Production Engineering (2021) 15:335–342
https://doi.org/10.1007/s11740-021-01019-5

A comparison of methods for determining performance based employee deployment in production systems

Jonas Ast[1] · Raed Wasseghi[1] · Peter Nyhuis[1]

Received: 23 November 2020 / Accepted: 8 January 2021 / Published online: 28 January 2021

Abstract
Employee deployment is a crucial process in production systems. Based on qualification and individual performance of employees, deployment decisions can lead to ambiguous outcomes. This paper first reviews the state of the art and further compares two methods based on combinatorial analysis for employee deployment. Therefore, this paper emphasizes the costs and benefits of a Brute Force and an alternative Greedy method. When considering the qualification and individual performance of each employee, both algorithms provide working solutions. In direct comparison, the outcome of the alternative Greedy algorithm is more efficient in terms of calculation time whereas the Brute Force method provides the combination with the global optimum. This means calculation time as well as quality of outcome differ. The exponential growth of employee allocation possibilities depends on the amount of employees and leads to high calculation times, when using a Brute Force method. The comparison of both methods reveal that the proposed alternative Greedy algorithm reaches nearly as high outcomes as the Brute Force does, with significantly less calculation time. Furthermore, this paper offers an insight into the impact of deployment decisions within production systems.

Keywords Optimization algorithm · Employee deployment · Operational planning · Production management · Task assignment

1 Introduction and problem description

In production systems with low level of automation and high number of employees task assignment is a relevant process to secure planned outcomes. The matching of employees and tasks (or workstations) require a description of requirements such as necessary qualifications. Even in an ideal setup where all employees hold all possible qualifications it can be assumed that due to varying level of competencies the overall outcome differs. The task of assigning the optimum regarding a possible target (e. g. performance) is known as the job assignment problem [1]. The general problem of assigning the best fit is the exponentially growing amount of combinations, which is caused by the factorial of possible combinations. In a production system with five employees there are $c = 5! = 120$ combinations. Adding five more employees leads to a system with $c = 10! = 3.628.800$ possible combinations. This effect known as combinatorial explosion is a well known problem in many research areas such as biology, chemistry and computer sciences [2]. Combinatorial explosion leads to an exponentially increasing amount of possible solutions for an optimization problem and is responsible for long calculation times. Transferring this to a production system makes it difficult to find the best fit for job assignments in environments with many employees. To narrow the search space for possible solutions, boundary conditions can be implemented. The criteria used for this investigation are qualification and individual performance. Each employee is evaluated for each task or workstation by differentiating between mandatory qualifications and an evaluation of efficiency based on a scale from 75 to 120%, whereas 100% is the equivalent of an average employee with normal performance. In this scenario an employee without a mandatory qualification has an efficiency level equal to 0. A production system with n employees and m tasks has a performance matrix with the dimension $n \times m$. By using combinatorial analysis methods the assignment of jobs can be processed. The purpose of this paper is to compare two

✉ Jonas Ast
ast@ifa.uni-hannover.de

1 Institute of Production Systems and Logistics, Leibniz University Hanover, An der Universität 2, 30823 Garbsen, Germany

basic methods regarding costs and benefits with the goal of providing an optimum for the employment based on performance. For companies the knowledge about consequences of erroneously located employees can lead to a higher understanding of drivers for positive and negative outcomes.

2 State of the art

2.1 Employee deployment

The main characteristic of employee deployment is the assignment of available workforce to activities or organizational units at a certain time [3]. At the current stage, the assignment of workforce to tasks is managed subjectively by opinion or experience of respective managers [4]. Therefore, several human and organizational factors that impact the work performance and thus the cost efficiency of the employee deployment, are not considered comprehensively. Existing systems and professional solutions do not prioritize human resources by a target system of such influential factors, but rather consider short term cost efficiency as an essential condition [5]. The world of work in production transforms due to the emergence of new trends such as resource scarcity and new technologies. As a result, values like flexibility, transparency and autonomy become more relevant [6]. This changes the view of employees as a production factor and directs the focus onto employees as a complex resource with characteristics and influential factors such as goals, desires and emotions that determine the performance of the workforce [7, 8]. For that reason, the deployment of this complex resource requires new strategies in order to consider a broader target system of factors. With the advent of technological advances new methods for the optimization of employee assignment, that do not only consider economic factors, but also allow to involve human factors, emerge.

2.2 Assignment problem and its variations and methods

This section discusses the (linear) assignment problem as a fundamental problem in operations research, that is implemented across many branches [9]. There are several forms of the linear assignment problem, which vary depending on the context of use. The assignment problem and its variations are generally subordinate to operations research, and are usually sub-categorized under the combinatorial optimization, which is a specific subarea of operations research. Depending on the forms and conditions of variations, the respective assignment problems are assigned to the subdivisions of the combinatorial optimization, such as linear optimization or the transportation theory. There are several variations of the assignment problem, such as the Generalized Assignment Problem [10], Linear Bottleneck Assignment problem [11], the Quadratic Assignment problem [12] or the Weapon-Target Assignment problem [13] depending on the context of use. The classification of the assignment problem in the overall context thus varies. The classical assignment problem is considered a special class of linear optimization problems and a variation of the transportation theory. Under this consideration several linear methods, such as the simplex method and its derivations as well as the primal-dual Hungarian method are existent to solve the assignment problems and some of its variations [14, 15]. However, the assignment problem becomes a discrete optimization problem for positive integer values and can be solved heuristically by approximation methods, such as the Greedy method [14]. These methods can be categorized under Exact method, Heuristic techniques as well as Hybrid methods. In addition, further categorization of the Heuristic techniques can be made, for instance, local search based (e.g. Greedy) or population search based (e.g. Particle Swarm Optimization) [9]. The Particle swarm optimization (PSO) combines local and global search methods and is characterized by globally distributed particles that move within the search space, with the goal to achieve the optimal position. Salman et al. [16] have adapted the PSO technique to the assignment problem and concluded that the PSO provides a viable method. Beside the large number of problem variations and solving method variations, several classifications regarding the classical assignment problem addressing the graph and output size are existent. Ramshaw and Tarjan [17] demonstrate three derivatives of the assignment problem, based on the output size and graph structure. In addition, the terms balanced and unbalanced bipartite graph describe an assignment problem with either the same number of elements $n(employees) = m(tasks)$ (balanced graph) or an unequal number of elements $n(employees) \neq m(tasks)$ (unbalanced graph) [17]. All these variations, derivations and solutions are based on the fundamental goal of the classical assignment problem, that is mathematically defined as

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \rightarrow \min, \tag{1}$$

with

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, m, \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i = 1, \ldots, n, \tag{3}$$

where x_{ij} is the occupancy of a task by a employee. Each entry can take values of 0 or 1. The goal of the traditional assignment problem is to find in a bipartite graph, an assignment or matching of a known size, in which the sum of values of the edges is a minimum. Taking the mathematical description into account, c_{ij} describes the cost or effectiveness of assigning the i-th employee to the j-th task. Equations (2) and (3) demonstrate that each employee needs to be assigned to only one task and vice versa [9]. Adapting the assignment problem to the context of employee deployment, the new goal of the problem dealt by, is to achieve the maximum performance value instead of the minimum costs.

Therefore, this paper presents two methods in order to provide optimal solutions for the average performance (P_{best}) achieved by the deployment of the employees. The discrete assignment problem hereby considers a bipartite graph weighted with performance values that are determined by the productivity of the employees at each workstation or task. In this paper the Brute Force algorithm as an exact method is selected in order to provide the optimal solution for the assignment problem that can be compared to the proposed alternative Greedy Algorithm as heuristic method. Due to its simplicity, it can be assumed that the Greedy method provides the potential basis to be adapted efficiently for specific assignment problems. For this reason, the hereby modified assignment problem is solved heuristically with an alternative Greedy algorithm (AGA) that covers the fundamental principles of the classical Greedy algorithm.

3 Conceptualization of optimization algorithms

Due to the necessity to further develop exiting methods, the following section provides basic knowledge about the proposed algorithms. Furthermore, information regarding the necessary initialization and shared information for both algorithms will be explained.

3.1 Boundaries and initialization

The initialization step for both algorithms includes the generation of a square matrix. The values of the matrix are generated randomly and represent the performance of each employee that is determined by the efficiency. This requires the actual productivity of each employee. In comparison to the planned productivity for the task, the performance of the employee can be evaluated. Due to data restrictions the procurement with this specific data can be an issue. Thus, this papers focuses on the value which can be added to employment decisions if the required data is available. The used values are integer percentage performance values and are set to zero when they are below a certain performance limit. The general productivity can be calculated by comparing input and output of a process [8]. Transferred to tasks within production systems, the input can be described as time and the output as completed tasks e.g. produced products.

$$Productivity = \frac{Output}{Input} = \frac{Completed\ Tasks}{Time} \tag{4}$$

As shown in Eq. (4) the productivity can be calculated based on the provided information of the task. To evaluate the performance of an employee, the productivity can be compared with a plan-productivity which is based on the performance of an average employee with normal performance [18].

$$Performance = \frac{Actual\ Productivity}{Planned\ Productivity} \tag{5}$$

Equation (5) describes the formula for calculating the degree of performance based on established methods for evaluating workplaces [19, 20]. Transferred to this application, the range to evaluate the individual performance of employees ranges from 75 to 120% whereas 100% is the equivalent of an average employee with normal performance. For the proposed methods the productivity limit is equal 75%, which is a threshold value due to the goal of providing minimum qualified employees for the production system. Therefore, all productivity values below 75% in the matrix are set to 0 in order to eliminate the possibility to assign workforce with an insufficient productivity value to a workplace. In order to avoid disruptions in the assignment process of the AGA, the condition is added that at least two employees can potentially work at one workstation and that one employee has the skills to be assigned to at least two workstations. The target value is the average performance value of the deployment situation. This means, the average performance (P_{best}) within the production system is the object of optimization. Furthermore, for both algorithms implemented in this paper, a continuously updating P_{best} value is generated that represents the highest achievable productivity value at the end of the computing process of each algorithm. Moreover, each algorithm has a specific adaptation to its initialization step. The Brute Force Algorithm generates $n!$ permutations of possible assignment combinations $B(i \rightarrow n!)$, whereas the AGA generates an array that provides the information whether an employee is already assigned.

3.2 Brute Force algorithm

The Brute Force method is a widely known optimization method for small search areas. It is based on the principle of calculating all possible solutions for a specific problem to find the highest or lowest possible outcome. Therefore, the bigger the search space the higher is the amount of needed calculation power due to an increase in possible outcomes.

This is based on the previously described combinatorial explosion problem. The exponentially growing amount of possible combinations lead to an exponential growth of calculation time. Thus, the Brute Force method is limited to a specific amount of combinations which means in this scenario to a limited amount of employees. Nevertheless, the Brute Force algorithm is a valid method in limited scenarios, because of the fact that it always finds the best possible solution. The Brute Force method which is used in this paper is shown in Fig. 1. After initializing the parameters, the algorithm proceeds to generate $n!$ permutations based on the number of employees n. Each permutation is transferred into a $n \times m$ matrix with one value in each row and column depending on the value in the permutation. Afterwards all elements > 0 inside the $n \times m$ matrix are set equal to one. When generating the permutations as well as the $B(i)$ matrix repeatedly the calculation time is constant for this part. Because the rows represent an employee and the columns a task the $n \times m$ matrix displays the employee allocation. Each employee gets assigned to a task which means the matrix represents the job allocation. Further the allocation matrix is multiplied with the qualification/individual performance matrix. Thus, the resulting matrix visualises the distribution of the outcome of each tasks. The average value of all n values gives an insight into the average performance of the production system. Due to the sense of a Brute Force algorithm, this calculation needs to be done for $n!$ combinations. This leads to a loop in which the P_{best} stands for the highest average performance for all already calculated variations. After $n!$ calculations P_{best} is definitely the highest possible average outcome of the productions system. Further information can be gained from the loop and the allocation of the combination with the highest average can be presented. For the purpose of this investigation, the maximum amount of employees is limited to 10 due to limited resources. The motivation of using the Brute Force algorithm for allocating employees to tasks is to use it as the basis for further comparisons. Optimization algorithms require a quality criterion in order to provide a possibility to evaluate the outcome. Therefore, the solution of the Brute Force algorithm is used as quality criterion in order to evaluate the result of the AGA.

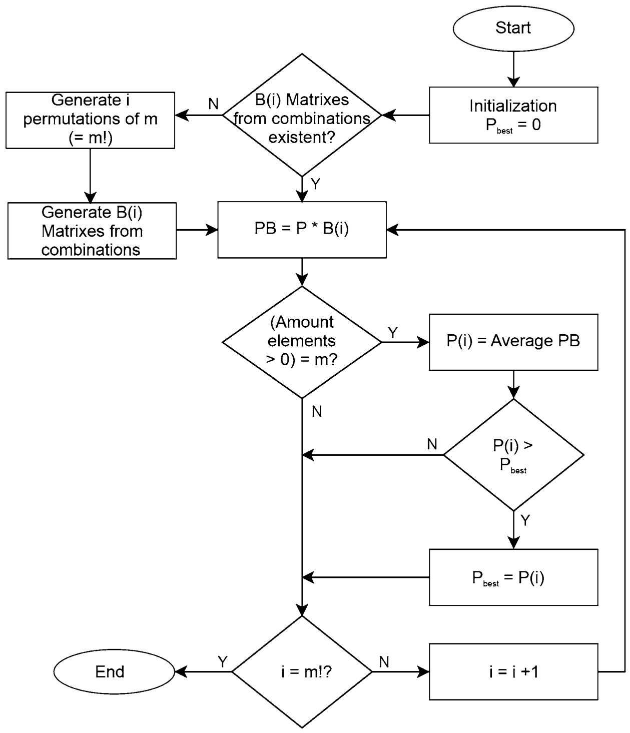
Start
Initialization
P_best = 0
B(i) Matrixes from combinations existent?
N
Y
Generate i permutations of m (= m!)
Generate B(i) Matrixes from combinations
PB = P * B(i)
(Amount elements > 0) = m?
Y
N
P(i) = Average PB
P(i) > P_best
N
Y
P_best = P(i)
i = m!?
Y
N
End
i = i +1

Fig. 1 Model of the Brute Force algorithm

3.3 An alternative Greedy Algorithm

The AGA is a combination of the Greedy principle and the well-known n-queens problem, which has been introduced by Bezzel in [21]. The n-queens problem is a combinatorial problem that is defined by the placement of n non-attacking queens on an $n \times n$ chessboard [21]. Current research shows several methods, such as Brute Force, Backtracking and Permutation Generation algorithms that provide the amount of possible combinations to solve the n-queens problem [22]. However, Wirth [23] presents an alternative Backtracking method that is recursive and proceeds to allocate queen-figures in successive columns starting with column i of n maximum columns. When adapting Wirth's recursive Backtracking algorithm to the proposed employee allocation problem, new constraints have to be set. These constraints are based on the Greedy Algorithm, which starts with the best combination of the matrix. The classic Greedy Algorithm provides a value that is at least 50% ($K = 0.5$) of the optimal solution [24].

$$\frac{P_{best}(classic\ Greedy)}{P_{best}(Optimal\ solution)} \geq K. \tag{6}$$

The proposed AGA seeks to provide better solutions while keeping the beneficial aspects of the classic Greedy Algorithm, such as high computing speed [24]. The main objective of the proposed method is to allocate employees to workstations, whereby the performance of each employee depends on the workstation accordingly. For that reason, each column represents a workstation and each row in the $n \times m$ Matrix represents an employee. Two main constraints, shown in the program flow chart (Fig. 2), define the allocation procedure of the proposed Backtracking Algorithm:

(a) Select the highest available value in the column.
(b) Do not select a value equal to 0 in any column.

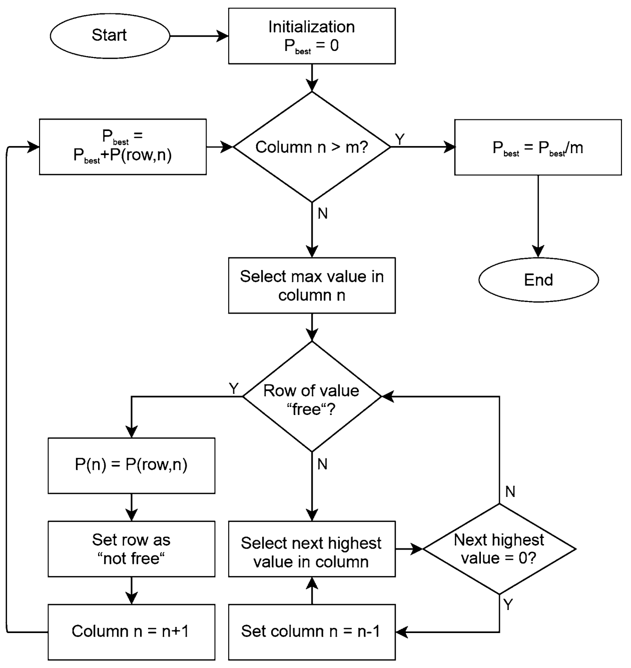

Fig. 2 Model of an alternative Greedy Algorithm

Concerning these two constraint, a third constraint is necessary in order to avoid allocation problems occurring in case the last available values for the last columns are equal to 0. The third constraint defines the Backtracking character and is described as follows:

(c) If after considering (a) and (b) no solution is found, go back ($n - 1$), set the selected highest available value as "not free" and apply condition (a) again.

This eliminates the possibility of an employee with an insufficient performance value being allocated to a workstation. With the application of all stated conditions, the AGA allocates the employees with the highest available performance value to the respective workstation while successively prioritizing the workstations. Thus, the first workstation is staffed with the highest prioritization, while the last workstation has the lowest priority.

4 Exemplary calculation

The comparison is based on generating a random performance matrix with each new calculation process. This simulates a different unit of workforce or different organizational set up. For the purpose of this paper, the P-matrix is randomly generated by an initial algorithm but can be based on a real set of employees within a production system. This requires a process to evaluate the performance of the employees but can be simplified by using a specific scale. By multiplying the randomly generated performance matrix with the deployment matrix, the obtained matrix displays the resulting assignment of each employee and the performance value at the respective task. To generate various results and therefore a representative mathematical model, the procedure for each algorithm has been repeated 1000 times ($w = 1000$). The process as shown in Fig. 3 shows that each repetition w generates a random performance $n \times m$ square matrix P. For the purpose of this paper, the amount of employees is equal to the amount of tasks ($n = m$). Hereby, n represents the amount of employees and m tasks. Afterwards both algorithms calculate their optimal result for the average performance within the production system and save them into a $w \times 2$ matrix, which is $Q(i, model) = P_{best}(i, model)$, whereby $model = 1$ is the AGA and $model = 2$ the Brute Force. In addition, the computing time of each algorithm for each calculation process is saved for further comparison into another matrix $T(i, model) = t_{i,model}$. The abort criterion occurs when variable i, which counts $i = i + 1$ with each repetition, is equal to w. In this program flow chart the general initialization step combines both initialization steps of the algorithms. Furthermore, the calculation process is performed for the number of employees from five to ten employees. Hereby, the number of employees is limited to ten not only due to the significantly high computational time, but also due to the limits of the working memory. The results of $w = 1000$ calculations are compared within two tables (Tables 1 and 2). The calculation was processed with a varying amount of employees in the production system, to simulate a comparison within differing production systems.

Table 1 compares the benefits of the algorithms by displaying the performance in comparison. Therefore,

Table 1 Benefits of algorithms: performance coverage (K), amount of combinations with higher results (A) and ratio of A to valid combinations (R) for $w = 1000$ calculations

Employees (n)	K_{min} (%)	$\overline{K}$ (%)	K_{max} (%)	A_{min}	$\overline{A}$	A_{max}	R_{min} (%)	$\overline{R}$ (%)	R_{max} (%)
5	85.47	97.50	100	0	2.59	26	0	14.85	92.86
6	84.78	97.15	100	0	6.87	90	0	12.17	87.14
7	87.10	96.91	100	0	18.88	425	0	8.91	74.02
8	88.85	96.88	100	0	48.05	1236	0	4.95	69.66
9	88.75	96.73	100	0	172.23	6639	0	3.00	59.10
10	88.80	96.71	100	0	576.22	22616	0	1.73	45.51

Table 2 Cost of algorithms: computing time t for alternative Greedy Algorithm (AGA) and Brute Force algorithm (BF) for $w = 1000$ calculations

Employees (n)	$t_{BF,min}$	$t_{AGA,min}$	$\overline{t_{BF}}$	$\overline{t_{AGA}}$	$t_{BF,max}$	$t_{AGA,max}$
5	$1,47 \cdot 10^{-4}\ s$	$2,57 \cdot 10^{-5}\ s$	$2,56 \cdot 10^{-4}\ s$	$6,35 \cdot 10^{-5}\ s$	$3,90 \cdot 10^{-3}\ s$	$7,80 \cdot 10^{-3}\ s$
6	$7,93 \cdot 10^{-4}\ s$	$2,52 \cdot 10^{-5}\ s$	$1,00 \cdot 10^{-3}\ s$	$6,17 \cdot 10^{-5}\ s$	$5,90 \cdot 10^{-3}\ s$	$4,50 \cdot 10^{-3}\ s$
7	$5,70 \cdot 10^{-3}\ s$	$3,50 \cdot 10^{-5}\ s$	$6,70 \cdot 10^{-3}\ s$	$8,20 \cdot 10^{-5}\ s$	$1,72 \cdot 10^{-2}\ s$	$5,10 \cdot 10^{-3}\ s$
8	$4,94 \cdot 10^{-2}\ s$	$6,53 \cdot 10^{-5}\ s$	$7,26 \cdot 10^{-2}\ s$	$1,62 \cdot 10^{-4}\ s$	$0,14\ s$	$6,20 \cdot 10^{-3}\ s$
9	$0,48\ s$	$7,76 \cdot 10^{-5}\ s$	$0,56\ s$	$2,13 \cdot 10^{-4}\ s$	$1,26\ s$	$2,55 \cdot 10^{-2}\ s$
10	$4,89\ s$	$8,48 \cdot 10^{-5}\ s$	$5,81\ s$	$2,39 \cdot 10^{-4}\ s$	$10,54\ s$	$3,75 \cdot 10^{-2}\ s$

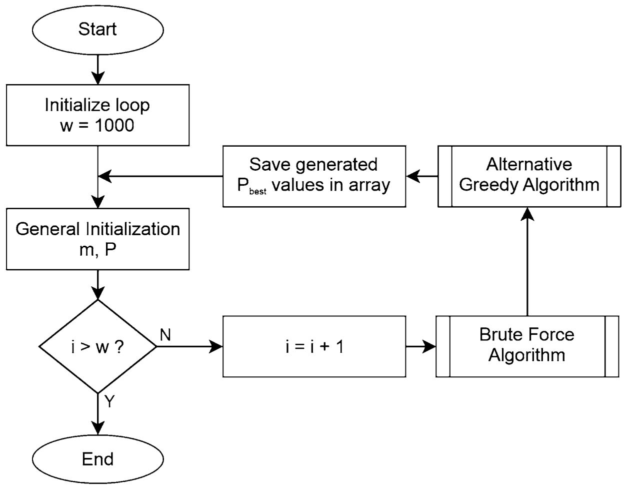
Start
Initialize loop w = 1000
Save generated P_best values in array
Alternative Greedy Algorithm
General Initialization m, P
i > w ?
N
Y
i = i + 1
Brute Force Algorithm
End

Fig. 3 Procedure of calculation process

K describes the ratio between the highest result of both algorithms.

$$K = \frac{P_{best}(AGA)}{P_{best}(BF)} \tag{7}$$

The variable A is the amount of combinations with higher results than the result proposed by the AGA.

$$A = \sum_{i=1}^{w} x_i \tag{8}$$

with

$$x_i = \begin{cases} 1 & P_{best,i}(BF) > P_{best,i}(AGA) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

And R describes the ratio of the amount of combinations with higher results to the amount of valid combinations.

$$R = \frac{A}{Valid\ Combinations} \tag{10}$$

Table 2 compares the costs of the algorithms in terms of calculation time.

5 Results and discussion

This chapter discusses the main results achieved by comparing the optimal solution with the heuristically obtained solution. Objects of this comparison are the maximal computed performance of each algorithm, the calculation time as well as the coverage value K of the proposed AGA and the amount of combinations that would provide a higher overall performance value than the heuristically obtained performance value. The results of the exemplary calculation show distinct differences within the results. Table 1 indicates that the coverage of the P_{best} of each algorithm is constant over an increasing amount of employees. By determining the coverage based on the best possible outcome provided by the Brute Force algorithm, the AGA reaches up to 100% coverage. The average over $w = 1000$ calculations with different amount of employees is $\sim$ 97%. The Brute Force algorithm always provides the best combination with the highest result but also shows the span of possible and valid results. It can be assumed that there is a varying amount of combinations between the provided solutions of the two algorithms. This amount of combinations with higher results than the result of the AGA (A) increases with more employees but decreases in comparison to the maximum amount of valid combinations. Especially when the coverage is lower, a higher amount of combinations can be expected. The comparison of costs and benefits of the two algorithm is based on the quality of the result (benefit) and the calculation time (cost). The exponentially increasing calculation time of the Brute Force algorithm is a result of the already described combinatorial explosion. Due to the nature of the Brute Force algorithm the quality of the provided outcome is the best optimum of the combinatorial problem. Whereas the computing time of the Brute Force algorithm increases exponentially, the computing time of the AGA increases low but steady. This can be explained due to the structure and method of the AGA. In direct comparison to the BF, the result of the AGA reaches an average of $\sim$ 97% of the optimum which is considered as a high quality outcome. By comparing the calculation times (Table 2) the discrepancy is considered as an enormous difference. Therefore, the comparison shows the coverage of the outcome is high which means the outcome of the AGA is nearly as good as the outcome of the Brute Force algorithm.

By considering the calculation time as the costs of applying the algorithms, the Brute Force turns out to be the inferior algorithm. Thus, the minimal discrepancy of the outcome does not justify the huge difference in calculation time. The ratio between additional expenses and quality of outcome supports the conclusion that the Brute Force algorithm may serve the highest possible outcome but considering the effort it is definitely the inferior algorithm. When calculation time is not an issue (e.g. within research purposes) the Brute Force algorithm can be used as a valid algorithm for comparing purposes within certain parameters. The Brute Force method is limited to the effect of combinatorial explosion. Within practical applications the AGA turns out to be a valid method in terms of providing viable solutions with minimal effort.

6 Limitations

In order to affirm the obtained results in Sect. 4 and validate the proposed AGA an extended research regarding the implementation of the AGA is necessary. Even though the maximum of 10×10 performance matrix provides viable results, it does not show the limits of the proposed AGA. Therefore, it is uncertain how the AGA behaves in larger production systems and at what point the combinatorial explosion occurs. The calculation times indicate that the time of AGA is not limited to an increasing amount of employees. Besides, it would be interesting to analyse the differences in the output when implementing the AGA onto a divided production system (e.g. 50×50 matrix divided into 25 10×10 matrices) rather than on a large matrix (e.g. 50×50). Another major limitation of the proposed AGA is the successive prioritization of the assignment by columns, i.e. workstations. This leads to a situation in which the last workstations are occupied by the remaining employees and thus not prioritized on the same level as the first workstations. However, this limitation might be used strategically in order to arrange a value-oriented order of workstations, so that high-value workstations with high added value are prioritized more than low-value workstations. The Brute Force algorithm, on the other hand, prioritizes over all workstations equally. Furthermore, the final major limitations of this paper is the criteria under which the employees are assigned to their respective task/workstation. In the exemplary calculation (4) the assignment problem considers exclusively the performance value of each employee based on once productivity at each workplace. The difficulty of the Brute Force algorithm in a case, where more criteria are considered in order to assign the employees is restricted by its early combinatorial explosion. The AGA on the other hand has shown appropriate results in a short computing time, thus provides a potential basis to adapt this algorithm to a larger target system that does not only include the employees' performance or productivity, but also considers other influential human- and production-related indicators. Therefore, it is essential to not only investigate the AGA in terms of problem size, but also regarding an advanced input data model.

7 Conclusion

A comparison of two basic optimization methods to assign available and valid workforce to tasks or workstations is the central work of this paper. Its results are intended for optimized employee deployment within production systems. Although the alternative Greedy algorithm (AGA) does not guarantee the optimal solution, it is reliably close to the optimum. Considering the need for a quality criterion when providing results with an optimization algorithm, the shown results describe the reliability of the results provided by the AGA. In the proposed exemplary calculation the model shows values that are at average $\sim 97\%$ of the optimal solution independent from the amount of employees, which is a remarkable improvement in contrast to the classic Greedy Algorithm. The significant advantage of the AGA is the much shorter computing time compared to the Brute Force Algorithm. This results in a high scalability of this optimization method. Furthermore, the AGA retains a comparable flexibility due to the included backtracking method. This flexibility combined with the significantly high scalability leads to the conclusion, that the proposed AGA can be a promising candidate to complex employee deployment situations within production systems. After considering the compared algorithmic methods, it can be stated that the proposed AGA offers tangible advantages and potential for further development and implementation. The Brute Force method reaches its limits due to an increasing amount of employees in the system. The importance of this research is defined by the increasing digitalization as well as the emerging trends that influence the employee deployment in production systems. This paper sets the basis for further investigation of combinatorial optimization methods in the context of employee deployment in production systems and could lead to an algorithmic method that makes use of the advantageous characteristics of existing methods. This might result in an applicable assistant system that meets recently arising HRM requirements in the digital era and regards the workforce as a complex resource.

Acknowledgements This publication is written as part of the project "teamIn". The research and development project is funded by the German Federal Ministry of Education and Research (BMBF) and the European Social Fund (ESF) within the program "Future of Work" (Grant no.: 02L18A140) and is managed by the Project Management Agency Karlsruhe (PTKA). The author is responsible for the content of this publication.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Liu L, Gao X (2009) Fuzzy weighted equilibrium multi-job assignment problem and genetic algorithm. Appl Math Model 33(10):3926–3935
2. Leland BA, Christie BD, Nourse JG, Grier DL, Carhart RE, Maffett T, Welford SM, Smith DH (1997) Managing the combinatorial explosion. J Chem Inf Comput Sci 37(1):62–70
3. Spengler (2019) Moderne Personalplanung. Springer, Wiesbaden. https://doi.org/10.1007/978-3-658-25935-8
4. Li N, Li Y, Sun M, Kong H, Gong G (2017) An optimization method for task assignment for industrial manufacturing organizations. Appl Intell 47(4):1144–1156. https://doi.org/10.1007/s10489-017-0940-1
5. Cabanillas C, García JM, Resinas M, Ruiz D, Mendling J, Ruiz-Cortés A (2010) Priority-based human resource allocation in business processes. In: Maglio PP, Weske M, Yang J, Fantinato M (eds) Service-oriented computing, lecture notes in computer science, vol 6470. Springer, Berlin, pp 374–388. https://doi.org/10.1007/978-3-642-45005-1_26
6. Erner M (2019) Management 4.0—Unternehmensführung im digitalen Zeitalter. Springer, Berlin. https://doi.org/10.1007/978-3-662-57963-3
7. Holtbrügge D (2018) Personalmanagement, 7th edn. Springer, Berlin. https://doi.org/10.1007/978-3-662-55642-9
8. Schlick C, Luczak H, Bruder R (2010) Arbeitswissenschaft. Springer, Heidelberg. https://doi.org/10.1007/978-3-540-78333-6
9. Faudzi S, Abdul Rahman S, Abd Rahman R (2018) An assignment problem and its application in education domain: a review and potential path. Adv Oper Res 2018:1–19. https://doi.org/10.1155/2018/8958393
10. Kundakcioglu OE, Alizamir S (2009) Generalized assignment problem. In: Floudas CA, Pardalos PM (eds) Encyclopedia of optimization. Springer, Boston, pp 1153–1162. https://doi.org/10.1007/978-0-387-74759-0_200
11. Burkard RE, Derigs U (eds) (1980) Assignment and matching problems: solution methods with FORTRAN-programs: lecture notes in economics and mathematical systems, vol 184. Springer, Berlin. https://doi.org/10.1007/978-3-642-51576-7
12. Koopmans TC, Beckmann M (1957) Assignment problems and the location of economic activities. Econometrica 25(1):53. https://doi.org/10.1007/978-3-658-25935-80
13. Ahuja RK, Kumar A, Jha KC, Orlin JB (2007) Exact and heuristic algorithms for the weapon-target assignment problem. Oper Res 55(6):1136–1146. https://doi.org/10.1007/978-3-658-25935-81
14. Dempe S, Schreier H (2006) Operations research: Deterministische Modelle und Methoden, 1st edn. Teubner Studienbücher Wirtschaftsmathematik, Wiesbaden. https://doi.org/10.1007/978-3-8351-9055-92
15. Kuhn HW (1955) The hungarian method for the assignment problem. Nav Res Logist Q 2(1–2):83–97. https://doi.org/10.1007/978-3-658-25935-83
16. Salman A, Ahmad I, Al-Madani S (2002) Particle swarm optimization for task assignment problem. Microprocess Microsyst 26(8):363–371. https://doi.org/10.1007/978-3-658-25935-84
17. Ramshaw L, Tarjan RE (2012) On minimum-cost assignment in unbalanced bipartite graphs
18. Gummersbach A, Bülles P, Nicolai H, Schieferecke A, Hinschläger M, Mockenhaupt A (2012) Produktionsmanagement: REFA-Schriftenreihe. Handwerk und Technik, Hamburg
19. REFA (2012) REFA-Lexikon: industrial engineering und Arbeitsorganisation, 4th edn. Hanser, München
20. Pekuri A, Haapasalo H, Herrala M (2011) Productivity and performance management—managerial practices in the construction industry. Int J Perform Meas 1:39–58
21. Bell J, Stevens B (2009) A survey of known results and research areas for n-queens. Discrete Math 309(1):1–31. https://doi.org/10.1016/j.disc.2007.12.0435
22. Erbas C, Sarkeshik S, Tanik MM (1992) Different perspectives of the n-queens problem. In: Agrawal JP (eds) Proceedings of the 1992 ACM annual conference on communications, ACM, New York, pp 99–108. https://doi.org/10.1145/131214.131227
23. Wirth N (1971) Program development by stepwise refinement. Commun ACM 14(4):221–227. https://doi.org/10.1007/978-3-658-25935-86
24. Korte B (1985) Was ist kombinatorische optimierung? Chin J Oper Res 85370(4):1–27