Highlights

Analytic Hierarchy Process: how to force consistency of pairwise comparisons matrix

Tatiana Balbi Fraga, Ítalo Ruan Barbosa de Aquino, Regilda da Costa e Silva Menêzes

- generative method for forcing consistenc ;
- multi-product p-batch processing time maximization (MPBPTM) problem definition;
- linear integer programming model for the MPBPTM problem;
- exact optimization method for solving the MPBPTM problem.

Analytic Hierarchy Process: how to force consistency of pairwise comparisons matrix

Tatiana Balbi Fraga[a,*], Ítalo Ruan Barbosa de Aquino[a], Regilda da Costa e Silva Menêzes[a]

[a] *Agreste Academic Center - Federal University of Pernambuco, Avenida Marielle Franco, Nova Caruaru, Caruaru, 55014-900, PE, Brazil*

Abstract

Saaty's Analytic Hierarchy Process is an important method for assigning weight to multiple criteria. Despite this method represents an important inovation, its logic is not complicated at all. First a pairwise comparisons matrix is generated for the multiple criteria, and the normalized eigenvector of this matrix is used as the weight of criteria. But, since pairwise matrixes are usually generated manually and based only on some employee knowhow, there is a huge complexity on generating a consistent pairwise matrix. Especially when many criteria are used. This paper presents two algorithms that can be used to adjust inconsistent matrices, forcing such matrices to have a better consistency rate. The first method is a constructive method that uses the data inserted in the matrix to build a new improved one. The second method iteratively identifies inconsistencies, making minor changes in order

*corresponding author

Email addresses: tatiana.balbi@ufpe.br (Tatiana Balbi Fraga), italo_ruan_@hotmail.com (Ítalo Ruan Barbosa de Aquino), regilda.smenezes@ufpe.br (Regilda da Costa e Silva Menêzes)

to improve the matrix consistency rate.

Keywords: multi-product batch, processing time maximization, mathematical model, analytical solution, LINGO

1. Introduction

2. Algorithms for forcing matrix consistency

Algorithm 1 function *consistencyRate*()

Require: $[a_{ij}]_{i,j=1}^{n}, RI[x]_{x=1}^{10}$
  $\lambda_{max} = mainEigenvalue([a_{ij}]_{i,j=1}^{n})$ [1]
  $CI = (\lambda_{max} - n)/(n-1)$
  $CR \leftarrow CI/RI[n]$
  return CR

2.1. Constructive algorithm

Algorithm 2 function *constructivelyForceConsistency*($[a_{ij}]_{i,j=1}^{n}$)

Require: $[a_{ij}]_{i,j=1}^{n}$
Ensure: $consistencyRate([a_{ij}]_{i,j=1}^{n}) \leq 0.1$
  $CR = consistencyRate([a_{ij}]_{i,j=1}^{n})$
  if $CR \leq 0.1$ **then**
    return $[a_{ij}]_{i,j=1}^{n}$
  else
    for $3 \leq k \leq n$ **do**
      $([a_{ij}]_{i,j=1}^{k}, CR) \leftarrow forceConsistency([a_{ij}]_{i,j=1}^{k})$
      if $CR \leq 0.1$ **then**
        break for
      end if
    end for
  end if
  return $([a_{ij}]_{i,j=1}^{n}, CR)$

Algorithm 3 function $forceConsistency([a_{ij}]_{i,j=1}^{k})$

Require: $[a_{ij}]_{i,j=1}^{n} \quad | \quad consistencyRate([a_{ij}]_{i,j=1}^{n-1}) \leq 0.1$
Ensure: $consistencyRate([a_{ij}]_{i,j=1}^{n}) \leq 0.1$
  $CR \leftarrow consistencyRate([a_{ij}]_{i,j=1}^{n})$
  for $j \leq n-2$ **do**
    for $j+1 \leq k \leq n-1$ **do**
      $g \leftarrow 0$
      $s \leftarrow 0$
      if $a_{nj} > a_{nk}$ and $a_{jk} \leq 1$ **then**
        $g \leftarrow j$
        $s \leftarrow k$
      else if $a_{nj} < a_{nk}$ and $a_{jk} \geq 1$ **then**
        $g \leftarrow k$
        $s \leftarrow j$
      end if
      if $g \neq 0$ **then**
        $([a_{ij}]_{i,j=1}^{n}, CR) \leftarrow reduce([a_{ij}]_{i,j=1}^{n}, g, s)$
        if $CR \geq 0.1$ **then**
          $([a_{ij}]_{i,j=1}^{n}, CR) \leftarrow encrease([a_{ij}]_{i,j=1}^{n}, g, s)$
        end if
      end if
      if $CR \leq 0.1$ **then**
        break for
      end if
    end for
    if $CR \leq 0.1$ **then**
      break for
    end if
  end for
  return $([a_{ij}]_{i,j=1}^{n}, CR)$

Algorithm 4 function $reduce([a_{ij}]_{i,j=1}^{n}, g, s)$

Require: $g, s < n; [a_{ij}]_{i,j=1}^{n}$
  while $a_{ng} > a_{ns}$ **do**
    $a_{ng} \leftarrow a_{ng}^{--}$
    $a_{gn} \leftarrow a_{gn}^{++}$
    $aux_{CR} = consistencyRate([a_{ij}]_{i,j=1}^{n})$
    if $aux_{CR} < CR$ **then**
      $CR = aux_{CR}$
      if $CR \leq 0.1$ **then**
        break
      end if
    else
      $a_{ng} \leftarrow a_{ng}^{++}$
      $a_{gn} \leftarrow a_{gn}^{--}$
      break
    end if
  end while
  return $([a_{ij}]_{i,j=1}^{n}, CR)$

2.2. Iterative algorithm

3. Tests and results

4. Conclusions and suggestions for future works

In this paper we presented ...

5. CRediT authorship contribution statement

T.B. Fraga: Conceptualization, Project administration, Supervision, Software, Methodology, Validation, Formal analysis, Writing – original draft, Writing – review & editing. Í.R.B. Aquino: Data curation. R.C.S. Menêzes: Data curation.

Algorithm 5 function $increase([a_{ij}]_{i,j=1}^{n}, g, s)$

Require: $g, s < n; [a_{ij}]_{i,j=1}^{n}$
  while $a_{ng} > a_{ns}$ **do**
    $a_{ns} \leftarrow a_{ns}^{++}$
    $a_{sn} \leftarrow a_{sn}^{--}$
    $aux_{CR} = consistencyRate([a_{ij}]_{i,j=1}^{n})$
    if $aux_{CR} < CR$ **then**
      $CR = aux_{CR}$
      if $CR \leq 0.1$ **then**
        break
      end if
    else
      $a_{ns} \leftarrow a_{ns}^{--}$
      $a_{sn} \leftarrow a_{sn}^{++}$
      break
    end if
  end while
  return $([b_{ij}]_{i,j=1}^{n}, CR)$

Algorithm 6 function a_{ij}^{++}

Require: $a_{ij} \quad | \quad a_{ij} \in \{1/9, 1/7, 1/5, 1/3, 1, 3, 5, 7, 9\}$
  if $a_{ij} \geq 1$ and $a_{ij} \neq 9$ **then**
    $a_{ij} \leftarrow a_{ij} + 2$
  else if $a_{ij} < 1$ **then**
    $a_{ij} \leftarrow 1/(1/a_{ij} - 2)$
  end if
  return a_{ij}

Algorithm 7 function a_{ij}^{--}

Require: $a_{ij} \quad | \quad a_{ij} \in \{1/9, 1/7, 1/5, 1/3, 1, 3, 5, 7, 9\}$
  if $a_{ij} \geq 3$ **then**
    $a_{ij} \leftarrow b_{ij} - 2$
  else if $a_{ij} \leq 1$ and $a_{ij} \neq 1/9$ **then**
    $a_{ij} \leftarrow 1/(1/a_{ij} + 2)$
  end if
  return a_{ij}

Algorithm 8 function iterativelyForcingConsistency()

Require: $\mathrm{M}[a_{ij}]_{i,j=1}^{n}$
Ensure: $consistencyRate([b_{ij}]_{i,j=1}^{n}) \leq 0.1$
  return $\mathrm{M}[b_{ij}]_{i,j=1}^{n}$

6. Acknowledgments

We are enormously grateful to Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and to Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the financial support provided to our projects. We also thank LINDO systems team for the LINGO software license, without which this work would not have been possible and the to the owner of the company in the plastics sector, who allowed us to learn about his company's production process. Finally, we would like to thank Pró-reitoria de Extensão e Cultura da UFPE (PROExC) and the Research Director of Propesqi (Pró-reitoria de Pesquisa e Inovação da UFPE) for their support and recognition of our work, and dear Professors Antônio José da Silva Neto and João Flávio Vieira de Vasconcellos from IPRJ/UERJ, who contributed significantly to the formation of essential skills for the development of our projects. We also thank my co-worker Marcos Luiz Henrique, for having helped by evaluating the mathematical model and solution method proposed in this paper.

References

Eilon. (1985). Multi-product batch production on a single machine - A problem revisited. *OMEGA Int. J. of Mgmt Sci.*, Vol. 13 (5), pp. 453–468.

Fowler, J.W. and Mönch, L. (2022). A survey of scheduling with parallel batch (p-batch) processing. *European Journal of Operational Research*, Vol. 298, pp. 1–24.

Fraga, T.B. (2023). COPSolver: open source software for solving combinatorial optimization and other decision prob- lems - library for solving the multi-product p-batch processing time maximization problem *Software Impacts*, invited paper, in press.

Fumero Y., Moreno M. S., Corsano, G., Montagna, J. M. (2016). A multiproduct batch plant design model incorporating production planning and scheduling decisions under a multiperiod scenario. *Applied Mathematical Modelling*, Vol. 40, pp. 3498–3515.

He, Y., Hui, C-W. (2008). A rule-based genetic algorithm for the scheduling of single-stage multi-product batch plants with parallel units. *Computers and Chemical Engineering*, Vol. 32, pp. 3067–3083.

Kashan, A. H., and Ozturk, O. (2022). Improved MILP formulation equipped with valid inequalities for scheduling a batch processing machine with nonidentical job sizes. *Omega*, Vol. 112, pp. 102673.

Kim, M., Jung, J. H. and Lee, I. (1996). Intelligent scheduling and monitoring for multi-product networked batch processes. *Computers chem. Engn*, Vol. 20 (Suppl.), pp. 1149–1154.

Li, C., Wang, F., Gupta, J.N.D., Chung, T. (2022). Scheduling identical parallel batch processing machines involving incompatible families with different job sizes and capacity constraints. *Computers & Industrial Engineering*, Vol. 169, pp. 108115.

Liu, G., Li, F., Yang, X., and Qiu. S. (2020). The multi-stage multi-product batch-sizing problem in the steel industry. *Applied Mathematics and Computation*, Vol. 369, 124830.

Méndez, C.A., Henning, G.P., Cerdá, J. (2000). Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers and Chemical Engineering*, Vol. 24, pp. 2223–2245.

Méndez, C.A., Cerdá, J. (2003). Dynamic scheduling in multiproduct batch plants. *Computers and Chemical Engineering*, Vol. 27, pp. 1247–1259.

OMEGA Journal. (1993). Single Machine Multi-product Batch Scheduling: Testing Several Solution Methods. *OMEGA Int. J. of Mgmt Sci.*, Vol. 21 (6), pp. 709–711.

Petkov, S. B., and Maranas, C. D. (1998). Design of Single-Product Campaign Batch Plants under Demand Uncertainty. *AIChE Journal*, Vol. 44 (4), pp. 896–911.

Ravemark, D. E., and Rippin, D. W. T. (1998). Optimal design of a multi-product batch plant. *Computers chem. Engng*, Vol. 22 (1-2), pp. 177–183.

Shi, B., Qian, X., Sun, S., Yan, L. (2017). Rule-based scheduling of multi-stage multi-product batch plants with parallel units. *Chinese Journal of Chemical Engineering*, in press.