SOLVING NON-QUADRATIC MATRICES IN ASSIGNMENT PROBLEMS WITH AN IMPROVED VERSION OF VOGEL’S APPROXIMATION METHOD

Maximilian Selmair
Alexander Swinarrew
BMW Group
80788 Munich, Germany
maximilian.selmair@bmw.de

Klaus-Jürgen Meier
University of Applied Sciences Munich
80335 Munich, Germany
klaus-juergen.meier@hm.edu

Yi Wang
University of Plymouth
PL4 8AA Plymouth, United Kingdom
yi.wang@plymouth.ac.uk

Abstract—The efficient allocation of tasks to vehicles in a fleet of self-driving vehicles (SDV) becomes challenging for large-scale systems (e.g., more than hundred vehicles). Operations research provides different methods that can be applied to solve such assignment problems. Integer Linear Programming (ILP), the Hungarian Method (HM) or Vogel’s Approximation Method (VAM) are frequently used in related literature (Paul 2018; Dinagar and Keerthivasan 2018; Nahar et al. 2018; Ahmed et al. 2016; Kornukoğlu and Ballı 2011; Balakrishnan 1990). The underlying paper proposes an adapted version of VAM which reaches better solutions for non-quadratic matrices, namely Vogel’s Approximation Method for non-quadratic Matrices (VAM-nq). Subsequently, VAM-nq is compared with ILP, HM and VAM by solving matrices of different sizes in computational experiments in order to determine the proximity to the optimal solution and the computation time. The experimental results demonstrated that both VAM and VAM-nq are five to ten times faster in computing results than HM and ILP across all tested matrix sizes. However, we proved that VAM is not able to generate optimal solutions in large quadratic matrices constantly (starting at approx. $15 \times 15$) or small non-quadratic matrices (starting at approx. $5 \times 6$). In fact, we show that VAM produces insufficient results especially for non-quadratic matrices. The result deviate further from the optimum if the matrix size increases. Our proposed VAM-nq is able to provide similar results as the original VAM for quadratic matrices, but delivers much better results in non-quadratic instances often reaching an optimum solution. This is especially important for practical use cases since quadratic matrices are rather rare.

I. INTRODUCTION

The transportation problem is an extensively studied topic in operational research (Díaz-Parra et al. 2014). The methods for solving the mentioned problem aim to minimise the total transportation cost while bringing goods from several supply points (e.g., warehouses) to demand locations (e.g., customers). In general, each transport origin features a fixed amount of goods that can be distributed. Correspondingly, every point of transport destination requires a certain amount of units (Shore 1970). The underlying use case, where tasks have to be assigned to self-driving vehicles (SDVs), differs in some regards from the classical transportation problem. In our case, each vehicle has a capacity restriction of one, i.e., a maximum of one load carrier can be transported at a time. Furthermore, each task corresponds to a demand of one. This basically means that every task can only be allocated to one single vehicle. Additionally, the amount of available vehicles does rarely match the number of unassigned tasks in practice. Since the size of the matrices depends on those two factors, non-quadratic matrices (e.g., $10 \times 50$) are common. There are different approaches that can be applied to solve this kind of problem, e.g., ILP, HM and VAM. While ILP and HM manage to always generate an optimal solution, VAM often fails to do so. Furthermore, those methods vary greatly in the computational demand necessary to solve assignment problems. There are two major reasons involved in the motivation for improving the original VAM. For one, the authors wanted to keep the great performance (computational time) of the original VAM. Secondly, the insufficient results for non-quadratic matrices should be improved significantly, i.e., reaching the optimum. Following these considerations, an improved VAM version, as proposed in this paper, was developed and compared with the three established methods. The goal was to find a solution that provides optimal or near-optimal results while at the same requiring a small amount of resources (computing power).

Despite its age, the approximation method proposed by those authors is still in use nowadays and is subject to recent operations research as the contributions by Banik and Hasan (2018), Ezekiel and Edeki (2018), Hlatká et al. (2017), Ahmed
et al. (2016), and Gani and Baskaran (2014) show.

Already Shimshak et al. (1981) extended the original VAM with rules that apply in case of ties, e.g. the same maximum cost differences occur. Out of the four cases using either individual rules or a combination of them, only one case manages to generate slightly better solutions than the original VAM in regards to costs. Furthermore, they used only small matrices ($5 \times 5$, $5 \times 10$ and $10 \times 10$) which does not provide any information on the results achieved in large-scale applications. Goyal (1984) further tried to improve Shimshak’s approach in case of unbalanced transportation problems, i.e. the total supply does not correspond to the total demand. Again, only a conclusion to this paper.

The last chapter contains a brief overview of the experiments as well as the discussion of the corresponding results. The last chapter contains a brief conclusion to this paper.

This paper is structured as follows: in the second chapter a detailed description of VAM as well as a brief explanation of the HM and the ILP are given. The third chapter features the detailed description of VAM as well as a brief explanation of the HM and the ILP are given. The third chapter features the detailed description of VAM and the HM. The ILP approach will be adapted to some areas from conventional examples (e.g. the vehicles can change the classic VAM in order to achieve better results and move closer to an optimal solution which can be achieved for example by ILP or HM (Paul 2018; Dinagar and Keerthivasan 2018; Nahar et al. 2018; Ahmed et al. 2016; Korukoğlu and Ballı 2011; Balakrishnan 1990; Goyal 1984; Shimshak et al. 1981).

An example for VAM can be found in Table I through Table III. Here, the rows are represented by vehicles ($V_i$) and the columns by tasks ($T_j$). The costs ($c_{ij}$) are the corresponding cell values. The row differences can be found in $\Delta i$ and the column differences are saved in $\Delta j$. Starting with the initial matrix (Table I), it is evident that the biggest difference can be found in the third row featuring the lowest value in the third column (Table II). Accordingly, task 3 is assigned to vehicle 3. After the allocation, the third row and column are eliminated (Table III).

**TABLE I. Initial Matrix to be solved by VAM**

<table>
<thead>
<tr>
<th>$c_{ij}$</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$\Delta i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V1$</td>
<td>200</td>
<td>100</td>
<td>400</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>$V2$</td>
<td>60</td>
<td>80</td>
<td>30</td>
<td>350</td>
<td>80</td>
</tr>
<tr>
<td>$V3$</td>
<td>210</td>
<td>300</td>
<td>70</td>
<td>150</td>
<td>30</td>
</tr>
<tr>
<td>$V4$</td>
<td>120</td>
<td>510</td>
<td>340</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>$V5$</td>
<td>70</td>
<td>80</td>
<td>40</td>
<td>400</td>
<td>30</td>
</tr>
<tr>
<td>$\Delta j$</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE II. Matrix featuring the identified biggest difference (80)**

<table>
<thead>
<tr>
<th>$c_{ij}$</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$\Delta i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V1$</td>
<td>200</td>
<td>100</td>
<td>400</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>$V2$</td>
<td>60</td>
<td>80</td>
<td>30</td>
<td>350</td>
<td>80</td>
</tr>
<tr>
<td>$V3$</td>
<td>210</td>
<td>300</td>
<td>70</td>
<td>150</td>
<td>30</td>
</tr>
<tr>
<td>$V4$</td>
<td>120</td>
<td>510</td>
<td>340</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>$V5$</td>
<td>70</td>
<td>80</td>
<td>40</td>
<td>400</td>
<td>30</td>
</tr>
<tr>
<td>$\Delta j$</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
### B. Integer Linear Programming

As already stated previously, ILP is able to find an optimal solution for different scenarios, even large-scale problems. Initially, one has to formulate an objective function as well as applicable restrictions in order to receive correct results. According to Osman (1995) and following the adoption of the ILP to fit the use case at hand, the objective function reads as follows:

\[
\min \sum_{j \in J} \sum_{v \in V} d_{jv} \cdot c_{jv} \quad (1)
\]

\[
\sum_{v \in V} d_{jv} = 1 \quad \forall j \in J \quad (2)
\]

\[
\sum_{j \in J} d_{jv} \leq 1 \quad \forall v \in V \quad (3)
\]

\[
d_{jv} \in \{0, 1\} \quad \forall j \in J, \forall v \in V \quad (4)
\]

The goal of the objective function (1) is to minimise the sum of all costs \(c_{jv}\) for all jobs \(J = 1, \ldots, m\) and for all vehicles \(V = 1, \ldots, n\) which is the result of multiplying the decision variable \(d_{jv}\) with the corresponding costs which arise when a job \(j\) is assigned to a vehicle \(v\). The first constraint (2) ensures that every job is assigned to a vehicle while the second constraint (3) makes sure that each vehicle’s capacity of 1 is not exceeded, i.e. each vehicle can execute a maximum of one job at a time. The last constraint (4), which applies for both jobs and vehicles, restricts the decision variable \(d_{jv}\) to binary values.

### C. Hungarian Method

The Hungarian Method was initially proposed by Kuhn (1955) to solve the Generalised Assignment Problem (GAP). Similar to the ILP, the HM is able to find an optimal solution to said problem. The algorithm solves \(n \times n\) matrices (e.g. \(10 \times 10\)) by carrying out the following steps until an optimum solution is found:

1. Find the minimum value in each column and subtract this value from all other values in the corresponding column.
2. Find the minimum value in each row and subtract this value from all other values in the corresponding row.
3. Draw lines through the columns and rows so that all zero values of the matrix are covered by as few lines as possible.
4. Check if the number of lines equals \(n\). If it does, an optimal allocation of the zero values is possible. If the number of lines is smaller than \(n\), an optimal allocation is not yet feasible and step 5 has to be carried out.
5. Find the smallest value which is not covered by a line and a) subtract this value from each not covered row and b) add it to each covered column.
6. Return to step 3.

It has to be noted that in case of \(n \times m\) matrices (e.g. \(10 \times 40\)), an extension takes places to generate \(n \times n\) matrices (e.g. \(40 \times 40\)) since the method only works with quadratic matrices. The additional cells are filled with values that are of the same size as the highest value of the original matrix. This extension requires additional computing power since instead of 400 cells \((10 \times 40)\), the algorithm has to consider 1600 cells \((40 \times 40)\). It is evident that this is a drawback when non-quadratic matrices are to be solved. This is always the case when more tasks than vehicles have to be considered or vice versa.

### D. Comparison ILP / VAM / HM

In order to compare the three methods, experiments have been carried out with different quadratic and non-quadratic matrices using an Intel Core i7-6820HQ 2.70 GHz featuring 32 GB RAM. Figure 1 shows clearly that ILP requires the most computational time for quadratic matrices. Especially, in large matrices the time it takes to finish the calculations rises significantly. HM and VAM on the other hand do not require a lot of time to finish calculating the matrices. In fact, there is almost no difference between them up until \(80 \times 80\) matrices where the HM starts to take longer than VAM. From this point forward, the difference between HM and VAM grows continuously with increasing matrix size. This might lead to the conclusion that it is more sensible to use the HM since it is able to produce optimal solutions while maintaining a relatively low computation time. However, looking at Figure 2 shows that the computation time for HM increases significantly if non-quadratic matrices are involved. This is due to the fact that HM has to generate additional rows or columns to produce quadratic matrices since it is not able to deal with non-quadratic problem instances (see section II). VAM on the other hand can deal with quadratic and non-quadratic matrices regardless of their size in a relatively small amount of time which shows VAMs great scalability.

### III. Improved VAM for Non-Quadratic Matrices (VAM-NQ)

Prior experiments have shown, that the original VAM is not able to produce optimal or at least near-optimal results for non-quadratic matrices (see Figure 3). In fact, the results are in some cases more than 100 % worse than the optimum. It was determined that choosing the row or column featuring the maximum difference from the smaller dimension leads to those insufficient results. This means for example that if the matrix contains more columns than rows, choosing a column with the maximum difference (which is achieved by subtracting cell values in the smaller dimension) might result in worse outcomes. The same obviously applies vice versa if there are more rows than columns. This can be explained with the fact that the bigger dimension obviously features more values and...
can be twice as bad as the optimum value.

In case of continuously with increasing matrix size, even in rather small than the optimum (see Figure 3). The deviations increase in some cases, the cases where it fails are up to 200 % worse.

In general, there are two possible versions of non-quadratic matrices. Either there are more columns than rows or more rows than columns. The description below is based on the first case when a matrix contains more columns than rows. Accordingly, the rows and columns in the description have to be switched when the second case occurs. VAM-nq solves allocation matrices featuring more columns than rows by carrying out the following steps:

1) Calculate the difference between the smallest and the second-smallest cell value for each row.
2) Select the row featuring the biggest difference. If there is a tie among rows, choose the row containing the smallest cell value.
3) Determine the smallest cell value for the selected row and allocate the corresponding task to a vehicle.
4) Eliminate the corresponding row and column that have been used for the allocation.
5) Check if there are still vehicles and tasks left to allocate, and repeat steps 1 - 4 in case that there are.

Upon comparison of the original and the adapted VAM, it becomes evident that there are some variations and simplifications. For one, VAM-nq considers only the rows in case that there are more columns than rows (step 1). Accordingly, only the biggest differences in the rows and the corresponding smallest cell values are considered (step 2 and 3). Applying those variations to the second case (more rows than columns) would mean that only columns, their biggest differences and smallest cell values are considered in steps 1 through 3. With Table IV and Table V, the example of subsection II-A is solved with both versions showing that the proposed VAM-nq provides significant better results even in small non-quadratic cases.
In order to evaluate the performance of VAM-nq as well as its ability to reach optimal solutions, experiments have been carried out by using AnyLogic to generate matrices of different sizes.

### A. Design

The matrices have been randomly generated and randomly filled with uniformly distributed costs ranging from 0 to 1.400. Each matrix has been solved 5,000 times to provide meaningful results. The following overview shows which matrices have been used to generate and evaluate the corresponding key performance indicators (KPIs):

- **Mean Deviation of VAM** from the optimal solution as seen in Figure 3: 15 non-quadratic 5 × n matrices with 5 = {6,…, 20}
- **Mean Computation Times for VAM, HM and ILP** as seen in Figure 1 and Figure 2:
  a) 20 quadratic matrices starting with 10 × 10 and rising to 200 × 200 in steps of 10
  b) 20 non-quadratic matrices starting with 50 × 50 and rising to 50 × 525 in steps of 25
- **Mean Deviation of VAM and VAM-nq from the optimal solution**:
  a) 50 non-quadratic 50 × n matrices with 5 = {51,…, 100}
  b) 17 different mixed matrices (5 × 5, 5 × 50, 10 × 10, 10 × 20, 10 × 30, 10 × 40, 20 × 20, 20 × 60, 30 × 30, 10 × 100, 40 × 40, 50 × 50, 50 × 100, 100 × 100, 100 × 200, 100 × 300)

### B. Results of the Experiments

As can be seen from Figure 4, both the original VAM and VAM-nq are not always able to produce an optimal solution, but are instead on average deviating from it. It is also evident that in non-quadratic instances (as seen in Figure 4) the deviation gap between the original VAM and VAM-nq rises continuously when the size of the non-quadratic matrix is increased. While the deviation of the original VAM continues to grow, the deviation of VAM-nq approaches 0%, i.e. an optimal solution is generated more often. This shows clearly that the proposed method is more suitable to deal with non-quadratic instances than the original method. Figure 5 shows the results of experiments performed by using the original VAM and VAM-nq for different problem instances. In this case, it is also evident that in non-quadratic instances the original VAM produces results that are up to 300% worse than the corresponding optimal solution. VAM-nq on the other hand displays almost no deviation and manages on average to generate an optimal solution in all non-quadratic cases. However, it is also recognisable that in quadratic matrices the original VAM is always slightly better than VAM-nq, but the differences in those cases are negligible.

### V. Conclusion

Experiments have shown that VAM is substantially faster in calculating results than HM and CPLEX-solver (ILP) across all matrix sizes. However, VAM is not able to generate optimum solutions in large quadratic matrices (starting with approx. 15 × 15) or small non-quadratic matrices (starting with approx. 5 × 6). In fact, VAM produces insufficient results in those cases and deviates greatly from the optimum. The proposed adapted version of VAM, introduced as VAM-nq, is able to provide slightly worse results than the original VAM for quadratic instances, but delivers much better results in non-quadratic instances reaching an optimum solution in most of the cases. Based on those findings, the authors propose to use an algorithm that includes both the original VAM and the improved VAM-nq and which is able to switch between those two according to the underlying situation. In case that the underlying matrix is quadratic, the original VAM method

---

**TABLE IV. Solution of the original method (VAM) with objective of 320**

<table>
<thead>
<tr>
<th>cij</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>200</td>
<td>100</td>
<td>400</td>
<td>50</td>
</tr>
<tr>
<td>V2</td>
<td>60</td>
<td>80</td>
<td>30</td>
<td>350</td>
</tr>
<tr>
<td>V3</td>
<td>210</td>
<td>300</td>
<td>70</td>
<td>150</td>
</tr>
<tr>
<td>V4</td>
<td>120</td>
<td>510</td>
<td>340</td>
<td>80</td>
</tr>
<tr>
<td>V5</td>
<td>70</td>
<td>80</td>
<td>40</td>
<td>400</td>
</tr>
</tbody>
</table>

**TABLE V. Solution of VAM-nq with objective of 290**

<table>
<thead>
<tr>
<th>cij</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>200</td>
<td>100</td>
<td>400</td>
<td>50</td>
</tr>
<tr>
<td>V2</td>
<td>60</td>
<td>80</td>
<td>30</td>
<td>350</td>
</tr>
<tr>
<td>V3</td>
<td>210</td>
<td>300</td>
<td>70</td>
<td>150</td>
</tr>
<tr>
<td>V4</td>
<td>120</td>
<td>510</td>
<td>340</td>
<td>80</td>
</tr>
<tr>
<td>V5</td>
<td>70</td>
<td>80</td>
<td>40</td>
<td>400</td>
</tr>
</tbody>
</table>

Fig. 4. Mean deviation of the original VAM and VAM-nq from the optimal solution for non-quadratic matrices in percent (5,000 samples each)
should be used by the algorithm. For non-quadratic problem instances however, the algorithm should switch to the improved VAM-nq.

REFERENCES


AUTHOR BIOGRAPHIES

Maximilian Selmair is doctoral student at the University of Plymouth. Recently employed at the SimPlan AG, he was in charge of projects in the area of material flow simulation. Currently he is working on his doctoral thesis with a fellowship of the BMW Group. His email address is: maximilian.selmair@hmw.de and his website can be found at maximilian.selmair.de.

Alexander Swinarew is a masters graduate in logistics at OTH Regensburg who worked at BMW on his masters thesis during which the proposed method was developed. His email address is: alexander.swinarew@gmail.com.

Prof. Dr. Klaus-Jürgen Meier holds the professor-ship for production planning and logistic systems in the Department of Engineering and Management at the University of Applied Sciences Munich and he is the head of the Institute for Production Management and Logistics (IPL). His e-mail address is: klaus-juergen.meier@hm.edu.

Dr. Yi Wang is a lecturer in business decision making in the Faculty of Business, University of Plymouth, UK. He has special research interests in supply chain management, logistics, operation management, culture management, information systems, game theory, data analysis, semantics and ontology analysis, and neuromarketing. His e-mail address is: yi.wang@plymouth.ac.uk.