

2.4 Model for the Weekly Problem

2.4.1 *Formulation of the Weekly Problem*

The goal of the weekly problem is to create a set of $d \leq 5$ daily tours such that the overall profit (risk score) is maximized and the daily time limit (of 7 hours) is respected. This is a variant of the *vehicle routing problem* (VRP) known as the *team orienteering problem* (TOP), named after a game in which teams of players must bring flags of varying worth back to the starting point within a fixed time limit (see [4], p.3). Once a TOP is solved (through mathematical programming or by using heuristic methods), the schedule data is processed by completing the week (i.e., including $5 - d$ more days into it) and then assigning each weekly schedule to nearby inspectors. Since there is no requirement (at this stage) that each weekly schedule has a unique inspector (this condition is subsequently enforced by constraints (2.3) in the annual problem), weekly schedules can be generated quickly by simply changing which (feasible) inspector is assigned to it.

2.4.2 *TOP: Assumptions and Data*

We wish to construct d tours for each *city*. We use the following notation for this problem.

Notation	
Symbol	**Meaning**
$1, \ldots, n$	Locations that can be visited
$0,\ n+1$	Hotel (duplicated)
V	$\{0, 1, \ldots, n, n+1\}$, the set of nodes
A	$\{(i,j) \mid i \in V^{n+1}, j \in V^0\}$, the set of arcs, where $V^i = V \setminus \{i\}$
G	(V, A), the resulting directed graph
p_i	Profit at location $i \in V$ ($p_0 = p_{n+1} = 0$)
t_{ij}	Time for arc $(i,j) \in A$, including travel time from i to j and time to inspect j ($t_{0,n+1} = 0$)
T	Daily time limit (currently 7 hours, but could vary based on the day and the inspector)

2.4.3 *TOP: Model*

We introduce the following variables: y_i^l, where $y_i^l = 1$ if $i \in V$ is selected in tour l and $y_i^l = 0$ otherwise, and x_{ij}^l, where $x_{ij}^l = 1$ if $(i,j) \in A$ is selected in tour l and $x_{ij}^l = 0$ otherwise.

$$\max \sum_{i \in V} \sum_{l=1}^{d} p_i y_i^l \tag{2.6}$$

$$\sum_{j \in V^i} x_{ij}^l = y_i^l \quad i \in V^{n+1}, l = 1, \ldots, d \tag{2.7}$$

$$\sum_{j \in V^i} x_{ji}^l = y_i^l \quad i \in V^0, l = 1, \ldots, d \tag{2.8}$$

$$\sum_{l=1}^{d} y_i^l \le 1 \quad i \in V \setminus \{0, n+1\} \tag{2.9}$$

$$y_0^l = y_{n+1}^l = 1 \quad l = 1, \ldots, d \tag{2.10}$$

$$\sum_{(i,j) \in A} t_{ij} x_{ij}^l \le T \quad l = 1, \ldots, d \tag{2.11}$$

$$x_{0i}^l \le x_{0j}^{l+1} \quad i \in V^0, j \in V^0, i \le j, l = 1, \ldots, d-1 \tag{2.12}$$

$$y_i^l \in \{0,1\} \quad i \in V, l = 1, \ldots, d, \quad x_{ij}^l \in \{0,1\} \quad (i,j) \in A, l = 1, \ldots, d \tag{2.13}$$

The objective (2.6) is to maximize the total risk score visited during the week. Equations (2.7) and (2.8) refer to the condition that each location may only be visited once. In particular, equations (2.7) mean that, for every tour l, the number of arcs *leaving* node i is equal to 1 if node i is visited and 0 otherwise. Similarly, equations (2.8) mean that, for every tour l, the number of arcs *entering* node i is equal to 1 if node i is visited and 0 otherwise. Inequalities (2.9) enforce the condition that every location (other than the hotel) may only be visited once throughout the week. Constraints (2.10) enforce the condition that the hotel must be included at the beginning and end of the tour. Constraints (2.11) mean that, for each tour l, the daily time limit must be respected.

Inequalities (2.12) are a kind of *symmetry-breaking constraints*. They are not required to model the problem but they speed up the solution method considerably. These inequalities ensure that the first building visited in the daily tour l has an index smaller than or equal to the first building in the daily tour $l+1$. Without the symmetry-breaking constraints, a given set of daily tours would be represented by different solutions, corresponding to the permutations of the daily tour indices (for example, two solutions could be generated by transposing the tour of the first day and the tour of the second day).

Model (2.6)–(2.13) is still not sufficient to obtain solutions that have the proper form. Figure 2.1 displays a solution that could be obtained with the model for $d = 1$. That solution contains two *subtours*, one that starts and ends at the hotel, as required by constraints (2.10), and another that does not contain the hotel. No constraints currently forbid such subtours. Therefore we need to introduce *subtour elimination constraints* (SEC). Classical SEC formulations involve an exponential number of constraints, a drawback that can be overcome by using cutting-plane methods. Since developing such methods requires a significant amount of time, we chose to formulate the SEC in a "compact" way by using a polynomial set of constraints. The latter constraints are *multicommodity flows* constraints (see [1], p. 410).

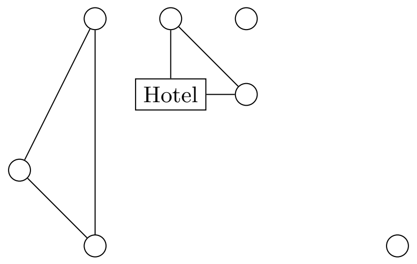

Fig. 2.1: A possible solution containing two subtours

The idea underlying multicommodity flow is to send one unit of flow of an "imaginary" product (in the sense that this product does not represent any physical object being transported) from the hotel to each node in each daily tour. This forces each daily tour to be connected. This can be implemented by adding to the model the following constraints, where w_{ij}^{kl} is the flow on $(i,j) \in A$ whose final destination is $k \in V^0$ in tour l.

$$\sum_{j \in V^i} w_{ij}^{kl} - \sum_{j \in V^i} w_{ji}^{kl} = \begin{cases} y_k^l, & i = 0 \\ -y_k^l, & i = k \\ 0, & i \neq 0, k \end{cases} \qquad i \in V, k \in V^0, l = 1, \dots, d$$

$$0 \leq w_{ij}^{kl} \leq x_{ij}^l \quad (i,j) \in A, k \in V^0, l = 1, \dots, d$$

Because of time constraints, no exact methods for the weekly problem were implemented. Instead the team developed heuristic methods, as described in the next section.

2.5 Heuristic Methods for the Weekly Problem

2.5.1 *TOP: Heuristic Methods*

Heuristic methods are often preferred to solve this type of problem, since they can provide useful (if not always optimal) solutions significantly faster than exact methods ([4], p. 4). In order to "feed" the annual problem, we require a large number of weekly solutions. It is thus crucial to compute such solutions quickly. Heuristic methods are also interesting for solving the TOP model presented above, because they can provide starting solutions that accelerate the running time of exact algorithms. For VRP-type problems, the *local search* heuristic methods are effective and can be implemented relatively quickly.

2.5.2 *TOP: Local Search*

Local search is a type of heuristic method by which routes (obtained via a "greedy" algorithm or simply empty routes) are improved by a series of *moves*. There are several types of moves (a more detailed description of local search moves can be found in [4], pp. 4-6).

- Location-based moves:
  - **Add (i, j):** add location i (not in tour) immediately after j (in tour);
  - **Switch (i, j):** insert location i (not in tour) in place of j (in tour).
- Routing-based moves:
  - **2-opt (i, j):** replace 2 arcs (one starting at i, the other ending at j) in a tour by 2 other arcs (intra-route);
  - **2-opt* (i, j):** replace 2 arcs (one starting at i, the other ending at j) in different tours by 2 other arcs (inter-route);
  - **Relocate (i, j):** remove a sequence of visits (starting at i and ending at j) from a tour and insert the sequence in a tour (intra- or inter-route);
  - **Swap (i, j):** swap locations i and j in a tour or in 2 tours (intra- or inter-route).

The goal of these moves is to improve the current solution by changing one of the arcs within it. This is usually carried out by trying each move on each random (i, j) couple and determining whether the solution thus obtained is better than the original. Alternate solutions may also be obtained by restarting after perturbing the solutions (*Iterated Local Search*) or restarting the entire process from scratch (*Multistart/GRASP*).

These methods can become "stuck" on solutions that are locally optimal only, and so methods such as *Simulated Annealing* (SA), *Tabu Search* (TS), or *Variable Neighborhood Search* (VNS) can be used to try to obtain solutions that are globally optimal.

In order for the heuristic methods to be efficient, complex data structures need to be implemented for structuring and keeping solutions in memory. Doubly-linked lists and arrays of pointers allow moves to be performed in $O(1)$ time, and feasibility checking for forward and backward times to be performed in $O(1)$ time (see [5]).

2.6 Conclusions

Over the course of the workshop, the team was able to propose a mathematical formulation of the problem. This involved decomposing the problem in a "natural" fashion based on the planning horizon: a year subdivided into weeks. The team built a mathematical programming model for the annual and weekly problems, with a Python code for solving the annual problem. The team also wrote a Python code for preprocessing the data provided by the company and to solve the weekly problem using local search methods.

Because of the limited scope of the workshop, several goals were not attained. In particular, one still has to combine the solutions for the weekly schedules in order to solve the annual problem. The ultimate goal is to propose a method generating high-quality schedules for the annual problem. Exact methods to generate optimal results for both the weekly problem and the annual problem would also be desirable.

Acknowledgments

We would like to thank The Co-operators for submitting this problem, as well as Mathieu Giguère and Jean-Michel Plante for their help and support throughout the workshop.

References

1. G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.
2. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Intersci. Ser. Discrete Math. Optim. Wiley, New York, 1988.
3. J. Renaud, F. F. Boctor, and G. Laporte. An improved petal heuristic for the vehicle routeing problem. *J. Oper. Res. Soc.*, 47(2):329–336, 1996.
4. P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: a survey. *European J. Oper. Res.*, 209(1):1–10, 2011.
5. T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. Time-window relaxations in vehicle routing heuristics. *J. Heuristics*, 21(3):329–358, 2015.