OPSEARCH
https://doi.org/10.1007/s12597-020-00475-4

THEORETICAL ARTICLE

A variable neighbourhood search enhanced estimation of distribution algorithm for quadratic assignment problems

T. G. Pradeepmon[1,2] · **Vinay V. Panicker**[1] · **R. Sridharan**[1]

Accepted: 13 August 2020

Abstract
Quadratic Assignment Problem (QAP) is one of the most complex combinatorial optimization problems. Many real-world problems such as printed circuit board design, facility location problems, assigning gates to airplanes can be modelled as QAP. Problems of size greater than 35 is not able to solve optimally using conventional optimization methods. This warrants the use of evolutionary optimization methods for obtaining optimal or near optimal solutions for QAPs. This work proposes a hybridization on a univariate Estimation of Distribution Algorithm, namely the Population Based Incremental Learning Algorithm (PBILA), with Variable Neighbourhood Search (VNS) for solving QAPs. The proposed algorithm is employed to solve benchmark instances of QAP and the results are reported. The results of this work reveals that PBILA on its own is not efficient for solving the QAPs. However, when hybridised with VNS, the algorithm performs well providing best known solutions for 95 test instances out of the 101 instances considered. For most of the test instances, the percentage deviation is less than one percentage. The overall average percentage deviation of the obtained solutions from the best-known solutions is 0.037%, which is a significant improvement when compared with state-of-the-art algorithms.

Keywords Quadratic assignment problem · Estimation of distribution algorithms · Variable neighbourhood search

✉ Vinay V. Panicker
vinay@nitc.ac.in

[1] Department of Mechanical Engineering, National Institute of Technology Calicut, Kozhikode, Kerala, India

[2] Department of Mechanical Engineering, Muthoot Institute of Technology and Science, Ernakulam, Kerala, India

Published online: 28 August 2020

1 Introduction

The Quadratic Assignment Problem (QAP) was introduced by Koopmans and Beckmann [61] as a model for the allocation of indivisible resources. Since then, the QAP has been used for modelling applications such as backboard wiring [91], economic problems [55], facility locations [44], turbine balancing [85], placement of electronic components [40], scheduling [78], typewriter keyboard and control panel design [58], university examination scheduling [4], hospital planning [56] and many more. In spite of the extensive research on this problem, QAP still remains as one of the hardest combinatorial optimization problems and is well known for its diverse applications [8, 38, 45].

Garey and Johnson [47] proved that QAP is NP-Hard and thus, there is no polynomial time algorithm that can solve the problem. QAP instances of size larger than 35 are considered to be very large and cannot be solved in reasonable computational time [8]. QAP is also proven to be very difficult to obtain an approximate solution for large instances with guaranteed performance [52]. The exact QAP solution algorithms are based primarily on dynamic programming, plane algorithm cutting, and branch-and-bound algorithms. Of these, only branch-and-bound algorithms are guaranteed to obtain an optimal solution, which is also guaranteed for size problems below 30 [66], while heuristic and metaheuristic methods provide near-optimal solutions within an acceptable computational time. The set of benchmark instances provided by various researchers are used for assessing the performance of these algorithms. Some of the familiar heuristic and metaheuristics algorithms reported in the literature include Simulated Annealing [72], Genetic Algorithms [5], Tabu Search [31], Ant Colony Optimization [57], Neural Networks [99], Memetic Algorithms [19] and Iterated Local Search [86] and these have been successful in solving QAP, at least to a near optimal solution. In Zaied and Shawky [107], Burkard et al. [26] and Loiola et al. [67] present detailed reviews on QAP with formulations, application areas and solution methodologies.

In the present study, Population-Based Incremental Learning Algorithm (PBILA), a member of Estimation of Distribution Algorithms (EDA), is hybridized with Variable Neighbourhood Search (VNS) and the resulting algorithm is applied for solving the QAP. The EDAs are relatively new variants of Genetic Algorithm in which the crossover and mutation operations are omitted and the new members of the population are generated using a probability distribution of the current population. A VNS incorporating three different neighbourhood structures of QAP is employed to intensify the quality of solutions during the iterations of the EDA. The proposed algorithm is then used to solve the benchmark problems taken from the QAP Library [24].

The rest of the paper is organized as follows: In Sect. 2, the related works about state-of-the-art algorithms for solving QAPs are presented. Section 3 explains the solution methodology adopted focussing on the application of EDA for solving QAPs as well as the Variable Neighbourhood Search procedures. Section 4 describes the experimental setup, the benchmark instances of QAP and the results obtained. Section 5 concludes the paper.

2 Related work

The QAP was introduced in 1957 by Koopmans and Beckmann as a mathematical model for the allocation of indivisible resources where the consideration of the cost of interplant transportation is inevitable for the development of price system [61]. Since its introduction, many researchers have been working on it, developing different formulations, finding new areas of application and evolving new methodologies for finding optimal solutions. Koopmans and Beckmann [61] formulated QAP as a maximization problem in integer linear programming approach. Later, many other formulations such as permutation formulation [15], concave quadratic formulation [18], mixed integer linear programming formulation [77], trace formulation [34], graph formulation [106] and quadratic 0–1 formulation [17, 103] are developed.

The QAP is celebrated for its capability to represent a variety of real-world problems such as plant layout [20], backboard wiring on electrical circuits [22], placement algorithms in VLSI design [41], design of typewriter keyboards and control panels [9], hospital and campus planning [7, 50], ordering of runners in a relay race team [54], ranking of archaeological data [62], the analysis of chemical reactions [43] and many more [23]. Many classical combinatorial optimization problems such as travelling salesman problems [33], graph partitioning problems [60] and maximum clique problems [80] can also be formulated as a QAP.

The QAP has been proven to be of NP-Hard nature and there is currently no polynomial time algorithm available that can optimally solve the problem. Thus, the heuristic and metaheuristic algorithms became popular among researchers working on QAP. There are a number of exact, heuristic and metaheuristic algorithms available in the literature that provide exact and near-optimal solutions for QAP instances. The exact algorithms are guaranteed to provide optimal solutions but are limited to solving only small sized problems. The exact methods used to solve QAPs are Branch-and-Bound [2, 21, 29, 49], Dynamic Programming [98] and Cutting plane algorithms [18]. The heuristic algorithms are quick in providing near-optimal solutions. But the gap between the solution obtained and the optimal solution also increases as the problem size increases [79, 105]. Construction methods [11, 42], limited enumeration methods [101] and methods of improvement [10, 64, 71] are the different categories of heuristic methods. Metaheuristic methods do not guarantee an optimal solution, but in a short time, regardless of the problem size, a near-optimal solution is guaranteed and the solution obtained may also be the optimal one.

Metaheuristic methods are generic iterative procedures for general purposes that guide a heuristic search for promising regions in the solution space of an optimization problem. These methods can be applied to a wide range of optimization problems to make them adapted to a specific problem relatively fewer modifications are required. Generally, these methods are classified into single solution methods and population-based methods. Genetic Algorithm (GA), Simulated Annealing, Tabu Search, Ant Colony Optimization (ACO), etc., and many hybrid algorithms are among the metaheuristic methods. A number of works that use GA

and its variants to solve QAPs have been reported. Tate and Smith [93] reported good results for small instances of QAP by using simple GA. But simple GA is not able to obtain the best-known solutions for larger size problems of size above 20. A number of researchers hybridized GA with other methods to overcome this shortcoming in order to obtain good solutions for higher-sized instances [6, 37, 39, 73, 74]. The literature contains a variety of GA variants that have been used to solve QAPs [6, 12, 32, 94, 104].

Burkard and Rendl [25] were the first to use Simulated Annealing to solve QAPs and it was refined by Connolly [30]. Additional Simulated Annealing applications for QAP resolution can be found in [82, 84, 102]. Parallel Simulated Annealing implementations can be found in [81] to improve performance (in some cases up to 50–100 times better performance can be achieved by parallelization). Simulated Annealing's performance comparison with Tabu Search can be found in Battiti and Tecchiolli [16] and it is argued that Simulated Annealing performs better for comparatively fewer iterations. But, as the complexity of the problem instance increases, more iterations are needed and, in that case, Simulated Annealing is outperformed by Tabu Search. Skorin-Kapov [89] made the first implementation of Tabu Search to solve QAP. Tabu Search is the main candidate for the parallelisation of QAP resolution algorithms and hardware implementations [92]; Matsui et al. [31, 69, 100, 109]. Drezner [36] extended the concentrated tabu search to include more permissible moves for the quadratic assignment problem. There are two extensions suggested and tested. James et al. [59] introduced a cooperative parallel tabu search algorithm (CPTS) where processors exchange information over the course of the QAP solution algorithm.

Gambardella et al. [46] proposed the first implementation of ACO to solve QAP using a HAS-QAP system coupled with a local search. See and Wong [88] provide an in-depth review of ACO concepts, their application, and various ACO algorithms or variants developed to solve QAPs. The literature contains a number of hybrid metaheuristics and variants of simple algorithms. Tseng and Liang [97] proposed a hybrid metaheuristic called ANGEL combining ACO, GA and a local search method with two main phases, namely the ACO phase and GA phase. Over one hundred instances of QAP benchmarks have been tested and the results show that with a high success rate, the proposed algorithm can achieve the optimal solution. Song et al. [90] proposed the Neural Meta-Memes Framework as a combination of various algorithms, namely Tabu Search, Simulated Annealing, Iterated Local Search, and Genetic Algorithm and the proposed framework has been successfully applied to QAPs.

3 Solution methodology

3.1 Estimation of distribution algorithms

Estimation of distribution algorithms (EDAs) [13, 63, 68) are a set of relatively new algorithms that explore the solution space by sampling the probabilistic model constructed from the favourable solutions evolved till now. They are considered to

be variants of genetic algorithms in which the reproduction operations, crossover and mutation are replaced with probabilistic sampling. EDAs belong to the class of population-based stochastic optimization algorithms. As in other population-based algorithms, the EDAs also start with an initial random population sampled from the set of all permissible solutions. The members of the population are then ranked according to their fitness value—higher the fitness, better the solution. The subset of most promising solutions is selected from this ranked population using a selection operator. Then the algorithm constructs a probabilistic model from the selected set of promising solutions. The new set of solutions for the next generation is sampled from this model and the algorithm repeats until the termination criterion is satisfied and returns the best solution found over the generations. The common termination criteria adopted are a maximum number of iterations, homogeneous population, or lack of improvement in the solutions for a certain number of iterations. Figures 1 and 2 represent the pseudocode and flowchart for general EDA respectively.

Based on this general procedure, researchers have developed a number of different algorithms for various categories of optimization problems. The basic categorization of EDAs is done based on the complexity of the probabilistic models demonstrating the relationship between the variables. The broad categories of EDAs consist of univariate, bivariate and multivariate models. There are multiple algorithms in each of these categories and a short review of them is given in the succeeding subsections. The different aspects of EDAs have been studied by many and can be obtained from the works of Hauschild and Pelikan [53], Ceberio et al. [27], Pelikan et al. [83] and Santana et al. [87].

3.2 Population-based incremental learning algorithm

Population Based Incremental Learning Algorithm (PBILA) is one of the simplest EDAs, which assumes no dependence among the variables. The statistical model in use is a real-valued vector with each element independently representing the probability of assigning value 1 to each corresponding bit in a binary string (candidate

1 $\Phi_0 \leftarrow$ Generate the initial population (p individuals)
2 Evaluate the population Φ_0
3 $k = 1$
4 Repeat
  a. $\Psi_{k-1} \leftarrow$ Select $q \leq p$ individuals from Φ_{k-1}
  b. Estimate a new model μ from Ψ_{k-1}
  c. $\Phi_{new} \leftarrow$ Sample p individuals from μ
  d. Evaluate Φ_{new}
  e. $\Phi_k \leftarrow$ Select m individuals from $\Phi_{k-1} \cup \Phi_{new}$
  f. $k = k + 1$
  Until stop condition

Fig. 1 Pseudocode of general EDA

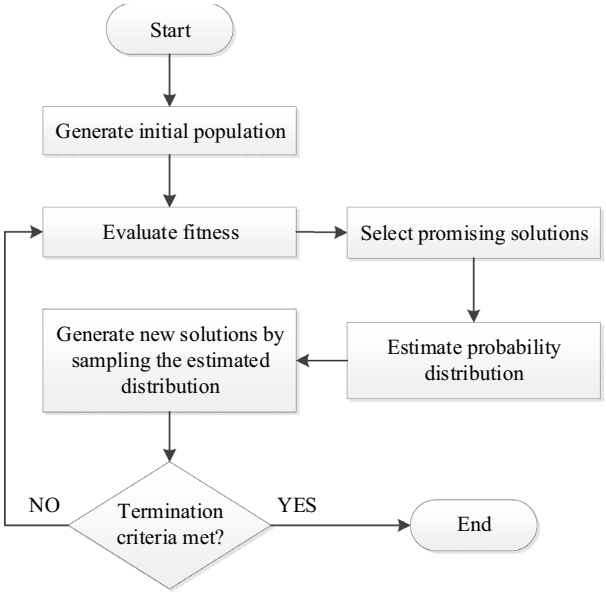

Fig. 2 Flow chart of general EDA

solution). PBILA starts with a probability vector with all elements set to 0.5, which means that each bit in a generated individual is set to 0 or 1 with equal probability. During evolution, the value of each element is updated using the best individual in the population and the probability value drifts away from 0.5, modifying its estimation of the structure of good individuals. Typically, PBILA will converge to a vector with each element close to 0 or 1. During each generation, the probability vector is updated as per the equation $\Pi(t+1) = (1 - \alpha) \cdot \Pi(t) + \alpha \cdot X_{Best}$, where $\Pi(t)$ represents the probability vector for tth generation, α represents the learning factor, and X_{Best} is the best solution in the current population. But instead of using the best solution alone, the proposed algorithm uses all solutions in the current generation to update the probability vector. This is continued till the probability vector when rounded, becomes a valid solution [13]. Figure 3 depicts the pseudocode of population based incremental learning algorithm.

There are two decision parameters to be set, viz. (i) The value of the learning rate parameter (α) and (ii) the number of individuals used to update the vector. In the current algorithm, α is set to 0.4, and the number of individuals is set to half the population size.

3.3 Variable neighbourhood search

The Variable Neighbourhood Search (VNS) is an advancement over the iterated local search method. The VNS was introduced by Mladenovic and Hansen [75] for solving the travelling salesman problem. Since then, many researchers have

1. Initialize a probability vector $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ with $1/n$ at each position. Here, each π_i represents the probability of 1 for the i^{th} position in the solution.
2. Generate population Φ of p solutions by sampling probabilities in Π.
3. Select set Ψ from Φ consisting of q promising solutions, where $q <= p$.
4. Estimate univariate marginal probabilities $\pi(x_i)$ for each x_i.
5. For each i, update π_i using $\pi_i = \pi_i + \alpha\,(\pi(x_i) - \pi_i)$.
6. Go to step 2 until the termination criterion is met.

Fig. 3 Pseudocode of the population based incremental learning algorithm

adopted VNS to solve several different combinatorial optimization problems. The VNS works with a set of predefined neighbourhood structures of the problem under consideration. By systematically and sequentially exploring these neighbourhoods, VNS finds better solutions. It starts with a single solution known as the incumbent solution. A local search is then employed on the incumbent solution to explore one of its predefined neighbourhoods. The resulting solutions from the local search are compared with the incumbent solution and the best solution found so far becomes the incumbent solution. If the incumbent solution changes during the local search, the local search restarts with the new incumbent solution. If no better solution is found during the local search, the neighbourhood structure changes to the next neighbourhood structure and the local search on this new structure is done. Thus, by systematically and sequentially changing the neighbourhood and using local search, VNS directs the search in a promising direction to obtain better and better solutions [51]. Figure 4 represents the pseudocode for the general VNS algorithm.

The neighbourhood structures selected for implementations in the present research are insertion neighbourhood, swap neighbourhood and 3-permute

1. Select the set of neighbourhood structures n_k with $k = 1, \ldots, k_{max}$, that will be used in the search.
2. Provide an initial solution x.
3. Set $k := 1$
4. Until $k = k_{max}$, repeat the following steps:
   a. generate a point x' at random from the k^{th} neighbourhood of x.
   b. apply some local search method with x' as the initial solution; denote the obtained local optimum as x''.
   c. if the solution thus obtained is better than the incumbent solution, move there ($x := x''$), and continue the search with the current neighbourhood structure; otherwise, set $k := k + 1$.

Fig. 4 Pseudocode of variable neighbourhood search

neighbourhood. Since mutation is considered as the neighbourhood search procedure in GA, most of the neighbourhood structures for permutation problems are the result of mutation operations. The neighbourhood structures considered in this case are obtained through mutation operations on permutation representation of QAP. The three neighbourhood structures are explained in the following subsections.

3.4 Insertion neighbourhood

The insertion neighbourhood is based on insertion mutation. In insertion mutation, an element is randomly chosen from the parent string and is inserted into a randomly selected place [70]. For example, consider the parent string (1 2 3 4 5 6), and suppose that the randomly selected element is 3. Then, 3 is removed and inserted back into the string in a randomly selected position, say after 6. Hence, the resulting offspring is (1 2 4 5 6 3). The insertion neighbourhood is obtained by inserting all the elements in all the possible positions of the incumbent solution. Thus, in the above example, there are 30 members in the insertion neighbourhood. The pseudocode for obtaining insertion neighbourhood is shown in Fig. 5.

3.5 Swap neighbourhood

The swap neighbourhood is based on swap mutation operator. The swap mutation operator randomly selects two elements in the parent string and exchanges them [14]. For example, consider the parent solution string represented by (1 2 3 4 5 6), and suppose that the second and the fifth elements are randomly selected and swapped which results in the solution string as (1 5 3 4 2 6). The swap neighbourhood is obtained by selecting all the possible combinations of two elements from the incumbent solution and swapping the two selected elements. For the above example, there are 15 members in the swap neighbourhood. The pseudocode for swap neighbourhood is provided in Fig. 6.

```
INS_pseudo_code
{
    For each element Xi in position i of the solution do
    {
    Insert Xi in all possible positions other than i in the solution.
    }
}
```

Fig. 5 Pseudocode of insertion neighbourhood generation

```
SNS_pseudo_code
{
    For each position i in the solution
    For each position j >= i +1 in the solution do
    {
    Exchange or swap the elements in positions i and j.
    }
}
```

Fig. 6 Pseudocode of swap neighbourhood generation

3.6 3-Permute neighbourhood

The 3-permute neighbourhood is based on the 3-permute mutation operation. In 3-permute mutation, three consecutive elements are selected randomly and the permutations of these elements replace the initial three elements in the parent string. As a result of this process, there will be five new offsprings and the best among them is selected. For example, consider the parent string (1 2 3 4 5 6), and select the three consecutive members 1, 2, and 3. The permutations of these members are: (1 3 2), (2 1 3), (2 3 1), (3 1 2), and (3 2 1). Insert these sub-strings one by one to replace the three members 1, 2, and 3 to generate the neighbourhood members like (1 3 2 4 5 6), (2 1 3 4 5 6), (2 3 1 4 5 6), etc. The 3-permute neighbourhood is generated by selecting all possible combination of three consecutive members in the parent string. All the possible permutations are then inserted back one by one into the incumbent solution to form the 3-permute neighbourhood members. The pseudocode for swap neighbourhood is provided in Fig. 7.

```
3PNS_pseudo_code
{
    For i = 1 to (n-2)
    {
    P = Permute [X(i), X(i+1), X(i+2)];
    For j = 1 to 5
    {
        Replace [X(i), X(i+1), X(i+2)] in incumbent solution with P(j);
    }
    }
}
```

Fig. 7 Pseudocode of 3-permute neighbourhood generation

3.7 The proposed algorithm

The PBILA employed here is the same as that described in Sect. 3.2. The pseudocode for the proposed PBILA-VNS is shown in Fig. 8. Here, all the members of the population undergo the VNS procedure before the elite solutions are selected and thus VNS intensifies the search procedure.

3.7.1 Illustration of the proposed PBILA-VNS algorithm

Initialize a probability matrix Π as $n \times n$ matrix with all values set to $1/n$. Here, each (i, j) element in the matrix represents the probability that ith department is located in jth location.

$$\Pi = \begin{bmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$$

Generate population $\Phi = \begin{Bmatrix} 5 & 2 & 1 & 4 & 6 & 3 \\ 2 & 3 & 5 & 6 & 1 & 4 \\ & & & \vdots & & \\ 3 & 5 & 6 & 2 & 1 & 4 \end{Bmatrix}$ of p solutions by sampling probabilities in Π.

Run VNS on each of the p solutions.

$$\Phi = \begin{Bmatrix} 3 & 1 & 2 & 4 & 6 & 5 \\ 4 & 6 & 5 & 3 & 1 & 2 \\ & & & \vdots & & \\ 6 & 5 & 2 & 4 & 1 & 3 \end{Bmatrix}$$

Calculate the objective function value.

1. Initialize a probability vector $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ with $1/n$ at each position. Here, each π_i represents the probability of 1 for the i^{th} position in the solution.
2. Generate population Φ of p solutions by sampling probabilities in Π.
3. Apply VNS on each of the members in the population.
4. Select set Ψ from Φ consisting of q promising solutions, where $q <= p$.
5. Estimate univariate marginal probabilities $\pi(x_i)$ for each x_i.
6. For each i, update π_i using $\pi_i = \pi_i + \alpha\,(\pi(x_i) - \pi_i)$
7. Go to step 2 until the termination criterion is met.

Fig. 8 Pseudocode of the PBILA-VNS algorithm

$$\Omega = \begin{Bmatrix} 20253 \\ 20253 \\ \vdots \\ 20361 \end{Bmatrix}$$

Select set Ψ from Φ consisting of q promising solutions, where $q \leq p$.

$$\Psi = \begin{Bmatrix} 1 & 3 & 2 & 6 & 4 & 5 \\ 4 & 6 & 5 & 3 & 1 & 2 \\ & & & \vdots & & \\ 6 & 4 & 5 & 1 & 3 & 2 \end{Bmatrix}$$

Estimate univariate marginal probabilities $\pi'_{(i,j)}$ for each (i, j) using Ψ.

For calculating univariate marginal probabilities, convert each solution in Ψ to the corresponding permutation matrix. For example,

$$[1 \quad 3 \quad 2 \quad 6 \quad 4 \quad 5] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Add the permutation matrices and divide it by n to obtain Π'

$$\text{i.e., } \Pi' = \frac{1}{12}\begin{bmatrix} 2 & 0 & 5 & 2 & 0 & 3 \\ 5 & 0 & 2 & 3 & 0 & 2 \\ 0 & 7 & 0 & 0 & 5 & 0 \\ 3 & 0 & 2 & 5 & 0 & 2 \\ 2 & 0 & 3 & 2 & 0 & 5 \\ 0 & 5 & 0 & 0 & 7 & 0 \end{bmatrix}$$

For each (i, j), update Π using $\pi_{(i,j)} = \pi_{(i,j)} + \alpha\,(\pi'_{(i,j)} - \pi_{(i,j)})$.

$$\Pi = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} + 0.4\left(\begin{bmatrix} \frac{1}{6} & 0 & \frac{5}{12} & \frac{1}{6} & 0 & \frac{1}{4} \\ \frac{5}{12} & 0 & \frac{1}{6} & \frac{1}{4} & 0 & \frac{1}{6} \\ 0 & \frac{7}{12} & 0 & 0 & \frac{5}{12} & 0 \\ \frac{1}{4} & 0 & \frac{1}{6} & \frac{5}{12} & 0 & \frac{1}{6} \\ \frac{1}{6} & 0 & \frac{1}{4} & \frac{1}{6} & 0 & \frac{5}{12} \\ 0 & \frac{5}{12} & 0 & 0 & \frac{7}{12} & 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}\right) = \begin{bmatrix} \frac{1}{6} & \frac{1}{10} & \frac{4}{15} & \frac{1}{6} & \frac{1}{10} & \frac{1}{5} \\ \frac{4}{15} & \frac{1}{10} & \frac{1}{6} & \frac{1}{5} & \frac{1}{10} & \frac{1}{6} \\ \frac{1}{10} & \frac{1}{3} & \frac{1}{10} & \frac{1}{10} & \frac{4}{15} & \frac{1}{10} \\ \frac{1}{5} & \frac{1}{10} & \frac{1}{6} & \frac{4}{15} & \frac{1}{10} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{10} & \frac{1}{5} & \frac{1}{6} & \frac{1}{10} & \frac{4}{15} \\ \frac{1}{10} & \frac{4}{15} & \frac{1}{10} & \frac{1}{10} & \frac{1}{3} & \frac{1}{10} \end{bmatrix}$$

Repeat steps 2–5 until the termination criterion is satisfied. The termination criterion used is the number of iterations and its value is $10 \times$ number of facilities. Save the solution with the lowest cost. There are multiple solutions for the above problem with a total cost of 20,253 and one of them is (1 3 2 5 6 4).

4 Computational study

The QAP library, QAPLIB (http://anjos.mgi.polymtl.ca/qaplib/ or https://www.opt.math.tugraz.at/qaplib/) provides a number of benchmark instances of QAPs presented by Burkard et al. [24]. The test instances can be categorised into four depending on the type of instances they represent. The four types are as follows:

- Type I—Unstructured and randomly generated instances
- Type II—Instances with grid distances
- Type III—Real-life instances
- Type IV—Real-life like instances.

Out of the total 134 problems provided in the QAP library, 101 problems which include the QAPs of size up to 50 are considered in the present research.

The algorithms are coded in MATLAB and the programme is run in PCs with Intel Core i3 Processor with 4 GB RAM and running on windows 7. Each problem is solved ten times and the best, worst and average results are reported. The percentage deviation from the known best solution is also calculated. The results are given in Tables 1, 2, 3 and 4.

Tables 1, 2, 3 and 4 provide the results obtained while solving the QAP benchmark instances of category Type I to Type IV respectively using the PBILA and PBILA-VNS algorithms.

Out of the total 101 test instances solved PBILA is able to obtain BKS for only six problems whereas PBILA-VNS is able to obtain BKS for 95 problems. For 49 test instances the PBILA-VNS is able to get the known best solution in all the ten trials. Only for a single problem the minimum solution obtained showed a deviation of more than one percent. From this comparison itself, it is evident that the hybridization of PBILA with VNS improves the efficiency of PBILA. Out of the six problems for which the best known solution was not obtained, five belong to the Type I category and one to the Type II category. Moreover, the PBILA-VNS is able to provide best known solution for all the real-life instances (Type III category) and real-life like instances (Type IV category).

4.1 Comparison with state-of-the-art algorithms

In order to compare the performance of the proposed PBILA-VNS algorithm, it is compared with some of the recent state-of-the-art algorithms published in the literature. The algorithms considered for the comparative analysis are:

1. Backtracking Search Algorithm (BSA) [108]
2. BSA with Iterated Local Search (BSAL) [108]
3. MultiStart Hyper-heuristic Algorithm (MSH-QAP) [35]
4. Great deluge and tabu search hybrid with two-stage memory support (TMSGD-QAP) [1]

Table 1 Solution obtained and corresponding percentage deviations for Type I problem instances

Sl. no.	Problem	Size	BKS	PBILA						PBILA-VNS					
				Solution			% Deviation			Solution			% Deviation		
				B	W	A	B	W	A	B	W	A	B	W	A
1	lipa20a	20	3683	3815	3861	3838.9	3.6	4.8	4.2	3683	3683	3683	0.0	0.0	0.0
2	lipa20b	20	27,076	32,006	33,027	32,670.9	18.2	22.0	20.7	27,076	27,076	27,076	0.0	0.0	0.0
3	lipa30a	30	13,178	13,548	13,622	13,590.9	2.8	3.4	3.1	13,178	13,178	13,178	0.0	0.0	0.0
4	lipa30b	30	151,426	182,295	184,973	183,644.9	20.4	22.2	21.3	151,426	151,426	151,426	0.0	0.0	0.0
5	lipa40a	40	31,538	32,135	32,329	32,254.5	1.9	2.5	2.3	31,538	31,817	31,565.9	0.0	0.9	0.1
6	lipa40b	40	476,581	582,114	592,100	588,698.2	22.1	24.2	23.5	476,581	476,581	476,581	0.0	0.0	0.0
7	lipa50a	50	62,093	63,242	63,432	63,364.0	1.9	2.2	2.0	62,093	62,093	62,093	0.0	0.0	0.0
8	lipa50b	50	1,210,244	1,476,913	1,494,434	1,485,999.4	22.0	23.5	22.8	1,210,244	1,210,244	1,210,244	0.0	0.0	0.0
9	rou12	12	235,528	254,090	270,908	262,914.6	7.9	15.0	11.6	235,528	238,134	235,885.8	0.0	1.1	0.2
10	rou15	15	354,210	392,664	415,942	402,440.8	10.9	17.4	13.6	354,210	358,912	354,946.6	0.0	1.3	0.2
11	rou20	20	725,522	773,252	829,576	802,947.4	6.6	14.3	10.7	725,582	728,020	726,780.6	0.0	0.3	0.2
12	tai12a	12	224,416	251,860	267,256	261,676.6	12.2	19.1	16.6	224,416	230,704	225,044.8	0.0	2.8	0.3
13	tai15a	15	388,214	417,828	441,944	430,003.0	7.6	13.8	10.8	388,214	388,988	388,295	0.0	0.2	0.0
14	tai17a	17	491,812	543,982	571,250	553,169.8	10.6	16.2	12.5	491,812	497,726	492,773.4	0.0	1.2	0.2
15	tai20a	20	703,482	771,084	804,916	786,898.4	9.6	14.4	11.9	705,622	710,926	709,373.8	0.3	1.1	0.8
16	tai25a	25	1,167,256	1,282,084	1,333,868	1,304,651.4	9.8	14.3	11.8	1,171,944	1,183,400	1,177,396.2	0.4	1.4	0.9
17	tai30a	30	1,818,146	1,960,568	2,026,832	2,007,077.2	7.8	11.5	10.4	1,818,146	1,829,626	1,825,036	0.0	0.6	0.4
18	tai35a	35	2,422,002	2,649,092	2,697,886	2,671,287.2	9.4	11.4	10.3	2,432,854	2,447,870	2,438,885.8	0.4	1.1	0.7
19	tai40a	40	3,139,370	3,435,252	3,518,274	3,480,794.6	9.4	12.1	10.9	3,159,982	3,181,492	3,163,478	0.7	1.3	0.8
20	tai50a	50	4,938,796	5,372,340	5,520,004	5,438,068.0	8.8	11.8	10.1	5,032,296	5,035,748	5,033,772	1.9	2.0	1.9

BKS best known solution; *B* best; *W* worst; *A* average

Table 2 Solution obtained and corresponding percentage deviations for Type II problem instances

Sl. no.	Problem	Size	BKS	PBILA						PBILA-VNS					
				Solution			% Deviation			Solution			% Deviation		
				B	W	A	B	W	A	B	W	A	B	W	A
21	nug12	12	578	626	670	652.8	8.3	15.9	12.9	578	586	578.8	0.0	1.4	0.1
22	nug14	14	1014	1086	1162	1122.4	7.1	14.6	10.7	1014	1016	1015	0.0	0.2	0.1
23	nug15	15	1150	1296	1372	1326.6	12.7	19.3	15.4	1150	1152	1150.2	0.0	0.2	0.0
24	nug16a	16	1610	1752	1864	1803.4	8.8	15.8	12.0	1610	1612	1610.2	0.0	0.1	0.0
25	nug16b	16	1240	1374	1460	1412.6	10.8	17.7	13.9	1240	1240	1240	0.0	0.0	0.0
26	nug17	17	1732	1898	1984	1934.0	9.6	14.5	11.7	1732	1734	1733	0.0	0.1	0.1
27	nug18	18	1930	2102	2206	2165.6	8.9	14.3	12.2	1930	1944	1935.2	0.0	0.7	0.3
28	nug20	20	2570	2872	2976	2911.0	11.8	15.8	13.3	2570	2570	2570	0.0	0.0	0.0
29	nug21	21	2438	2688	2886	2804.6	10.3	18.4	15.0	2438	2446	2440.2	0.0	0.3	0.1
30	nug22	22	3596	4044	4276	4132.4	12.5	18.9	14.9	3596	3596	3596	0.0	0.0	0.0
31	nug24	24	3488	3908	4204	3991.0	12.0	20.5	14.4	3488	3490	3488.4	0.0	0.1	0.0
32	nug25	25	3744	4172	4390	4257.8	11.4	17.3	13.7	3744	3750	3744.6	0.0	0.2	0.0
33	nug27	27	5234	5710	6152	5929.0	9.1	17.5	13.3	5234	5234	5234	0.0	0.0	0.0
34	nug28	28	5166	5656	6016	5832.0	9.5	16.5	12.9	5166	5204	5182	0.0	0.7	0.3
35	nug30	30	6124	6742	7130	6990.4	10.1	16.4	14.1	6124	6156	6135.4	0.0	0.5	0.2
36	scr12	12	31,410	34,578	42,656	37,602.8	10.1	35.8	19.7	31,410	31,410	31,410	0.0	0.0	0.0
37	scr15	15	51,140	60,346	73,974	67,756.6	18.0	44.6	32.5	51,140	51,140	51,140	0.0	0.0	0.0
38	scr20	20	110,030	135,434	163,718	148,607.2	23.1	48.8	35.1	110,030	110,030	110,030	0.0	0.0	0.0
39	sko42	42	15,812	17,336	17,940	17,668.4	9.6	13.5	11.7	15,812	15,852	15,828.6	0.0	0.3	0.1
40	sko49	49	23,386	25,728	26,686	26,193.6	10.0	14.1	12.0	23,412	23,480	23,439.6	0.1	0.4	0.2
41	tho30	30	149,936	170,496	179,308	175,105.4	13.7	19.6	16.8	149,936	150,454	150,032	0.0	0.3	0.1
42	tho40	40	240,516	275,482	291,072	282,404.2	14.5	21.0	17.4	240,516	241,190	240,709	0.0	0.3	0.1
43	wil50	50	48,816	51,476	52,008	51,715.0	5.4	6.5	5.9	48,816	48,850	48,838.2	0.0	0.1	0.0

BKS best known solution; *B* best; *W* worst; *A* average

5. Parallel Hybrid Algorithm (PHA) [95]
6. Fully Informed Parallel Genetic Algorithm (QAP-IPGA) [96]
7. Biogeography-Based Optimization Algorithm hybridized with Tabu Search (BBOTS) [65]
8. Breakout Local Search Algorithm using OpenMP (BLS-OpenMP) [8]
9. Graphics Processing Unit Algorithm using Level-2 Reformulation–Linearization Technique (GPU-QAP) [48]
10. Improved Hunting Search Algorithm (IHuS) [3]
11. Eight variants of Evolutionary Algorithm Using Conditional Expectation Value (PMX, PMX2, OX, OXI, POX, POX2, MUT and MUT1) [28]
12. Parallel Water Flow Algorithm with Local Search (WFA) [76].

The performance of PBILA-VNS is compared with that of algorithms available in literature and the results are provided in Table 5, 6, 7 and 8.

Table 5 provides the comparison of the algorithms with respect to their ability to solve the Type I QAP instances. MSH-QAP proposed by Dokeroglu and Cosar [35] is the only algorithm which provides solutions to all the Type I test instances considered for the study and the algorithm is able to provide BKS for 18 test instances and the PBILA-VNS is able to provide BKS for 15 test instances out of the 20 test instances considered. Thus, MHS-QAP outperforms the PBILA-VNS it terms of their ability to obtain the BKS for Type I QAP instances. The average percentage deviation of the obtained solutions from the best-known solutions of Type I test instances is 0.19%.

Table 6 provides the comparison of PBILA-VNS with other algorithms available in literature, in terms of their ability to obtain optimal solutions for test instances falling under the category of Type II. Only TMSGD-QAP proposed by Acan and Ünveren [1] reports percentage deviations from BKS for all the 23 test instances considered in this study. PBILA-VNS and TMSGD-QAP report the same percentage deviation for 22 test instances and for a single test instance the PBILA-VNS fails to obtain the BKS while the TMSGD-QAP is successful. The average percentage deviation of the obtained solutions from the best-known solutions of Type II test instances is 0.01%.

The performance comparison of PBILA-VNS with other algorithms in the literature in terms of solving the Type III category of test instances is presented in Table 7. Fifty test instances of category Type III are considered in this study and PBILA-VNS is able to obtain the BKS for all the test instances. The algorithms MHS-QAP, TMSGD-QAP and WFA reported the BKS for all the 50 test instances and thus PBILA-VNS performs equally with these algorithms. None of the other algorithms reported in the literature report the solution for all the 50 test instances of Type III considered in this study and thus it can be deduced that PBILA-VNS is performing better than other algorithms.

Table 8 depicts the comparative performance of EDA-VNS with other algorithms available in the literature in terms of solving Type IV test instances. For all the eight test instances considered, EDA-VNS is able to obtain the BKS. The MHS-QAP and WFA are the only other algorithm which report solutions for all the test instances and both are able to obtain the BKS for all the test instances under consideration.

Table 3 Solution obtained and corresponding percentage deviations for Type III problem instances

Sl. no.	Problem	Size	BKS	PBILA						PBILA-VNS					
				Solution			% Deviation			Solution			% Deviation		
				B	W	A	B	W	A	B	W	A	B	W	A
44	bur26a	26	5,426,670	5500653	5,575,377	5,555,694.9	1.4	2.7	2.4	5,426,670	5,427,776	5,426,891.2	0.0	0.0	0.0
45	bur26b	26	3,817,852	3,853,662	3,910,260	3,881,881.5	0.9	2.4	1.7	3,817,852	3,817,952	3,817,872	0.0	0.0	0.0
46	bur26c	26	5,426,795	5,506,416	5,616,186	5,566,393.8	1.5	3.5	2.6	5,426,795	5,426,795	5,426,795	0.0	0.0	0.0
47	bur26d	26	3,821,225	3,866,369	3,920,388	3,894,423.0	1.2	2.6	1.9	3,821,225	3,821,225	3,821,225	0.0	0.0	0.0
48	bur26e	26	5,386,879	5,485,158	5,572,019	5,515,427.8	1.8	3.4	2.4	5,386,879	5,386,879	5,386,879	0.0	0.0	0.0
49	bur26f	26	3,782,044	3,825,741	3,917,313	3,870,433.1	1.2	3.6	2.3	3,782,044	3,782,044	3,782,044	0.0	0.0	0.0
50	bur26g	26	10,117,172	10,281,336	10,440,437	10,368,383.3	1.6	3.2	2.5	10,117,172	10,117,172	10,117,172	0.0	0.0	0.0
51	bur26h	26	7,098,658	7,184,824	7,297,772	7,261,791.6	1.2	2.8	2.3	7,098,658	7,098,658	7,098,658	0.0	0.0	0.0
52	chr12a	12	9552	18,062	24,086	20,190.0	89.1	152.2	111.4	9552	9916	9588.4	0.0	3.8	0.4
53	chr12b	12	9742	13,700	25,248	17,997.6	40.6	159.2	84.7	9742	9742	9742	0.0	0.0	0.0
54	chr12c	12	11,156	16,584	22,086	19,218.8	48.7	98.0	72.3	11,156	11,566	11,263.6	0.0	3.7	1.0
55	chr15a	15	9896	18,884	26,300	22,750.4	90.8	165.8	129.9	9896	9896	9896	0.0	0.0	0.0
56	chr15b	15	7990	18,664	27,050	21,720.0	133.6	238.5	171.8	7990	8640	8273.4	0.0	8.1	3.5
57	chr15c	15	9504	20,970	31,122	24,948.2	120.6	227.5	162.5	9504	10,446	9659.6	0.0	9.9	1.6
58	chr18a	18	11,098	26,054	38,096	33,569.4	134.8	243.3	202.5	11,098	11,608	11,170.6	0.0	4.6	0.7
59	chr18b	18	1534	2034	2448	2307.0	32.6	59.6	50.4	1534	1534	1534	0.0	0.0	0.0
60	chr20a	20	2192	3876	5708	4816.4	76.8	160.4	119.7	2192	2392	2251.2	0.0	9.1	2.7
61	chr20b	20	2298	4102	5264	4643.6	78.5	129.1	102.1	2298	2444	2405.8	0.0	6.4	4.7
62	chr20c	20	14,142	30,264	48,992	42,682.2	114.0	246.4	201.8	14,142	14,996	14,494.8	0.0	6.0	2.5
63	chr22a	22	6156	7932	9980	8490.4	28.8	62.1	37.9	6156	6274	6205	0.0	1.9	0.8
64	chr22b	22	6194	8052	9480	8697.0	30.0	53.1	40.4	6194	6256	6233.2	0.0	1.0	0.6
65	chr25a	25	3796	8722	12,474	10,080.6	129.8	228.6	165.6	3796	4152	3897.6	0.0	9.4	2.7
66	els19	19	17,212,548	23,938,022	31,877,712	28,327,719.4	39.1	85.2	64.6	17,212,548	17,212,548	17,212,548	0.0	0.0	0.0
67	esc16a	16	68	68	86	81.2	0.0	26.5	19.4	68	68	68	0.0	0.0	0.0

Table 3 (continued)

Sl. no.	Problem	Size	BKS	PBILA						PBILA-VNS					
				Solution			% Deviation			Solution			% Deviation		
				B	W	A	B	W	A	B	W	A	B	W	A
68	esc16b	16	292	292	296	293.6	0.0	1.4	0.5	292	292	292	0.0	0.0	0.0
69	esc16c	16	160	168	184	173.6	5.0	15.0	8.5	160	160	160	0.0	0.0	0.0
70	esc16d	16	16	18	32	22.2	12.5	100.0	38.8	16	16	16	0.0	0.0	0.0
71	esc16e	16	28	30	38	33.8	7.1	35.7	20.7	28	28	28	0.0	0.0	0.0
72	esc16g	16	26	28	40	34.0	7.7	53.8	30.8	26	26	26	0.0	0.0	0.0
73	esc16h	16	996	996	1076	1034.0	0.0	8.0	3.8	996	996	996	0.0	0.0	0.0
74	esc16i	16	14	14	26	20.8	0.0	85.7	48.6	14	14	14	0.0	0.0	0.0
75	esc16j	16	8	8	14	10.4	0.0	75.0	30.0	8	8	8	0.0	0.0	0.0
76	esc32a	32	130	216	286	241.0	66.2	120.0	85.4	130	134	133	0.0	3.1	2.3
77	esc32b	32	168	312	348	324.8	85.7	107.1	93.3	168	192	170.4	0.0	14.3	1.4
78	esc32c	32	642	692	768	723.2	7.8	19.6	12.6	642	642	642	0.0	0.0	0.0
79	esc32d	32	200	248	274	259.4	24.0	37.0	29.7	200	200	200	0.0	0.0	0.0
80	esc32e	32	2	2	6	2.8	0.0	200.0	40.0	2	2	2	0.0	0.0	0.0
81	esc32g	32	6	8	14	10.0	33.3	133.3	66.7	6	6	6	0.0	0.0	0.0
82	esc32h	32	438	492	528	512.6	12.3	20.5	17.0	438	438	438	0.0	0.0	0.0
83	had12	12	1652	1692	1748	1715.0	2.4	5.8	3.8	1652	1652	1652	0.0	0.0	0.0
84	had14	14	2724	2770	2928	2848.2	1.7	7.5	4.6	2724	2724	2724	0.0	0.0	0.0
85	had16	16	3720	3810	3988	3887.0	2.4	7.2	4.5	3720	3720	3720	0.0	0.0	0.0
86	had18	18	5358	5538	5620	5589.0	3.4	4.9	4.3	5358	5358	5358	0.0	0.0	0.0
87	had20	20	6922	7218	7340	7281.6	4.3	6.0	5.2	6922	6922	6922	0.0	0.0	0.0
88	kra30a	30	88,900	107,450	112,050	109,256.0	20.9	26.0	22.9	88,900	90,100	89,260	0.0	1.3	0.4
89	kra30b	30	91,420	105,840	115,330	111,049.0	15.8	26.2	21.5	91,420	91,540	91,474	0.0	0.1	0.1

Table 3 (continued)

Sl. no.	Problem	Size	BKS	PBILA						PBILA-VNS					
				Solution			% Deviation			Solution			% Deviation		
				B	W	A	B	W	A	B	W	A	B	W	A
90	kra32	32	88,700	104,410	113,320	108,387.0	17.7	27.8	22.2	88,700	89,700	88,800	0.0	1.1	0.1
91	ste36a	36	9526	12,602	15,384	14,090.8	32.3	61.5	47.9	9526	9612	9565.2	0.0	0.9	0.4
92	ste36b	36	15,852	30,516	37,086	33,806.2	92.5	134.0	113.3	15,852	15,852	15,852	0.0	0.0	0.0
93	ste36c	36	8,239,110	10,606,824	12,189,492	11,678,257.0	28.7	47.9	41.7	8,239,110	8,269,034	8,245,766.8	0.0	0.4	0.1

BKS best known solution; *B* best; *W* worst; *A* average

Table 4 Solution obtained and corresponding percentage deviations for Type IV problem instances

Sl. no.	Problem	Size	BKS	PBILA						PBILA-VNS					
				Solution			% Deviation			Solution			% Deviation		
				B	W	A	B	W	A	B	W	A	B	W	A
94	tai12b	12	39,464,925	45,297,012	55,173,145	48,161,107.9	14.8	39.8	22.0	39,464,925	39,464,925	39,464,925	0.0	0.0	0.0
95	tai15b	15	51,765,268	52,559,671	53,411,263	52,966,235.3	1.5	3.2	2.3	51,765,268	51,765,268	51,765,268	0.0	0.0	0.0
96	tai20b	20	122,455,319	135,644,981	178,716,451	158,510,524.9	10.8	45.9	29.4	122,455,319	122,455,319	122,455,319	0.0	0.0	0.0
97	tai25b	25	344,355,646	461,940,625	532,771,309	499,084,082.5	34.1	54.7	44.9	344,355,646	344,355,646	344,355,646	0.0	0.0	0.0
98	tai30b	30	637,117,113	734,538,273	900,040,886	822,627,533.6	15.3	41.3	29.1	637,117,113	640,340,773	637,901,372.8	0.0	0.5	0.1
99	tai35b	35	283,315,445	333,247,207	376,614,691	357,696,807.8	17.6	32.9	26.3	283,315,445	283,315,445	283,315,445	0.0	0.0	0.0
100	tai40b	40	637,250,948	772,351,791	858,693,240	821,472,900.4	21.2	34.7	28.9	637,250,948	639,342,035	637,702,963.1	0.0	0.3	0.1
101	tai50b	50	458,821,517	578,294,820	637,636,141	608,868,753.6	26.0	39.0	32.7	458,897,442	460,726,849	460,170,733.4	0.0	0.4	0.3

BKS best known solution; *B* best; *W* worst; *A* average

Table 5 Comparison of PBILA-VNS with state-of-the-art algorithms on Type I problem instances

Sl. no.	Problem	Size	BKS	PBILA-VNS	BSA	BSAL	MSH-QAP	TMSGD-QAP	PHA	QAP-IPGA	IHuS	PMX	PMX2	OX	OXI	POX	POX2	MUT	MUT1	WFA
1	lipa20a	20	3683	**0.0**	1.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
2	lipa20b	20	27,076	**0.0**	15.3	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	–	–	**0.0**
3	lipa30a	30	13,178	**0.0**	0.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.3	0.9	1.2	0.4	0.5	**0.0**	0.3	0.4	**0.0**
4	lipa30b	30	151,426	**0.0**	1.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
5	lipa40a	40	31,538	**0.0**	1.2	**0.0**	**0.0**	**0.0**	**0.0**	0.5	0.8	1.2	1.1	1.3	1.3	1.3	1.1	1.3	1.3	**0.0**
6	lipa40b	40	476,581	**0.0**	1.4	**0.0**	**0.0**	**0.0**	–	–	–	4.2	**0.0**	5.2	3.4	4.2	**0.0**	4.0	3.0	**0.0**
7	lipa50a	50	62,093	**0.0**	1.3	0.7	**0.0**	**0.0**	**0.0**	0.8	–	1.2	1.0	1.3	1.3	1.3	1.3	1.2	1.3	0.8
8	lipa50b	50	1,210,244	**0.0**	17.6	**0.0**	**0.0**	**0.0**	–	–	–	6.0	**0.0**	9.3	7.2	10.4	**0.0**	9.3	5.8	**0.0**
9	rou12	12	235,528	**0.0**	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
10	rou15	15	354,210	**0.0**	0.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
11	rou20	20	725,522	**0.0**	0.0	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	–	–	**0.0**
12	tai12a	12	224,416	**0.0**	2.8	**0.0**	**0.0**	–	**0.0**	–	**0.0**	–	–	–	–	–	–	–	–	**0.0**
13	tai15a	15	388,214	**0.0**	1.2	**0.0**	**0.0**	–	–	–	**0.0**	–	–	–	–	–	–	–	–	**0.0**
14	tai17a	17	491,812	**0.0**	2.5	**0.0**	**0.0**	–	–	–	0.4	–	–	–	–	–	–	–	–	**0.0**
15	tai20a	20	703,482	0.3	1.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.9	–	–	–	–	–	–	–	–	**0.0**
16	tai25a	25	1,167,256	0.4	–	–	**0.0**	**0.0**	–	**0.0**	1.7	–	–	–	–	–	–	–	–	**0.0**
17	tai30a	30	1,818,146	**0.0**	–	–	**0.0**	**0.0**	**0.0**	0.6	1.8	–	–	–	–	–	–	–	–	0.6
18	tai35a	35	2,422,002	0.4	–	–	**0.0**	**0.0**	**0.0**	0.8	2.3	–	–	–	–	–	–	–	–	0.6
19	tai40a	40	3,139,370	0.7	3.4	1.7	0.3	0.1	**0.0**	0.7	–	–	–	–	–	–	–	–	–	0.7
20	tai50a	50	4,938,796	1.9	–	–	0.2	0.2	–	–	–	–	–	–	–	–	–	–	–	1.5
Average				0.19	3.26	0.15	0.02	0.02	**0.00**	0.29	0.88	2.15	0.49	3.05	2.26	2.94	0.39	2.70	1.95	0.20

Bold-faced zero indicates that the algorithm has provided the Best Known Solution

BKS best known solution

Table 6 Comparison of PBILA-VNS with state-of-the-art algorithms on Type II problem instances

Sl. no.	Problem	Size	BKS	PBILA-VNS	BSA	BSAL	MSH-QAP	BBOTS	TMSGD-QAP	PHA	QAP-IPGA	IHuS	PMX	PMX2	OX	OXI	POX	POX2	MUT	WFA
21	nug12	12	578	**0.0**	2.1	**0.0**	–	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
22	nug14	14	1014	**0.0**	0.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
23	nug15	15	1150	**0.0**	3.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
24	nug16a	16	1610	**0.0**	2.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
25	nug16b	16	1240	**0.0**	3.9	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	–	**0.0**
26	nug17	17	1732	**0.0**	0.7	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
27	nug18	18	1930	**0.0**	1.7	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
28	nug20	20	2570	**0.0**	0.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
29	nug21	21	2438	**0.0**	2.6	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	–	**0.0**
30	nug22	22	3596	**0.0**	2.7	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
31	nug24	24	3488	**0.0**	2.2	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
32	nug25	25	3744	**0.0**	4.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
33	nug27	27	5234	**0.0**	2.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
34	nug28	28	5166	**0.0**	3.9	**0.0**	**0.0**	0.2	**0.0**	–	–	–	–	–	–	–	–	–	–	**0.0**
35	nug30	30	6124	**0.0**	2.4	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
36	scr12	12	31,410	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	–	**0.0**
37	scr15	15	51,140	**0.0**	3.8	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	–	**0.0**
38	scr20	20	110,030	**0.0**	2.0	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	–	**0.0**
39	sko42	42	15,812	**0.0**	4.8	0.4	**0.0**	**0.0**	**0.0**	**0.0**	0.5	1.7	0.9	1.4	1.7	1.6	0.0	1.1	1.8	**0.0**
40	sko49	49	23,386	0.1	1.6	0.5	**0.0**	–	**0.0**	**0.0**	0.7	2.0	0.9	2.0	1.7	2.1	0.8	2.2	1.7	0.3
41	tho30	30	149,936	**0.0**	–	–	**0.0**	–	**0.0**	**0.0**	**0.0**	0.8	0.3	0.8	0.5	0.3	0.8	0.6	0.6	**0.0**
42	tho40	40	240,516	**0.0**	–	–	**0.0**	–	**0.0**	–	–	2.4	1.2	2.1	1.7	2.5	1.3	2.2	2.4	**0.0**
43	wil50	50	48,816	**0.0**	–	–	**0.0**	0.1	**0.0**	**0.0**	0.4	0.9	0.2	0.9	1.0	1.0	0.3	0.9	0.9	0.1
Average				0.01	2.39	0.04	**0.00**	0.02	**0.00**	**0.00**	0.10	1.53	0.70	1.45	1.31	1.51	0.63	1.40	1.48	0.02

Bold-faced zero indicates that the algorithm has provided the Best Known Solution

BKS best known solution

Table 7 Comparison of PBILA-VNS with state-of-the-art algorithms on Type III problem instances

Sl. no.	Problem	Size	BKS	PBILA-VNS	BSA	BSAL	MSH-QAP	BBOTS	TMSGD-QAP	IHuS	PMX	PMX2	OX	OXI	POX	POX2	MUT	MUT1	WFA
44	bur26a	26	5,426,670	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
45	bur26b	26	3,817,852	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
46	bur26c	26	5,426,795	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
47	bur26d	26	3,821,225	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
48	bur26e	26	5,386,879	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
49	bur26f	26	3,782,044	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
50	bur26g	26	10,117,172	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
51	bur26h	26	7,098,658	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
52	chr12a	12	9552	**0.0**	0.1	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
53	chr12b	12	9742	**0.0**	0.2	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
54	chr12c	12	11,156	**0.0**	0.6	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
55	chr15a	15	9896	**0.0**	0.8	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
56	chr15b	15	7990	**0.0**	5.6	**0.0**	**0.0**	0.3	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
57	chr15c	15	9504	**0.0**	4.6	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
58	chr18a	18	11,098	**0.0**	4.4	**0.0**	**0.0**	0.1	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
59	chr18b	18	1534	**0.0**	0.3	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
60	chr20a	20	2192	**0.0**	6.2	**0.0**	**0.0**	0.9	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
61	chr20b	20	2298	**0.0**	4.6	**0.0**	**0.0**	–	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
62	chr20c	20	14,142	**0.0**	0.0	**0.0**	**0.0**	0.6	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
63	chr22a	22	6156	**0.0**	0.7	**0.0**	**0.0**	–	**0.0**	–	1.1	2.4	2.1	0.4	1.1	4.0	2.1	0.4	**0.0**
64	chr22b	22	6194	**0.0**	16.2	**0.0**	**0.0**	–	**0.0**	–	2.1	3.6	2.9	2.6	1.8	2.4	1.8	2.4	**0.0**
65	chr25a	25	3796	**0.0**	8.3	**0.0**	**0.0**	–	**0.0**	10.7	–	–	–	–	–	–	–	–	**0.0**
66	els19	19	17,212,548	**0.0**	0.9	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
67	esc16a	16	68	**0.0**	0.3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
68	esc16b	16	292	**0.0**	8.9	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**

Table 7 (continued)

Sl. no.	Problem	Size	BKS	PBILA-VNS	BSA	BSAL	MSH-QAP	BBOTS	TMSGD-QAP	IHuS	PMX	PMX2	OX	OXI	POX	POX2	MUT	MUT1	WFA
69	esc16c	16	160	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
70	esc16d	16	16	**0.0**	12.5	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
71	esc16e	16	28	**0.0**	12.5	**0.0**	**0.0**	–	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
72	esc16g	16	26	**0.0**	7.7	**0.0**	**0.0**	–	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
73	esc16h	16	996	**0.0**	**0.0**	**0.0**	**0.0**	–	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
74	esc16i	16	14	**0.0**	**0.0**	**0.0**	**0.0**	–	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
75	esc16j	16	8	**0.0**	25.0	**0.0**	**0.0**	–	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
76	esc32a	32	130	**0.0**	21.5	**0.0**	**0.0**	–	**0.0**	1.6	1.5	1.5	3.1	3.1	1.5	**0.0**	**0.0**	1.5	**0.0**
77	esc32b	32	168	**0.0**	11.9	**0.0**	**0.0**	–	**0.0**	–	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	**0.0**
78	esc32c	32	642	**0.0**	0.6	**0.0**	**0.0**	–	**0.0**	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
79	esc32d	32	200	**0.0**	3.0	**0.0**	**0.0**	–	**0.0**	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
80	esc32e	32	2	**0.0**	**0.0**	**0.0**	**0.0**	–	**0.0**	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
81	esc32g	32	6	**0.0**	**0.0**	**0.0**	**0.0**	–	**0.0**	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
82	esc32h	32	438	**0.0**	1.8	**0.0**	**0.0**	–	**0.0**	–	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
83	had12	12	1652	**0.0**	1.5	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
84	had14	14	2724	**0.0**	0.2	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
85	had16	16	3720	**0.0**	0.2	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
86	had18	18	5358	**0.0**	0.1	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
87	had20	20	6922	**0.0**	2.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	–	–	–	–	–	–	–	–	**0.0**
88	kra30a	30	88,900	**0.0**	3.9	**0.0**	**0.0**	0.1	**0.0**	0.4	0.7	2.1	1.4	0.4	1.9	1.4	2.1	0.5	**0.0**
89	kra30b	30	91,420	**0.0**	0.6	**0.0**	**0.0**	0.1	**0.0**	0.2	0.5	0.2	0.2	0.2	0.8	0.5	0.8	0.2	**0.0**
90	kra32	32	88,700	**0.0**	–	–	**0.0**	0.3	**0.0**	–	–	–	–	–	–	–	–	–	**0.0**
91	ste36a	36	9526	**0.0**	5.1	0.1	**0.0**	–	**0.0**	–	3.5	1.2	1.9	2.8	2.0	1.0	3.4	2.0	**0.0**
92	ste36b	36	15,852	**0.0**	13.2	**0.0**	**0.0**	–	**0.0**	–	3.6	1.4	5.2	2.7	3.6	2.0	3.8	3.6	**0.0**

Table 7 (continued)

Sl. no.	Problem	Size	BKS	PBILA-VNS	BSA	BSAL	MSH-QAP	BBOTS	TMSGD-QAP	IHuS	PMX	PMX2	OX	OXI	POX	POX2	MUT	MUT1	WFA
93	ste36c	36	8,239,110	**0.0**	–	–	**0.0**	–	**0.0**	–	1.9	1.1	2.4	1.9	1.8	0.6	1.5	1.5	**0.0**
Average				**0.00**	4.65	**0.00**	**0.00**	0.08	**0.00**	0.92	0.91	0.84	1.09	0.86	0.89	0.76	0.93	0.78	**0.00**

Bold-faced zero indicates that the algorithm has provided the Best Known Solution

BKS best known solution

Table 8 Comparison of PBILA-VNS with state-of-the-art algorithms on Type IV problem instances

Sl. no.	Problem	Size	BKS	PBILA-VNS	BLS-OpenMP	GPU-QAP	BSA	BSAL	MSH-QAP	TMSGD-QAP	IHuS	WFA
94	tai12b	12	39,464,925	**0.0**	–	–	1.5	**0.0**	**0.0**	–	**0.0**	**0.0**
95	tai15b	15	51,765,268	**0.0**	–	–	0.1	**0.0**	**0.0**	–	**0.0**	**0.0**
96	tai20b	20	122,455,319	**0.0**	**0.0**	**0.0**	0.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
97	tai25b	25	344,355,646	**0.0**	**0.0**	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**
98	tai30b	30	637,117,113	**0.0**	**0.0**	**0.0**	–	–	**0.0**	**0.0**	**0.0**	**0.0**
99	tai35b	35	283,315,445	**0.0**	**0.0**	1.8	–	–	**0.0**	**0.0**	0.1	**0.0**
100	tai40b	40	637,250,948	**0.0**	**0.0**	2.3	3.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
101	tai50b	50	458,821,517	**0.0**	**0.0**	–	–	–	**0.0**	**0.0**	**–**	**0.0**
Average				**0.00**	**0.00**	0.83	1.33	**0.00**	**0.00**	**0.00**	0.02	**0.00**

Bold-faced zero indicates that the algorithm has provided the Best Known Solution

BKS best known solution

Thus, combining the results of comparison with the state-of-art algorithms available in the literature, PBILA-VNS is performing equally or better than most of the other algorithms under consideration. There is no single algorithm from the state-of-the-art algorithms considered for comparison performs better than the proposed PBILA-VNS. The overall average percentage deviation of the obtained solutions from the best-known solutions is 0.037%. Moreover, except WFA, none of the other state-of-the-art algorithms reported solutions for 101 QAP benchmark instances considered for this study.

5 Conclusion

In this study, a hybrid algorithm combining PBILA with VNS is proposed to solve one of the most complex combinatorial optimization problems, namely QAP. This study reveals that PBILA on its own fails to obtain the BKS for most of the test instances considered. Out of the total 101 test instances solved PBILA is able to obtain BKS for only six problems. But, when hybridized with VNS, the performance of the algorithm improved drastically. The proposed hybrid algorithm performs well by obtaining the BKS for most of the QAP test instances considered. PBILA-VNS is able to obtain the optimal solution at least once for 95 problems and for 49 test instances the PBILA-VNS is able to get the known best solution in all the ten trials. The results obtained for the PBILA-VNS are then compared with those of the recent state-of-the-art algorithms published in the literature.

From the comparison tables it is found that the proposed algorithm performs better or equally with the algorithms considered for comparison. Thus, this paper opens up a new direction in terms of the design of algorithms for solving complex combinatorial optimization problems. Applying the proposed PBILA-VNS for solving other combinatorial optimization problems can be an extension of this work. Hybridizing PBILA with other local search methods may improve the performance of the algorithm, which can provide a further scope for research.

Authors' contributions All authors contributed to the development of the solution methodologies and the preparation of the manuscript. The first draft of the manuscript and revised manuscript was written by Pradeepmon T. G.; R. Sridharan and Vinay V. Panicker did the editing of the manuscript. All authors read and approved the final manuscript.

Availability of data and material All data used in the work are available in the internet or are adapted from other published materials.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Code availability The code used in the work are developed by the authors and can be made available as and when required.

References

1. Acan, A., Ünveren, A.: A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem. Appl. Soft Comput. **36**, 185–203 (2015)
2. Adams, W.P., Johnson, T.A.: Improved linear programming based lower bounds for the quadratic assignment problem. DIMACS Ser. Discrete Math. Theor. Comput. Sci. **16**(1), 43–75 (1994)
3. Agharghor, A., Riffi, M.E., Chebihi, F.: Improved hunting search algorithm for the quadratic assignment problem. Indones. J. Electr. Eng. Comput. Sci. **14**(1), 143–154 (2019)
4. Ahmad, F., et al.: Quadratic assignment approach for optimization of examination scheduling. Appl. Math. Sci. **9**(130), 6449–6460 (2015)
5. Ahmed, Z.H.: A multi-parent genetic algorithm for the quadratic assignment problem. OPSEARCH, 1–19 (2015a)
6. Ahmed, Z.H.: An improved genetic algorithm using adaptive mutation operator for the quadratic assignment problem. In: Paper Presented at 38th International Conference on Telecommunications and Signal Processing (TSP), 2015, IEEE, 1–5 (2015b)
7. Ahuja, R.K., Orlin, J.B., Tiwari, A.: A greedy genetic algorithm for the quadratic assignment problem. Comput. Oper. Res. **27**(10), 917–934 (2000)
8. Aksan, Y., Dokeroglu, T., Cosar, A.: A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. Comput. Ind. Eng. **103**, 105–115 (2017)
9. Amico, M.D., et al.: The single-finger keyboard layout problem. Comput. Oper. Res. **36**(11), 3002–3012 (2009)
10. Anderson, E.J.: Mechanisms for local search. Eur. J. Oper. Res. **88**(1), 139–151 (1996)
11. Arkin, E.M., Hassin, R., Sviridenko, M.: Approximating the maximum quadratic assignment problem. Inf. Process. Lett. **77**(1), 13–16 (2001)
12. Azarbonyad, H., Babazadeh, R.: A genetic algorithm for solving quadratic assignment problem (QAP). Comput. Res. Reposit. abs/1405.5050 (2014)
13. Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report. Carnegie Mellon University (1994)
14. Banzhaf, W.: The "molecular" traveling salesman. Biol. Cybern. **64**(1), 7–14 (1990)
15. Barvinok, A., Stephen, T.: The distribution of values in the quadratic assignment problem. Math. Oper. Res. **28**(1), 64–91 (2003)
16. Battiti, R., Tecchiolli, G.: Simulated annealing and tabu search in the long run: a comparison on QAP tasks. Comput. Math. Appl. **28**(6), 1–8 (1994)
17. Bazaraa, M.S., Sherali, H.D.: Bender's partitioning scheme applied to a new formulation of the quadratic assignment problem. Naval Res. Logist. Q. **27**(1), 29–41 (1980)
18. Bazaraa, M.S., Sherali, H.D.: On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. J. Oper. Res. Soc. **33**(11), 991–1003 (1982)
19. Benlic, U., Hao, J.-K.: Memetic search for the quadratic assignment problem. Expert Syst. Appl. **42**(1), 584–595 (2015)
20. Bozorgi, N., Abedzadeh, M.: A multiple criteria facility layout problem using data envelopment analysis. Manag. Sci. Lett. **1**, 363–370 (2011)
21. Brixius, N.W., Anstreicher, K.M.: Solving quadratic assignment problems using convex quadratic programming relaxations. Optim. Methods Softw. **16**(1–4), 49–68 (2001)
22. Brixius, N.W., Anstreicher, K.M.: The Steinberg Wiring problem. Paper presented at Grötschel, M. (ed.) The Sharpest Cut: The Impact of Manfred Padberg and His Work, Philadelphia, USA: Society for Industrial and Applied Mathematics (MOS-SIAM Series on Optimization), pp. 293–307 (2004)
23. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment Problems. Society for Industrial and Applied Mathematics, Philadelphia (2009)
24. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB—a quadratic assignment problem library. J. Global Optim. **10**(4), 391–403 (1997)
25. Burkard, R.E., Rendl, F.: A thermodynamically motivated simulation procedure for combinatorial optimization problems. Eur. J. Oper. Res. **17**(2), 169–174 (1984)
26. Burkard, R.E., et al.: The Quadratic Assignment Problem. Paper presented at Du, D.-Z., Pardalos, P. (eds) Handbook of Combinatorial Optimization, Springer US, pp. 1713–1809 (1998)
27. Ceberio, J., et al.: A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Prog. Artif. Intel. **1**(1), 103–117 (2012)

28. Chmiel, W.: Evolutionary algorithm using conditional expectation value for quadratic assignment problem. Swarm Evolut. Comput. **46**(1), 1–27 (2019)
29. Clausen, J., Perregaard, M.: Solving large quadratic assignment problems in parallel. Comput. Optim. Appl. **8**(2), 111–127 (1997)
30. Connolly, D.T.: An improved annealing scheme for the QAP. Eur. J. Oper. Res. **46**(1), 93–100 (1990)
31. Czapiński, M.: An effective Parallel Multistart Tabu Search for Quadratic Assignment Problem on CUDA platform. J. Parallel Distrib. Comput. **73**(11), 1461–1468 (2013)
32. Day, R.O., Kleeman, M.P., Lamont, G.B.: Solving the multi-objective quadratic assignment problem using a fast messy genetic algorithm. Paper presented at Proceedings of Congress Evolutionary Computation (CEC'03), pp. 2277–2283 (2003)
33. Deineko, V.G., Woeginger, G.J.: A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. Math. Program. **87**(3), 519–542 (2000)
34. Ding, Y., Wolkowicz, H.: A low-dimensional semidefinite relaxation for the quadratic assignment problem. Math. Oper. Res. **34**(4), 1008–1022 (2009)
35. Dokeroglu, T., Cosar, A.: A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. Eng. Appl. Artif. Intell. **52**, 10–25 (2016)
36. Drezner, Z.: The extended concentric tabu for the quadratic assignment problem. Eur. J. Oper. Res. **160**(2), 416–422 (2005)
37. Drezner, Z.: Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. Comput. Oper. Res. **35**(3), 717–736 (2008)
38. Drezner, Z.: The quadratic assignment problem. In: Laporte, G., Nickel, S., Saldanha da Gama, F. (eds.) Location Science. Springer, Cham (2015)
39. Drezner, Z., MisevicIus, A.: Enhancing the performance of hybrid genetic algorithms by differential improvement. Comput. Oper. Res. **40**(4), 1038–1046 (2013)
40. Duman, E., Or, I.: The quadratic assignment problem in the context of the printed circuit board assembly process. Comput. Oper. Res. **34**(1), 163–179 (2007)
41. Emanuel, B., Wimer, S., Wolansky, G.: Using well-solvable quadratic assignment problems for VLSI interconnect applications. Discrete Appl. Math. **160**(4), 525–535 (2012)
42. Fleurent, C., Glover, F.: Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. Informs J. Comput. **11**(2), 198–204 (1999)
43. Forsberg, J.H., et al.: Analyzing Lanthanide-Induced Shifts in the NMR Spectra of Lanthanide(III) Complexes Derived from 1,4,7,10mbox-Tetrakis(N, N-diethylacetamido)-1,4,7,10-tetraazacyclododecane. Inorg. Chem. **34**(14), 3705–3715 (1995)
44. Francis, R.L., McGinnis Jr., F., White, J.A.: Facility Layout and Location—An Analytical Approach, 2nd edn. Prentice-Hall, New Jersey (Prentice-Hall International Series in Industrial and Systems Engineering) (1991)
45. Frieze, A.M., Yadegar, J.: On the quadratic assignment problem. Discrete Appl. Math. **5**(1), 89–98 (1983)
46. Gambardella, L.M., Taillard, É.D., Dorigo, M.: Ant colonies for the quadratic assignment problem. J. Oper. Res. Soc. **50**(2), 167–176 (1999)
47. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co, New York (1979)
48. Gonçalves, A.D., et al.: A graphics processing unit algorithm to solve the quadratic assignment problem using level-2 reformulation-linearization technique. Informs J. Comput. **29**(4), 676–687 (2017)
49. Hahn, P.M., et al.: Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem. Yugoslav J. Oper. Res. **11**(1), 41–60 (2001)
50. Hahn, P.M., Krarup, J.: A hospital facility layout problem finally solved. J. Intell. Manuf. **12**(5), 487–496 (2001)
51. Hansen, P., Mladenović, N., Pérez, J.M.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**(1), 367–407 (2010)
52. Hassin, R., Levin, A., Sviridenko, M.: Approximating the minimum quadratic assignment problems. ACM Trans. Algorithms **6**(1), 18:1–18:10 (2009)
53. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm Evol. Comput. **1**(3), 111–128 (2011)

54. Heffey, D.R.: Assigning runners to a relay team. Paper presented at Ladany, S.P., Machol, R.E. (eds.) Optimal Strategies in Sports (Studies in management science and systems, volume 5), North-Holland, Amsterdam: Elsevier North-Holland, pp. 169–171 (1977)
55. Heffley, D.R.: Decomposition of the Koopmans–Beckmann problem. Reg. Sci. Urban Econ. **10**(4), 571–580 (1980)
56. Helber, S., et al.: A hierarchical facility layout planning approach for large and complex hospitals. Flex. Serv. Manuf. J. **28**(1–2), 5–29 (2016)
57. Hong, G.: A hybrid ant colony algorithm for quadratic assignment problem. Open Electr. Electr. Eng. J. **7**(1), 51–54 (2013)
58. İşeri, A., Ekşioğlu, M.: Estimation of digraph costs for keyboard layout optimization. Int. J. Ind. Ergon. **48**, 127–138 (2015)
59. James, T., Rego, C., Glover, F.: A cooperative parallel tabu search algorithm for the quadratic assignment problem. Eur. J. Oper. Res. **195**(3), 810–826 (2009)
60. Karisch, S.E. (1995) Nonlinear approaches for quadratic assignment and graph partition problems. PhD diss., Technical University Graz, Austria
61. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica **25**(1), 53–76 (1957)
62. Krarup, J., Pruzan, P.: Computer-aided layout design. Paper presented at Balinski, M.L., Lemarechal, C. (eds.) Mathematical Programming in Use, Berlin, Germany: Springer Berlin Heidelberg (Mathematical Programming Studies), pp. 75–94 (1978)
63. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Boston, MA (2002)
64. Li, W.J., Smith, J.M.: Theory and methodology: an algorithm for quadratic assignment problems. Eur. J. Oper. Res. **81**, 205–216 (1995)
65. Lim, W.L., et al.: A biogeography-based optimization algorithm hybridized with tabu search for the quadratic assignment problem. Comput. Intell. Neurosci. **2016**, 1–12 (2016)
66. Loiola, E.M., et al.: A survey for the quadratic assignment problem. Eur. J. Oper. Res. **176**(2), 657–690 (2007)
67. Loiola, E.M., et al.: An analytical survey for the quadratic assignment problem, Council for the Scientific and Technological Development, of the Brazilian Gov (2004)
68. Lozano, J.A.: Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer, New York (2006)
69. Matsui, S., et al.: Exponential chaotic tabu search hardware for quadratic assignment problems using switched-current chaotic neuron IC. Paper presented at Proceedings of IEEE Internationa Joint Conference on Neural Networks, pp. 2221–2225 (2004)
70. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer-Verlag, London (1996)
71. Misevicius, A.: An intensive search algorithm for the quadratic assignment problem. Informatica **11**(2), 145–162 (2000)
72. Misevicius, A.: A modified simulated annealing algorithm for the quadratic assignment problem. Informatica **14**(4), 497–514 (2003)
73. Misevicius, A.: An improved hybrid optimization algorithm for the quadratic assignment problem. Math. Model. Anal. **9**(2), 149–168 (2004)
74. Misevicius, A., Guogis, E.: Computational study of four genetic algorithm variants for solving the quadratic assignment problem. Paper presented at International Conference on Information and Software Technologies, Springer, pp. 24–37 (2012)
75. Mladenovic, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
76. Ng, K.M., Tran, T.H.: A parallel water flow algorithm with local search for solving the quadratic assignment problem. J. Ind. Manag. Optim. **15**(1), 235–259 (2019)
77. Nyberg, A., Westerlund, T.: Tightening a discrete formulation of the quadratic assignment problem. Chem. Eng. Trans. **32**(1), 1309–1314 (2013)
78. Osman, H., Demirli, K.: Economic lot and delivery scheduling problem for multi-stage supply chains. Int. J. Prod. Econ. **136**(2), 275–286 (2012)
79. Osman, I.H., Laporte, G.: Metaheuristics: a bibliography. Ann. Oper. Res. **63**(5), 511–623 (1996)
80. Pardalos, P.M., Xue, J.: The maximum clique problem. J. Global Optim. **4**(3), 301–328 (1994)
81. Paul, G.: An efficient implementation of the simulated annealing heuristic for the quadratic assignment problem. Comput. Res. Reposit. abs/1111.1353 (2011)

82. Paul, G.: A GPU implementation of the simulated annealing heuristic for the quadratic assignment problem. Comput. Res. Reposit. abs/1208.2675 (2012)
83. Pelikan, M., Hauschild, M. W. and Lobo, F. G., Introduction to estimation of distribution algorithms, MEDAL Report. 2012003. Missouri, USA:Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), Department of Mathematics and Computer Science, University of Missouri,Columbia, USA. (2012)
84. Peng, T., Huanchen, W., Dongme, Z.: Simulated annealing for the quadratic assignment problem: a further study. Comput. Ind. Eng. **31**(3/4), 925–928 (1996)
85. Pitsoulis, L.S., Pardalos, P.M., Hearn, D.W.: Approximate solutions to the turbine balancing problem. Eur. J. Oper. Res. **130**(1), 147–155 (2001)
86. Ramkumar, A.S., et al.: Iterated fast local search algorithm for solving quadratic assignment problems. Robot. Comput. Integr. Manuf. **24**(3), 392–401 (2008)
87. Santana, R., Mendiburu, A., Lozano, J.A.: A review of message passing algorithms in estimation of distribution algorithms. Nat. Comput. **15**(1), 165–180 (2016)
88. See, P.C., Wong, K.Y.: Application of ant colony optimisation algorithms in solving facility layout problems formulated as quadratic assignment problems: a review. Int. J. Ind. Syst. Eng. **3**(6), 644–672 (2008)
89. Skorin-Kapov, J.: Tabu search applied to the quadratic assignment problem. ORSA J. Comput. **2**(1), 33–45 (1990)
90. Song, L.Q., Lim, M.H., Ong, Y.S.: Neural meta-memes framework for managing search algorithms in combinatorial optimization. Paper presented at IEEE Workshop on Memetic Computing (MC), 2011, pp. 1–6 (2011)
91. Steinberg, L.: The backboard wiring problem: a placement algorithm. SIAM Rev. **3**(1), 37–50 (1961)
92. Talbi, E.G., Hafidi, Z., Geib, J.M.: A parallel adaptive tabu search approach. Parallel Comput. **24**(14), 2003–2019 (1998)
93. Tate, D.M., Smith, A.E.: A genetic approach to the quadratic assignment problem. Comput. Oper. Res. **22**(1), 73–83 (1995)
94. Tosun, U.: A new recombination operator for the genetic algorithm solution of the quadratic assignment problem. Proc. Comput. Sci. **32**, 29–36 (2014)
95. Tosun, U.: On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem. Eng. Appl. Artif. Intell. **39**, 267–278 (2015)
96. Tosun, U., Dokeroglu, T., Cosar, A.: A robust island parallel genetic algorithm for the quadratic assignment problem. Int. J. Prod. Res. **51**(14), 4117–4133 (2013)
97. Tseng, L.Y., Liang, S.C.: A hybrid metaheuristic for the quadratic assignment problem. Comput. Optim. Appl. **34**(1), 85–113 (2006)
98. Urban, T.L.: Solution procedures for the dynamic facility layout problem. Ann. Oper. Res. **76**, 323–342 (1998)
99. Uwate, Y., et al.: Performance of chaos and burst noises injected to the hopfield NN for quadratic assignment problems. IEICE Trans. Fundam. Electr. Commun. Comput. Sci. **E87-A**(4), 937–943 (2004)
100. Wakabayashi, S., Kimura, Y., Nagayama, S.: FPGA implementation of tabu search for the quadratic assignment problem. Paper presented at Proceedings of IEEE International Conference on Field Programmable Technology (FPT 2006), pp. 269–272 (2006)
101. West, D.H.: Algorithm 608: approximate solution of the quadratic assignment problem. ACM Trans. Math. Softw. **9**(4), 461–466 (1983)
102. Wilhelm, M.R., Ward, T.L.: Solving Quadratic Assignment Problems by 'Simulated Annealing. IIE Trans. **19**(1), 107–119 (1987)
103. Wu, Z., et al.: Global optimality conditions and optimization methods for quadratic assignment problems. Appl. Math. Comput. **218**(11), 6214–6231 (2012)
104. Wu, Y., Ji, P.: Solving the quadratic assignment problems by a genetic algorithm with a new replacement strategy. Int. J. Hum. Soc. Sci. 151–155 (2007)
105. Xia, Y.: An efficient continuation method for quadratic assignment problems. Comput. Oper. Res. **37**(6), 1027–1032 (2010)

106. Yamada, S.: A new formulation of the quadratic assignment problem on r-dimensional grid. IEEE Trans. Circuits Syst. I Fundam. Theory Appl. **39**(10), 791–797 (1992)
107. Zaied, A.N.H., Shawky, L.A.E.-F.: A survey of the quadratic assignment problem. Int. J. Comput. Appl. **101**(6), 28–36 (2014)
108. Zhou, J., et al.: An exact penalty function method for optimising QAP formulation in facility layout problem. Int. J. Prod. Res. **55**(10), 2913–2929 (2017)
109. Zhu, W., Curry, J., Marquez, A.: SIMD tabu search for the quadratic assignment problem with graphics hardware acceleration. Int. J. Prod. Res. **48**(4), 1035–1047 (2010)