Hybrid teaching-learning-based optimization algorithms for the Quadratic Assignment Problem

Tansel Dokeroglu*

Computer Engineering Department of Turkish Education Association University, Ankara/Turkey

Abstract

Teaching-Learning-Based Optimization (TLBO) is a novel swarm intelligence metaheuristic that is reported as an efficient solution method for many optimization problems. It consists of two phases where all individuals are trained by a teacher in the first phase and interact with classmates to improve their knowledge level in the second phase. In this study, we propose a set of TLBO-based hybrid algorithms to solve the challenging combinatorial optimization problem, Quadratic Assignment. Individuals are trained with recombination operators and later a Robust Tabu Search engine processes them. The performances of sequential and parallel TLBO-based hybrid algorithms are compared with those of state-of-the-art metaheuristics in terms of the best solution and computational effort. It is shown experimentally that the performance of the proposed algorithms are competitive with the best reported algorithms for the solution of the Quadratic Assignment Problem with which many real life problems can be modeled.

Keywords: Teaching-learning, hybrid algorithm, robust tabu, quadratic assignment, stagnation

1. Introduction

The QAP is a mathematical model for the location of indivisible economic activities and it was first introduced by Koopmans and Beckmann (1957). Variations of the QAP have been studied over the years in the domains of telecommunications, transportation systems, and signal processing (Burkard, Karisch, & Rendl, 1991). Typewriter keyboard design, backboard wiring (Steinberg, 1961), layout design (Rossin, Springer, & Klein, 1999), turbine balancing (Pfister, 1998), scheduling (Lim, Yuan, & Omatu, 2000), and data allocation (Adl, & Rankoohi, 2009) are some instances of the problems that have been successfully modeled as a QAP. The service allocation problem with the purpose of minimizing the container re-handling operations at a shipyard (Cordeau, Gaudioso, Laporte, & Moccia, 2007), travelling salesman, bin-packing, maximum clique, linear ordering, and the graph-partitioning problem are among the applications of the QAP (Pentico, 2007).

In its simplest form, the QAP is the problem of assigning facilities to locations with a cost of transportation. The objective is to find an allocation such that the total cost of allocating and operating all facilities is minimized. The QAP can be formally modeled by using three $n \times n$ matrices, A, B, and C.

$$A = (a_{ik}) \tag{1}$$

where a_{ik} is the flow amount from facility i to facility k.

$$B = (b_{jl}) \tag{2}$$

where b_{jl} is the distance from location j to location l.

*Corresponding author
Email address: tansel@ceng.metu.edu.tr (Tansel Dokeroglu*)

$$C = (c_{ij}) \tag{3}$$

where c_{ij} is the cost of placing facility i at location j.

The Koopmans-Beckmann form of the QAP can be written as:

$$min_{\phi \epsilon S_n}(\sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik}b_{\phi(i)\phi(k)} + \sum_{i=1}^{n} c_{i\phi(i)}) \tag{4}$$

where S_n is permutation of integers 1,2 ,..., n. Each term $a_{ik}b_{\phi(i)\phi(k)}$ is the transportation cost from facility i at location ϕ (i) to facility k at location ϕ (k). Each term $c_{i\phi(i)}$ is the total cost for installing facility i, at location $\phi(i)$, plus the transportation costs to all other facilities k, installed at locations $\phi(1)$, $\phi(2)$,..., $\phi(n)$ (the range of the indexes i, l, j, k is 1,...,n). The QAP (A, B is an instance where A, B, and C are input matrices given at Equations 1,2,3. If there is no C term, we can write it as a QAP (A, B). Lawler introduced a four-index cost array $D = (d_{ijkl})$ instead of the three matrices and obtained the general form of the QAP as (Lawler, 1963):

$$min_{\phi \epsilon S_n}(\sum_{i=1}^{n}\sum_{k=1}^{n} d_{i\phi(i)k\phi(k)}) \tag{5}$$

The relationship with the Koopmans-Beckmann problem is:

$$d_{ijkl} = a_{ik}b_{jl}(i, j, k, l = 1, 2, ..., n; i \neq k \text{ or } j \neq l) \tag{6}$$

$$d_{ijij} = a_{ii}b_{jj} + c_{ij}(i, j = 1, 2, ..., n) \tag{7}$$

Although small QAP instances can be solved with exact solution methods, the optimal solutions for large instances cannot be found due to the computational limitations. However, metaheuristic approaches are robust tools with their ability to produce high-quality solutions for such conditions. Genetic Algorithms (GA) (Tate, & Smith, 1995), Simulated Annealing (Connolly, 1990), Neural Networks (Calzon-Bousono, 1995), GRASP (Li, Pardalos, & Resende, 1994), Robust Tabu Search (RTS) (Taillard, 1991), and Ant Colony Optimization (Gambardella, & Taillard, & Dorigo, 1999) are some of the well-known of these methods that have been successfully applied to the QAP.

Heuristic-based methods have different techniques ranging from simple local searches to complex learning processes. They guide the search strategically and use mechanisms to avoid becoming stuck into a local optimum. Metaheuristics such as GA (Holland, 1975; Goldberg, 1989; Dokeroglu, & Cosar, 2014), Ant Colony Optimization (ACO) (Dorigo, & Caro, 1999), Particle Swarm Optimization (PSO) (Kennedy, & Eberhart, 1995), Differential Evolution (DE) (Storn, & Price, 1997), Harmony Search (HS) (Geem, Kim, & Loganathan, 2001), Shuffled Frog Leaping (SFL) (Eusuff, & Lansey, 2003), Artificial Bee Colony (ABC) (Karaboga, & Basturk, 2007), Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009), and Grenade Explosion Method (GEM) (Ahrari, & Atai, 2010) have been applied to many engineering optimization successfully.

Teaching Learning Based Optimization (TLBO) has been recently introduced as a novel metaheuristic that is inspired by the knowledge passing mechanism of teachers and learners in a classroom (Rao, Savsani, & Vakharia, 2011; Rao, Savsani, & Vakharia, 2012a; Rao, Savsani, & Balic, 2012b; Rao, & Patel, 2012a; Rao, & Patel, 2012b; Rao, & Patel, 2012c; Rao, & Patel, 2012d; Rao, & Patel, 2012e). Learners are first trained by a knowledgable teacher and later they continue improving their knowledge level through interactions between classmates. The TLBO has an algorithm-specific parameterless concept that does not require any parameters to be optimized. Population size, number of generations, elite size, etc. are the common control parameters that need to be tuned by all of the population based metaheuristics (including TLBO). However, in addition to these parameters, PSO uses inertia weight, social and cognitive parameters, GA uses crossover and mutation rate, ABC uses number of bees, HS uses harmony memory consideration rate, pitch adjusting rate, and number of improvisations. The optimal tuning of these parameters is crucial for the optimization that either it may increase the computational effort or yield local optimal solutions. On the other hand, TLBO requires only the common control parameters to be tuned.

Remarkable results have been reported about the performance of TLBO in comparison with the other metaheuristics on different constrained benchmark functions, constrained mechanical design problems (Rao, Savsani, &

Vakharia, 2011), and on continuous non-linear numerical optimization problems (Rao, Savsani, & Vakharia, 2012a) in terms of computational efforts and solution quality. Although there are still ongoing discussions about its performance and promises, this success of the TLBO has drawn our attention and we modeled the solution of the challenging combinatorial optimization problem, QAP, with TLBO (Waghmare, 2013; Črepinšek, Liu, & Mernik, 2012). In our proposed hybrid algorithms (TLBO-RTS), individuals are trained with recombination operators by using TLBO methods before they are given to a RTS engine. The recombination operators do not require any parameter settings in accordance with the parameterless optimization concept of TLBO. There is also no need to apply an additional selection mechanism such as roulette wheel, tournament, or truncation. It is shown experimentally that proposed hybrid TLBO-RTS algorithm is an effective optimization approach for the QAP compared to the other state-of-the-art metaheuristics.

In section 2, state-of-the-art solution metaheuristics for the solution of the QAP are given. A detailed information about the TLBO optimization method is presented in section 3. RTS (used in the hybrid algorithms) is presented in section 4. Recombination operators and proposed hybrid TLBO-based algorithms are explained in section 5. The setup of the experimental environment, obtained results, and comparison with state-of-the-art metaheuristics are presented in section 6. Concluding remarks are provided in the last section.

2. Related Work

Several algorithms have been proposed for both exact and approximate solutions of the QAP. Exact algorithms are limited to solving small data sets of the QAP with massively parallel computers, whereas metaheuristics can provide (near-)optimal solutions within reasonable optimization times for larger problem instances.

Branch and Bound (BB) algorithms are the most elegant exact solution approaches to solve the QAP (Lawler, 1963; Carraresi, & Malucelli, 1992; Gilmore, 1962; Pardalos, Rendl, & Wolkowicz, 1994). Until 1990 an exact solution of the QAP instances of size 20 was not possible. Mautor, & Roucairol (1994) gave exact solutions for 'nug16', 'els19', and size-20 problem instances. Clausen and Perregaard (1997) solved 'nug20' with a parallel BB algorithm. Marzetta & Brungger (1995) solved 'nug25' instance by using parallel dynamic programming. Anstreicher, Brixius, Goux, & Linderoth (2002) developed a convex quadratic programming relaxation within a BB algorithm and this algorithm provided an exact solution for 'nug25' instance after 13 days of CPU time using sequential processing. Hahn, & Krarup, (2001) solved 'kra30a' after 99 days of work with a sequential workstation. Nystrom (1999) gave the optimal solution for 'ste36b' and 'ste36c' instances after 200 days of work in a distributed environment. Anstreicher, Brixius, Goux, & Linderoth, (2002) reported the exact solution of 'nug30', which required seven days with 650 processors (seven years of computation on a single CPU).

RTS, is one of the best known algorithms that can produce high quality solutions (Taillard, 1991). James, Rego, & Glover (2006) introduced a multi-start TS algorithm (JRG-DivTS). When the search stagnates, JRG-DivTS restarts from a diversified copy of the best solution found up to that point. Two categories of multi start TS methods are developed. The first category searches by modifying the so-called continuous diversification and describes a Restricted Descent TS (RDTS) and a Tabu Tenure Modification TS (TTMTS) which alter the tabu list matrix and tenure parameters. The second category implements a discontinuous diversification TS process and contains the Random Restart TS (RRTS). The performance of the second category methods are reported to be better than the first category and DivTS is used to build the proposed Cooperative Parallel TS (CPTS) algorithm. Misevičius, A. (2012) proposes an effective implementation of the heuristic algorithm based on an Iterated Tabu Search (ITS) paradigm recently.

Successful algorithms such as the Ant Colony Optimization (Gambardella, & Taillard, & Dorigo, 1999; Stützle, & Dorigo, 1999), Simulated Annealing (Connolly, 1990), and Neural Networks (Chakrapani, & Skorin-Kapov, 1992) have been proposed. The combination of population based metaheuristics and TS shows a hybrid behavior that provides competitive high quality solutions. The solution provided by TS procedure is combined with a so-called RTS (Taillard, 1991) by Misevičius (2005) while perturbing the solution via diversification operators. This algorithm effectively explores symmetric and asymmetric instances given by Taillard (Burkard, Karisch, & Rendl, 1991). Hybrid algorithms which exploit RTS like sequential metaheuristics produce high quality solutions in combination with GA variants (Misevičius, 2005). Ahuja, Orlin, & Tiwari, (2000) and Drezner (2005) have successfully incorporated GA variants into TS.

Zhang et al., (2010) apply two formulation reductions to the QAP. Xia (2010) proposes a Lagrangian Smoothing Algorithm (LagSA), where the continuation subproblems are solved by the truncated FrankWolfe algorithm. Fescioglu-Unver et al. (2011) address the application of the principles of feedback and self-controlling software to the TS algorithm with Self Controlling TS (SC-Tabu) algorithm. Duman et al. (2012) propose a new nature inspired metaheuristic approach based on the V flight formation of the migrating birds which is proven to be an effective formation in energy saving (Migrating Birds Optimization (MBO) algorithm).

The granularity of the solution space, the repetitive behavior of the metaheuristic methods, and the solution complexity make parallelization an attractive alternative for the QAP (Dokeroglu, Tosun, & Cosar, 2012; Luque & Alba, 2011; Kumar, 2002). Crainic and Toulouse (2003) illustrate the differences and the design choices of the parallel metaheuristic algorithms. By looking at the work assigned to processors, they grouped these algorithms into three types. In Type 1, parallelization is low and the master first assigns works to each processor, then collects the results. Type 2 divides decision variables among processors , dramatically reducing the search space assigned to each processor. The resulting solutions are combined by a master process. Type 3 is a multi-heuristic parallel model in which each processor executes a different heuristic. This model uses cooperative strategies that make use of information exchange. In this model, different heuristics can be employed at each processor.

If we inspect the parallel algorithms from the perspective of Crainic, & Toulouse, (2003), Taillard proposed an example of Type 1 algorithms (Taillard, 1991). He studied the subdivision of the neighborhood exploration between processors, represented by a permutation. The swap of two facilities to each other's previously occupied location is subdivided between processors. After calculating the costs, each processor sends its best move to all processors and the others perform the global best move on their copy of the permutation. Chakrapani and Skorin-Kapov (1992) examined a similar algorithm implemented on a single instruction-multiple data (SIMD) computer. James et al. (2005) developed another member of this category, the path-relinking algorithm which applies path-relinking to solutions in a global reference set. There is no known Type 2 algorithm for the QAP.

Taillard proposed a parallel Type 3 TS algorithm that uses different initial solutions (Taillard, 1991). There is no migration of solutions between the processors and when all threads complete, the best solution is returned. Talbi et al., (1997) proposed a multi-heuristic parallel TS with different initial starting states. Talbi and Bachelet (2006) proposed both an independent TS and a cooperative multi-threaded metaheuristic, COSEARCH, which combines the TS results with a GA for diversification purposes. Tsutsui and Fujimoto (2009) proposed a new parallel scheme using parallel graphic processing units (GPUs) to solve the QAP by combining the computational power of parallel machines with GAs for solving instances up to 40 locations. James, Rego, & Glover (2009) introduced a cooperative parallel TS algorithm (CPTS) for the QAP. Czapiński, M. (2013) proposed a Parallel Multistart Tabu Search (PMTS) running many Tabu Search instances on a GPU using CUDA.

3. Teaching-learning-based optimization

When we consider two teachers that teach a subject to the same merit level learners in two different classes, we can see that they exhibit varying successes in terms of teaching. The main difference between them is that a good teacher produces a better average achieved merit levels for the learners. Learners also acquire knowledge through interaction with their classmates. TLBO is based on this teaching process. A mathematical formulation is prepared and implemented for TLBO based optimization in (Rao, Savsani, & Vakharia, 2011; Rao, Savsani, & Vakharia, 2012a). The teacher is the most knowledgeable person in the population, so the best learner is employed as a teacher. The teacher disseminates knowledge among learners, which increases the knowledge level of the class and helps learners to get good grades. A teacher increases the average success of the class in accordance with his teaching capability, thereby increasing the level of learners. Teacher's success is measured by average knowledge level achieved by students in the class which is determined by quality of teaching as well as the learners' quality. Teacher works to improve class quality and when improvement stops, assigning a new and better quality teacher may be necessary to achieve even higher class quality. The quality of the students is judged from the mean value of the population. Teacher puts effort in order to increase the quality of the students and at some stage the students may require a new teacher, of superior quality than themselves. Therefore, a new teaching process will start with a new teacher.

Like other nature-inspired algorithms, TLBO is also a population-based method that uses the solutions presented by individuals search for the global optimum. The population in TLBO is considered as a class of students and each

individual in the class has a fitness value. The teacher is the individual with the best fitness value. TLBO consists of two phases, teacher phase and learner phase.

3.1. Teacher phase

Teacher quality determines the eventual achieved class quality. A teacher can increase class level only up to his own level and the capability of the class to learn also affects the level that a class can attain. This process depends on many factors. Let M_i be the mean and T_i be the teacher at any iteration i. T_i will try to move mean M_i towards its own level, so that the new mean will be T_i designated as M_{new}. The solution is updated according to the difference between the existing and the new mean given by:

$$Difference\_Mean_i := r_i(M_{new} - T_F M_i) \tag{8}$$

where T_F is a *teaching factor* that decides how the mean value will be changed by a teacher, and r_i is a random number in the range [0, 1]. The value of T_F can be either 1 or 2, which is again a heuristic step that is decided randomly with equal probability as T_F = round[1 + rand(0, 1){2-1}].

This difference modifies the existing solution according to the following expression:

$$X_{new,i} := X_{old,i} + Difference\_Mean_i \tag{9}$$

3.2. Learner phase

Learners increase their knowledge through input from the teacher and interaction between classmates. A learner interacts randomly with other learners with group discussions, presentations, formal communications, etc. A learner learns new things if the other classmate has more knowledge than him or her. A random classmate is selected from the population and this selected classmate trains his friend. If the resulting individual is better than the former one it is replaced with the new generated individual (see Algorithm 1).The modification of the learners for randomly selected two learners where $X_i \neq X_j$ is expressed as:

$$if(X_i < X_j) \;\; then \;\; X_{new,i} := X_{old,i} + r_i(X_i - X_j) \tag{10}$$

$$if(X_i > X_j) \;\; then \;\; X_{new,i} := X_{old,i} + r_i(X_j - X_i) \tag{11}$$

4. Robust Tabu Search

TS starts with a steepest descent algorithm and continues exploring with upward and downward moves towards the solution. Earlier moves affect the future moves and a tabu list prevents moving in a reverse direction for a certain number of moves. This way the process is forced to search through unexplored areas. When the search process gets stuck in a local minimum, it tries to find a better solution by escaping from it, even if it means moving to a worse solution. Moves that force the search back into a local minimum are not allowed by the tabu list.

TS has an adaptive memory to explore the search space and uses this memory to forbid returning to recently visited solutions. Variants of TS use different strategies such as diversification and intensification to focus on more promising search regions. Changing the tabu list size is the basic strategy. A smaller size tabu list enables the search in and around local minima, whereas bigger lists will help the search to escape from the local minima. Several aspiration criteria can be used with tabu restrictions to make a decision about a particular move. Taillard's RTS has an effective short-term memory with multiple-levels of aspiration criteria. Although the short-term memory is adequate for high-quality solutions, the longer term memory components have promising results. There are several TS algorithms exploring intensification and diversification strategies with a variety of parameter settings (Kelly, Laguna, & Glover, 1994; Laguna, Marti, & Campos, 1999). The long-term memory generally keeps track of the frequency of the components that occur in high-quality solutions. For long-term intensifications, components with a high frequency are selected for future searches. Diversifications can eliminate these components so that the search can move into unexplored regions.

Algorithm 1: TLBO

```
generate_population(population);
calculate_fitness_of_individuals (population);
for (k:=1 to number_of_generations) do
    for (i:=1 to number_of_individuals) do
        /* Learning from Teacher */
        T_F := round (1 + r);
        X_mean := calculate_mean_vector (population);
        X_teacher := best_individual (population);
        X_new := X_i + r(X_teacher − (T_F X_mean));
        if (X_new is better than X_i) then
            X_i := X_new;
        /* Learning from Classmates */
        j:=select_random_individual_from (population);
        if (X_i is better than X_j) then
            X_i,new := X_i + r(X_i − X_j);
        else
            X_i,new := X_j + r(X_j − X_i);
        if (X_i,new is better than X_i) then
            X_i := X_i,new;
show_the_best_individual(population);
```

RTS seeks a neighbor solution with the best evaluation. This evaluation forces the choice of a move that improves the objective function most. In order to avoid returning to the local optimum just visited, the reverse moves are forbidden (tabu list forbids the moves). There may be some moves exempt by the tabu criterions. These moves may lead to better solutions. Aspiration criterion is introduced to allow tabu moves whenever a move may have a chance to come up with a better result. Moreover, RTS uses tabu list size and number of failures as parameters. Number of failures define the range in which no improving solution is found in terms of number of iterations. According to Taillard, allowing more iterations provides better results (see Algorithm 2). Our proposed hybrid algorithms use RTS in its decision phase and we use the optimal RTS parameters set by Taillard (1991).

Exchanging the previous locations of two facilities with each other is an efficient way to generate new permutations for the neighborhood relations of the QAP. The computational simplicity of the swap makes it very appropriate when compared with the other larger exchange methods. In Taillard's RTS, these two-exchange moves save great amount of execution time. RTS uses a matrix to store the costs of each swap that will be executed in the current permutation and these costs are added to obtain the cost value of the new permutation. The fast evaluation of the moves is an important issue that contributes to the efficiency of the search.

5. Proposed Hybrid TLBO-RTS Algorithms

In this section, we give information about the sequential and parallel hybrid TLBO (TLBO-RTS) algorithms we propose. TLBO-RTS has two phases where TLBO constitutes the first phase and the second phase, RTS, explores the solution space of each individual that is produced by TLBO iterations.

5.1. Main characteristics of the algorithms

The most important factor of TLBO is that it provides an algorithm-specific parameterless optimization, which was the main goal of our study to develop such a combinatorial algorithm for the QAP (Rao, Savsani, & Vakharia,

Algorithm 2: Robust Tabu Search Algorithm (Taillard, 1991)

```
1  Authorized: If a move is not tabu, it is authorized.
2  Aspired: Allow tabu moves if they are decided to be interesting.
3  Tabu List: A list to forbid reverse move.
4  Neighbor: Each location in the permutation is considered as neighbor.
5  RTS (FLOW, DIST, MaxIter, BestPerm, MinSize(<n×n/2), MaxSize(<n×n/2), Aspiration(>n×n/2));
6  TABU_LIST = {};
7  CurCost = QAP_Cost(BestPerm);
8  CurSol = BestPerm;
9  Delta[i][j] = ComputeDelta(); /* i = 0,...,n, j = 0,...,n */
10 TABU_LIST[i][j] = - (n×i+j); /* i = 0,...,n-1, j = 0,...,n-1 */
11 for (iteration = 1; iteration < MaxIter; iteration++) do
12     i_retained = infinite;
13     MinDelta = infinite;
14     Already_Aspired = false;
15     for (each Neighbor (i, j)) do
16         current1 = TABU_LIST[i][CurSol[j]];
17         current2 = TABU_LIST[j][CurSol[i]];
18         Authorized = (current1 < iteration) || (current2 < iteration);
19         Aspired = (current1 < iteration-Aspiration)|| (current2 < iteration-Aspiration)|| (CurCost + Delta[i][j] <
             BestCost);
20         if ( (Aspired && Already_Aspired) || (Aspired && Delta[i][j] < MinDelta) ||
21         (!Aspired && !Already_Aspired && Delta[i][j] < MinDelta && Authorized) ) then
22             i_retained = i;
23             j_retained = j;
24             MinDelta = Delta[i][j];
25             if (Aspired) then
26                 Already_Aspired = true;
27     if (i_retained != infinite)) then
28         SWAP(CurSol[i_retained], CurSol[j_retained]);
29         CurCost = CurCost + Delta[i_retained][j_retained];
30         TABU_LIST[i_retained][CurSol[j_retained]] = iteration + getRandom(MinSize, MaxSize);
31         TABU_LIST[j_retained][CurSol[i_retained]] = iteration + getRandom(MinSize, MaxSize);
32         if (CurCost < BestCost) then
33             BestCost = CurCost;
34     UPDATE_MOVE_COSTS(FLOW, DIST, CurSol, Delta, i, j, i_retained, j_retained);
```

2011; Rao, Savsani, & Vakharia, 2012a; Waghmare, 2013). The common control parameters such as the number of individuals and generations are still needed to be set optimally by our algorithms as they are needed by all of the population based algorithms.

Recombination operators are efficiently used tools to pass the properties of the selected learners to the newly generated solutions (Goldberg, 1989; Tosun, Dokeroglu, & Cosar, 2013). The average mean value of individuals in the population can be improved with the properties of a teacher by using recombination operators. The recombination operators that work without any parameter settings are efficient tools to provide this feature of the TLBO. We have redesigned three conventional recombination operators to get rid of setting any parameter (the recombination operators do not need any additional parameters such as crossover rate (Holland, 1975)). These parameters are randomly selected. We have used three different recombination operators in the algorithms, Order Based Recombination operator (OBX), Cohesive Recombination operator (COHX), and Swap Path Recombination operator (SPX) (Ahuja, Orlin, & Tiwari, 2000; Drezner, 2003). SPX starts its process from a random starting point, OBX provides random number of facilities to pass to the learner, and COHX selects a random starting point to train the learner. We have neither defined nor tuned any additional parameters for a recombination operator when they are training the learners. In this sense, we can say that our proposed hybrid algorithms work with an algorithm-specific parameterless design in the TLBO phase. The individuals are first trained by a teacher (the fittest individual in the population) and later, they are trained by a randomly selected classmate by using recombination operators through generations. Therefore, we do not use any selection mechanism to train the learners either. Teacher trains all of the learners and learners train each other by a mechanism that randomly couples them. In GAs, this procedure goes on with selection mechanisms such as roulette-wheel, truncation, and tournament whereas, in TLBO-based algorithms all of individuals are trained by a teacher (fittest individual in the population) and random interactions of the learners between themselves (see Algorithm 3).

TLBO-RTS algorithm removes duplicate solutions in the classroom with randomly generated individuals as it is implemented in previous TLBO studies (Rao, Savsani, & Vakharia, 2011; Rao, Savsani, & Vakharia, 2012a) and the individuals are not eliminated by the new solutions unless new individuals are better. An additional Stagnation Prevention Procedure (SPP) is also proposed for the developed TLBO-RTS algorithms.

5.2. Solution representation and recombination operators

The representation of an individual is presented in Figure 1. The facilities are located on an array in accordance with the order of the sites. In the given example, facility 4 is placed at location 3.

Mutation is an operator that swaps the locations of two facilities. More than one interchange can be applied to the individual. In our algorithms we swap two facilities in a mutation operator. The facilities are randomly selected to produce new solutions (see Figure 2).

Swap Path Recombination operator (SPX) starts from a random facility of the teacher and examines all the facilities of the learner (Ahuja, Orlin, & Tiwari, 2000). If the facilities are the same it moves to the next facility, else it performs a swap of two facilities and so that the current positions of the teacher and learner become the same. If the new generated learner has a better fitness value it takes the place of the current learner. The facilities in the two resulting solutions are considered starting from the next position. The best solution obtained serves the new learner. The illustrative example of SPX is shown in Figure 3.

Order Based Recombination operator (OBX) preserves the relative order of facilities in the teacher solution. A number of facilities are randomly selected from the teacher and copied to the new learner. The other facilities are copied from the learner in their original order. Figure 4 shows how OBX recombination operator works (Lawrence, D. 1991).

Cohesive Recombination operator (COHX) is introduced by Drezner as an original and efficient recombination operator (Drezner, 2003). A median distance value is evaluated for a randomly chosen pivot value in the teacher (7 is the pivot in Figure 5) and a distance matrix is constructed. Sites closer than the median value to the pivot value are assigned from the teacher (Median value is 2 in the example). All other sites are assigned from the learner.

5.3. Stagnation Prevention Procedure (SPP)

Heuristics for the combinatorial problems suffer from the stagnation of the search. It is possible to define various methods to prevent this drawback, depending on the diversification and intensification implementations. In our SPP,

Algorithm 3: Teaching-learning-based-optimization-Robust Tabu Search (TLBO-RTS)

```
p: population;
teacher: the best individual in the population;
classmate: any individual in the population other than the teacher;
p ← generate_random_individuals (number_of_individuals);
evaluate fitness of individuals (p);
generate_population(population);
calculate_fitness_of_individuals (population);
for k:=1 to number_of_generations do
    /* Learning from Teacher */
    teacher=select_the_best_individual(p);
    for i:=1 to number_of_individuals do
        temp_individual:=learn_from_teacher (teacher, individual_i);
        if (temp_individual is better than individual_i )
         individual_i:=temp_individual;
    eliminate_duplicate_individuals_and_replace_with_random_individuals(p);
    /* Learning from Classmates */
    for i:=1 to number_of_individuals do
        select_a_random_classmate(classmate);
        temp_individual:=learn_from_classmate (classmate, individual_i);
        if (temp_individual is better than individual_i )
         individual_i:=temp_individual;
    eliminate_duplicate_individuals_and_replace_with_random_individuals(p);
    Stagnation Prevention Procedure (teacher); // Algorithm 4
for i:=1 to number_of_individuals do
    Robust Tabu Search(individual_i); /* see Algorithm 2 for Robust Tabu Search; */
return the best result in the population;
```

we apply mutation operator as a diversification tool to the locally optimal teachers. The teachers tend to be stuck in local optima during early stages of the optimization. If the teacher individual is not improved in m=3 successive generations, SPP is applied to the teacher. This diversification mechanism is proposed in (Dokeroglu, Tosun, & Cosar, 2013).

SPP works on teacher individual and uses the mutation operator. It starts from the leftmost facility of the teacher and swaps the current facility with the following one until it reaches to the end of the individual. The next loop starts from the second leftmost facility of the teacher and swaps the current facility through to the end of the solution and this continues until all the facilities have been swapped. Before each mutation operation, the new solution is set to the teacher so that the flow of the search continues in the teacher. Every new solution tries to replace the teacher (see Algorithm 4).

Algorithm 4: Stagnation Prevention Procedure

```
1  n:=problem size;
2  itr1:=0;
3  itr2:=0;
4  while itr1 ≤ (n-2) do
5      itr2:=itr1+1;
6      while itr2 ≤ (n-1) do
7          solution:=teacher;
8          Swap Facilities (solution, itr1, itr2);
9          Evaluate Fitness (solution);
10         if solution < teacher then teacher:= solution;
11         itr2++;
12     itr1++;
```

5.4. Parallel TLBO-RTS algorithm

In addition to the sequential TLBO-RTS algorithms, we have also implemented a simple parallel TLBO-RTS algorithm. With this algorithm, we aim to provide better results rather than decreasing the execution time of fitness evaluation. The parallel TLBO-RTS works on multiple island processors with larger total number of learners to explore the search space with more individuals that diversify to different dimensions. In order to generate distinctive populations at each processors, we use a random number generator that is seeded with the number of the processor. This mechanism produce diversification of starting populations at each island processor. Slave nodes generate their own population and execute the TLBO and RTS phases separately. Later they send their best result to the master node. The master processor selects the global best solution. It is possible to work with larger number of individuals and RTS iterations by using the parallel version of the TLBO-RTS algorithm to provide better solutions (see Algorithm 5).

Algorithm 5: Parallel TLBO-RTS

```
1  if Master Node then
2      for rank:=1 to number_of_slaves do
3          receive the best individual from slave[rank];
4  if Slave Node then
5      generate population;
6      execute TLBO phase;
7      execute RTS phase; /* see algorithm 3 for details*/
8      send the best result in the population to the master;
```

6. Experimental Results and Discussion

In this section, we present the results of the experiments obtained with the proposed sequential and parallel TLBO-RTS algorithms. 126 problem instances given in the QAPLIB are solved (Burkard, Karisch, & Rendl, 1991). Most of the state-of-the-art QAP solution algorithms are tested on these problem instances therefore, QAPLIB provides a fair environment to compare our solutions with the other algorithms in the literature. Problem instances of QAPLIB are derived either from real life applications or randomly generated problem instances such as the hospital layout (kra30), Manhattan distances of rectangular grids (Had12), and the backboard wiring (Ste36a).

First, we present our results on the global parameter settings of TLBO (number of individuals, generations) and RTS phases of the algorithm. Later, we give the results of our experiments on 55 problem instances (that are larger than 30 facilities/locations) with three proposed algorithms. We have selected the best performing algorithm, analyzed its performance on the four types of problem instances categorized by Stützle (2006). Later, the results of the parallel TLBO-RTS algorithm on the problem instances that are not solved optimally by the sequential version are given. In the last phase of the experiments, we compare our results to those of the recent state-of-the-art metaheuristics in terms of solution quality and computational effort and discuss the robustness and scalability issues of the proposed algorithms.

6.1. Experimental environment

Experiments with sequential versions of our algorithms are performed on a personal computer with Intel I5 3470 (3.20 GHz) processor and 8 GB RAM. Parallel algorithms are performed on a High Performance Cluster (HPC) computer which has 46 nodes, each with 2 CPUs giving 92 CPUs. Each CPU has 4 cores giving a total of 368 cores. Each node has 16GB of RAM giving 736 GB of total memory. High-bandwidth communication is available among nodes, using two 24 port Gbps ethernet switches, and one 24 port infiniband switch, providing very low latency messaging with a capacity of 8Gbps. C++ and the MPI libraries (Kumar, 2002) are used during the development.

6.2. Tuning the global parameters of TLBO phase

Before the experiments, we need to tune global parameters for the TLBO phase of the algorithms (the number of individuals and the number of generations). Tests are performed 10 times with generations up to 100 and the average values are reported. First, we made experiments to find the (near-)optimal number of individuals in the population. Figure 6 presents the results of the experiments ranging from 10 to 80 individuals (with SPX recombination operator). Small number of individuals such as 10 are observed to stuck into a local optimum. Larger number of individuals perform well but they have negative effect on the optimization time of the algorithms.

Test results to find the optimal number of generations are reported in Figure 8. All of the recombination operators improve their solution quality as the number of generations increases. 100 generations is set to be the (near-)optimal value for our experiments where the algorithms tend to slow down their improvements after 80 generations.

The total number of fitness evaluations performed during the optimization is the sum of evaluations in the TLBO and RTS phases of the algorithm. In the TLBO phase, the number of evaluations is equal to ($2 \times$ number_of_generation $\times$ number_of_individuals) and in the RTS phase, the number of fitness evaluations depends on the maximum number of failures. Five different parameter settings are used in the TLBO-RTS algorithms to take advantage of different parameters (see Table 1). Mutation ratio is selected as 1% for all parameter settings. The parameters presented in Table 2 give the settings for RTS phase of the algorithms.

6.3. Tuning the parameters for Robust Tabu Search phase

Maximum number of failures, tabu tenure lower/upper limits, and aspiration value are the parameters that must be well tuned for the RTS phase. We have used parameter settings that were proposed by Taillard (1991). Maximum number of failures has an important effect on the solution quality of the algorithms. Larger maximum number of failures (with well tuned tabu tenure lower/upper limits and aspiration value) provide good results however, they have longer execution times. Figure 7 gives the improvement of randomly generated 40 individuals by using RTS algorithm with increasing number of iterations for 'sko100a' instance. The experiments range from 1,000 up to 200,000 iterations. The average percentage deviation is 0.48% with 1,000 iterations and it improves to 0.08% with

200,000 iterations. In order to provide adaptivity in accordance with the problem size (n), the iteration parameter is obtained by multiplying the problem size with constant numbers such as ($2{,}000 \times n$).

The settings given in Tables 1 and 2 take advantage of different parameters for the proposed algorithms. The number of generations is set to 100 for all of the parameter settings, which is observed to be a (near-)optimal value for the algorithms and the number of maximum failures depends on the problem size, (n). Setting 1 provides a fast converging choice with smaller number of individuals and RTS iterations, setting 2 increases the number of maximum failures and spends more time to provide better results for hard problem instances such as Tai*a, setting 3 works with smaller number of maximum failures to produce solutions earlier than setting 2, setting 4 increases the number of the processors for better results, and setting 5 reduces the number of maximum failures to produce faster solutions for very large problem instances ($n \geq 150$) while still benefiting from larger number of multiprocessors.

6.4. Selecting the best performing TLBO-RTS algorithm

Three TLBO-RTS algorithms (TLBO-SPX, TLBO-OBX, and TLBO-COHX) are tested with 55 different problem instances (with sizes larger than 30) to select the best performing one. The results are presented in Table 3 (BKS=Best Known Solution of the problem instance in the QAPLIB, Found=the number of solutions that are equal to BKS out of 10 runs, APD=Average Percentage Deviation of the obtained results from the best results, BPD=Best Percentage Deviation, min.=optimization time of the algorithm in minutes). 31 of the 55 problem instances are found optimally. TLBO-OBX is selected as the best performing algorithm (with small differences). The best overall APD and BPD values found by TLBO-OBX are 0.089 and 0.078 respectively. It is observed that the optimization time of the algorithms increases in accordance with the size of the problem instances (n) and the most time consuming part of the algorithms is the RTS phase (depending on the value of maximum number of failures).

6.5. Additional computational analysis

After selecting the best performing TLBO-RTS algorithm (TLBO-OBX), we have extended the set of problem instances up to 126. The instances are classified according to the four categories given by Stützle (2006).

Type 1. *Unstructured, randomly generated instances* have distance matrix that is randomly generated based on a uniform distribution.

Type 2. *Instances with Grid-based distances* contain instances in which the distances are the Manhattan distance between points on a grid.

Type 3. *Real-life instances* are produced from real-life QAP applications.

Type 4. *Real-life-like instances* are generated instances that are similar to real-life QAP problems.

Tables 4-7 present the results of our experiments on 126 QAP problem instances by their categories. The results show that TLBO-RTS algorithm performs well. The APD percentage of the solutions was between 0.000-0.190 %. The optimization time is between 0.1 and 119 minutes for problem sizes 12 and 150, respectively. TLBO-RTS performed well for all of the problem categories. The best performance is observed on Type-3 instances with 0 % APD. TLBO-RTS has the worst performance on Type-1 instances with 0.190% APD due to the difficulty of solving Tai*a instances.

We have performed experiments with parallel TLBO-RTS algorithm on 24 problem instances (the optimal solutions were not found with our sequential algorithms for these problems). We have increased the number of individuals in the population, the number of RTS iterations, and the number of processors (the parameter settings of 4 and 5 given in Table 1). Each processor consists of 30-50 individuals which means that the overall population is increased as many as the number of processors times the number of the individuals. The number of RTS iterations was between ($2{,}000 \times n$) and ($5{,}000 \times n$). We have used 40-50 processors during the experiments and improved the solution quality of all of the 24 instances with parallel TLBO-RTS algorithm. 13 of the 24 instances are solved optimally. The results can be seen in Table 8. There are only 11 instances that we have not solved optimally in the QAPLIB. Tai*a instances were the hardest problems for our algorithms. For 24 problem instances, the overall APD and the average optimization are observed as 0.099% and 341.1 minutes respectively.

6.6. Comparison with the state-of-the-art metaheuristics

TLBO is a population based metaheuristic which uses a group of individuals to proceed for the optimum solution. There are many parameters that affect the performance of the algorithms. PSO requires learning factors, variation of weight, and maximum value of velocity; GA needs optimal crossover probability, mutation rate, and selection method; ABC needs optimal number of employed bees, onlooker bees and value of limit; ACO requires exponent parameters, pheromone evaporation rate and reward factor; HS needs optimal harmony memory consideration rate, pitch adjusting rate, and number of improvisations. Unlike the other population based algorithms TLBO does not require any algorithm-specific parameters to be tuned, this makes the implementation of our TLBO-RTS algorithm simpler. TLBO makes use of the best solution of the population to change the existing solution thereby increasing the convergence rate. TLBO does not divide the population into subpopulations and implements greediness to accept the good solution like ABC algorithm.

In the last part of our experiments, we compare the results of TLBO-RTS algorithm with the best performing sequential and parallel state-of-the-art algorithms in literature. The sequential algorithms we make comparisons with are:
Ant Colony Optimization GA/Local Search Hybrid ACO/GA/LS (Tseng, & Liang, 2005),
GA Hybrid with Concentric TS Operator GA/C-TS (Drezner, 2005),
GA Hybrid with a Strict Descent Operator GA/SD (Drezner, 2005),
Multi-Start TS Algorithm JRG-DivTS (James, Rego, & Glover, 2006),
Lagrangian Smoothing Algorithm (LagSA) (Xia, 2010)
Self Controlling Tabu Search (SC-Tabu) (Fescioglu-Unver et al.,2011)
Iterated Tabu Search (ITS) (Misevičius, A., 2012)
Migrating Birds Optimization (MBO) (Duman et al., 2012)

The parallel algorithms we make comparisons with are:
Independent Parallel TS (TB-MTS) (Talbi, & Bachelet, 2006),
Cooperative Parallel TS Hybrid with a GA (TB-COSEARCH) (Talbi, & Bachelet, 2006),
Cooperative Parallel TS (CPTS) (James, Rego, & Glover, 2009).
Robust Island Parallel Genetic Algorithm (QAP-IPGA) (Tosun, Dokeroglu, & Cosar, 2013).
Parallel Multistart Tabu Search (PMTS) (Czapiński, M. 2013)

Tables 9-12 present the comparison of our TLBO-RTS algorithm with the state-of-the-art metaheuristics. Figure 9 is the visualized version of Table 10 for 'Skorin-Kapov' problem instances. Our algorithm is observed to be competitive with the solution quality and the execution times of the other algorithms. Although the execution time of the algorithms changes depending on the hardware (the number of processors), programming language, and compilers, we report some of their running times to give an idea to the reader. It can be seen from the results of the experiments that the execution time of our algorithm is reasonable.

We have reported the available results of the best algorithms in accordance with the Types of the problem instances in terms of APD. The results of the best four algorithms are reported in bold face. For Type-1 problem instances (the hardest set), the APD of TLBO-RTS is 0.342 (the second best performing algorithm after ITS). We produced 5 optimal solutions out of 9. For Type-2 problems, TLBO-RTS produces 9 high quality solutions of 13 problems and is the third best performing algorithm. For Type-3 problems, TLBO-RTS gives the optimal solutions for all of the problem instances. For Type-4 problems, TLBO-RTS is the second best performing algorithm in the literature. As a result, our study illustrates that TLBO-RTS performs well on all of the problem instances in the QAPLIB and competitive with the other metahuristics.

The sequential version of TLBO-RTS algorithm has solved 102 of the 126 problem instances optimally. We have performed tests with parallel TLBO-RTS on the 24 problem instances that were not solved optimally. We were able to find additional 13 optimal solutions with the parallel TLBO-RTS algorithm. The total number of problems solved optimally by our proposed algorithm reach to 115, which is (91.3%) of the 126 problem instances. The results are reported in Table 8.

In order to illustrate the capabilities of our parallel TLBO-RTS algorithm, we have compared its results to the other parallel algorithms in the literature. Table 13 contains a summary of comparisons between our parallel TLBO-RTS

algorithm and the Cooperative Parallel TS (CPTS), Independent Parallel TS (TB-MTS), Cooperative TS hybrid with a GA (TB-COSEARCH), Robust Island Parallel Genetic Algorithm (QAP-IPGA), and Parallel Multistart Tabu Search (PMTS). To the best of our knowledge, these algorithms are the only published parallel metaheuristic algorithms. The other parallel algorithms from the literature either work on different problem domains or only report on single run results related to timing and solution quality, which is usually the best case result for that algorithm. We have performed our experiments with 40-50 processors and solved each instance 10 times to minimize the measurement errors. The results of the other algorithms are obtained from different execution environments and number of processors. CPTS uses 10 processors, TB-MTS and TB-COSEARCH use 150 processors, and QAP-IPGA uses processors ranging from 100 to 240. PMTS uses a GPU hardware with 15 multiprocessors and 32 scalar processor cores each. Our parallel TLBO-RTS is observed to be among the best performing parallel algorithms in the literature in accordance with the APD values of the selected problem instances.

6.7. Analysis of the robustness and scalability of the algorithms

Since the proposed TLBO-RTS algorithms involve randomization, achieving robustness becomes an important issue. Each problem instance in the experiments is solved 10 times and the average of the results are reported to clarify this point. For sequential TLBO-RTS, the APD is observed to be 0.0 % for 102 problem instances. For 126 problem instances, APD is 0.001 % for parallel TLBO-RTS algorithms, which is very robust with respect to the solutions found by the other algorithms in literature. The robustness of the proposed algorithms can even become better with additional processors.

Scalability is another important aspect for the performance evaluation of all parallel algorithms and measures the capability to continue to give good results when the number of processors increases. In other words, scalability enables us to solve the problem instances without extra delays in execution time. In our proposed parallel algorithms, we aim to increase the number of processors to have a chance to explore the search space in a better way rather than speeding up the fitness evaluation process of the individuals. The optimization time is not delayed due to the messaging as we increase the number of the processors. When the number of processors is increased up to 60, a messaging overhead of 30 seconds is added to the execution time of the algorithm in our HPC configuration. The proposed algorithms are acceptable and continue to substantially benefit from increased number of processors with negligible delays.

7. Conclusions and Future Work

In this study, we propose a set of hybrid Teaching-learning-based-optimization (TLBO) algorithms for the solution of the challenging Quadratic Assignment Problem (QAP). The algorithm-specific parameterless concept of TLBO provides an easy-to-use approach and there is no need to apply any selection mechanism. Recombination operators of the proposed algorithms do not require any fine tuning and our algorithms work well with random settings. In addition to these advantages, the performance of the proposed sequential and parallel hybrid TLBO-RTS algorithms are shown to be very competitive and among of the best performing algorithms for the QAP. There has been no TLBO-based solution method for the QAP. From this aspect, this is the first study to show how TLBO-based algorithms can be applied to the QAP, while also proposing interesting ideas on how parallel processing can be exploited for solving difficult QAP instances.

As future work, we plan to apply well-known machine learning techniques such as reinforcement learning, inductive logic programming, and support vector machines to train the learners. Also, multiobjective combinatorial problems such as two/three-dimensional bin packing can be modeled and solved by using TLBO.

Adl, R.K. & Rankoohi, S.M.T.R. (2009). A new ant colony optimization based algorithm for data allocation problem in distributed databases, Knowledge Information Systems, vol. 20, no. 3, pp. 349-373.

Ahuja, R.K., Orlin, J.B., & Tiwari, A. (2000). A greedy genetic algorithm for the quadratic assignment problem. Computers and Operations Research, Vol.27, 917-934.

Ahrari, A., & Atai, A.A. (2010). Grenade Explosion Method: A novel tool for optimization of multimodal functions. Applied Soft Computing, 10(4), 1132-1140.

Anstreicher, K., Brixius, N.W., Goux, J-P., & Linderoth, J. (2002) Solving large quadratic assignment problems on computational grids, Mathematical Programming, vol. 91, no. 3, pp. 563-588.

Battiti, R. & Tecchiolli, G. (1994) The reactive tabu search, ORSA Journal on Computing, vol. 6, no. 2, pp. 126-140.
Burkard, R.E., Karisch, S.E., & Rendl, F. (1991) QAPLIB a quadratic assignment problem library, European Journal of Operational Research, vol. 55, no. 1, pp. 115-119.
Burkard R.E. & Cela, E. (1999) Linear assignment problems and extensions, In Pardalos, P. and Du, D.-Z. (eds), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Supplement vol. A, pp. 75-149.
Calzon-Bousono, C. (1995) The hopfield neural network applied to the quadratic assignment problem, Neural Computing and Applications, vol. 3, no. 2, pp. 64-72.
Carraresi, P. & Malucelli, F. (1992) A new lower bound for the quadratic assignment problem, Operations Research, vol. 40, no. 1, pp. 22-27.
Chakrapani, J. & Skorin-Kapov, J. (1992) A connectionist approach to the quadratic assignment problem, Computers and Operations Reasearch, vol. 19, no 3-4, pp. 287-295.
Clausen, J. & Perregaard, M. (1997) Solving large quadratic assignment problems in parallel, Computational Optimization and Applications, vol. 8, no. 2, pp. 111-127.
Connolly, D.T. (1990) An improved annealing scheme for the QAP, European Journal of Operational Research, vol. 46, no. 1, pp. 93-100.
Cordeau, J-F., Gaudioso, Laporte, M.G., & Moccia, L. (2007) The service allocation problem at the Gioia Tauro Maritime Terminal, European Journal of Operational Research, vol. 176, no. 2, pp. 1167-1184.
Crainic, T.G. & Toulouse, M. (2003) Parallel strategies for metaheuristics. In: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics. Kluwer Academic Publishers.
Črepinšek, M., Liu, S.H., & Mernik, L. (2012) A note on teaching-learning-based optimization algorithm. Information Sciences, 212, 79-93.
Czapiński, M. (2013). An effective parallel multistart tabu search for quadratic assignment problem on CUDA platform. Journal of Parallel and Distributed Computing, 73(11), 1461-1468.
Dokeroglu, T., & Cosar, A. (2014). Optimization of one-dimensional Bin Packing Problem with island parallel grouping genetic algorithms. Computers & Industrial Engineering, 75, 176-1786.
Dokeroglu, T., Tosun, U., & Cosar, A. (2012) Parallel Optimization with Mutation Operator for the Quadratic Assignment Problem, In Proceedings of WIVACE 2012, Italian Workshop on Artificial Life and Evolutionary Computation, Parma/Italy.
Dokeroglu, T., Tosun, U., & Cosar, A. (2013) Evaluating the Performance of Recombination Operators with Island Parallel Genetic Algorithms, IFAC MIM, Saint Petersburg, Russia.
Dorigo, M., & Di Caro, G. (1999) Ant colony optimization: a new metaheuristic. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 2).
Drezner, Z. (2003) A new genetic algorithm for the quadratic assignment problem, INFORMS Journal on Computing, vol. 15, no. 3, pp. 320-330.
Drezner, Z. (2005) The extended concentric tabu for the quadratic assignment problem, European Journal of Operational Research, vol. 160, no. 2, pp. 416-422.
Duman, E., Uysal, M., & Alkaya, A. F. (2012). Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. Information Sciences, 217, 65-77.
Eusuff, M.M. & Lansey, K.E. (2003) Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management, 129(3), 210-225.
Fiechter, C.N. (1994) A parallel tabu search algorithm for large traveling salesman problems, Discrete Applied Mathematics vol. 51, no. 3, pp. 243-267.
Fescioglu-Unver, N., & Kokar, M. M. (2011). Self controlling tabu search algorithm for the quadratic assignment problem. Computers & Industrial Engineering, 60(2), 310-319.
Gambardella, L.M. & Taillard, E.D., & Dorigo, M. (1999) Ant colonies for the quadratic assignment problem, Journal of the Operational Research Society, vol. 50, no. 2, pp. 167-176.
Geem, Z. W., Kim, J.H., & Loganathan, G. V. (2001) A new heuristic optimization algorithm: harmony search. Simulation, 76(2), 60-68.
Gilmore, P.C. (1962) Optimal and suboptimal algorithms for the quadratic assignment problem, Journal of the Society of Industrial and Applied Mathematics, vol. 10, no. 2, pp. 305-313.
Goldberg, D. (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, Mass.
Hahn, P. & Krarup, J. (2001) A hospital facility layout problem finally solved, Journal of Intelligent Manufacturing, vol. 12, no. 5-6, pp. 487-496.
Holland, J.H. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, USA.
James, T., Rego, C., & Glover, F. (2005) Sequential and parallel pathrelinking algorithms for the quadratic assignment problem, IEEE Intelligent Systems, vol. 20, no. 4, pp. 58-65.
James, T., Rego, C., & Glover, F. (2006) Multi-start tabu search and diversification strategies for the quadratic assignment problem, Working Paper, Virginia Tech.
James, T., Rego, C., & Glover, F. (2009) A cooperative parallel tabu search algorithm for the QAP, European Journal of Operational Research, vol. 195, no. 3, pp. 810-826.
Karaboga, D., & Basturk, B. (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39(3), 459-471.
Kelly, J.P., Laguna, M., & Glover, F. (1994) A study of diversification strategies for the quadratic assignment problem, Computers and Operations Research, vol 21, no 8, pp.885-893.
Kennedy, J., & Eberhart, R. (1995) Particle swarm optimization. In Neural Networks, 1995. Proceedings., IEEE International Conference on (Vol. 4, pp. 1942-1948).
—
Koopmans, T.C. & Beckmann, M.J. (1957) Assignment problems and the location of economic activities, Econometrica, vol. 25, no. 1, pp. 53-76.
Kumar, V. (2002) Introduction to Parallel Computing. Addison-Wesley.

Laguna, M., Marti, R., & Campos, V. (1999) Intensification and diversification with elite tabu search solutions for the linear ordering problem, Computers and Operations Research, vol. 26, no. 12, pp. 1217-1230.
Lawler, E.L. (1963) The quadratic assignment problem, Management Science, vol. 9, no. 4, pp. 586-599.
Lawrence, D. (1991) Order-based genetic algorithms and the graph coloring problem. Handbook of genetic algorithms: 72-90.
Li, Y., Pardalos, P.M., & Resende, M.G.C. (1994) A greedy randomized adaptive search procedure for the quadratic assignment problem, Quadratic assignment and related problems, P. M. Pardalos & H. Wolkowicz, eds., DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 16, pp. 237-261, 1994.
Lim, M.H., Yuan,Y., & Omatu, S. (2000) Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem, Computational Optimization and Applications, vol. 15, no. 3, pp. 249-268.
Luque G. & Alba E. (2011) Parallel Genetic Algorithms, Theory and Applications. Springer.
Marzetta, A. & Brungger, A. (1999) A dynamic-programming bound for the quadratic assignment problem, Lecture Notes in Computer Science, vol. 1627, pp. 339-348.
Mautor, T. & Roucairol, C. (1994) A new exact algorithm for the solution of quadratic assignment problems, Discrete Applied Mathematics, vol. 55, no. 3, pp. 281-293.
Misevičius, A. (2003) Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem, Knowledge-Based Systems, vol. 16, pp. 261-268.
Misevičius, A. (2005) A tabu search algorithm for the quadratic assignment problem, Computational Optimization and Applications, vol. 30, no. 1, pp. 95-111.
Misevičius, A. (2012). An implementation of the iterated tabu search algorithm for the quadratic assignment problem. OR spectrum, 34(3), 665-690.
Nystrom, M. (1999) Solving certain large instances of the quadratic assignment problem: Steinberg examples. Department of Computer Science, California Institute of Technology.
Pardalos, P.M., Rendl, F., & Wolkowicz, H. (1994) The Quadratic Assignment Problem: A Survey and Recent Developments, In Quadratic Assignment and Related Problems, P. M. Pardalos & H. Wolkowicz, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 16, pp. 1-42.
Pentico, D.W. (2007) Assignment problems: a golden anniversary survey, European Journal of Operational Research, vol. 176, no. 4, pp. 774-793.
Pfister, G.F. (1998) In Search of Clusters (2nd Edition). Prentice Hall.
Rao, R.V., Savsani, V.J., & Vakharia, D.P. (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Design 43 (3) 303-315.
Rao, R.V., Savsani, & V.J., Vakharia, D.P. (2012a) Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems, Information Sciences 183 (1) 1-15.
Rao, R.V., Savsani, V. J., & Balic, J. (2012b). Teaching-learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems. Engineering Optimization, 44(12), 1447-1462.
Rao, R.V. & Patel, V., (2012a) An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems, International Journal of Industrial Engineering Computations 3-535-560.
Rao, R.V. & Patel, V., (2012b) Multi-objective optimization of heat exchangers using a modified teaching-learning-based-optimization algorithm, Applied Mathematical Modeling.
Rao, R.V. & Patel, V., (2012c) Multi-objective optimization of combined Brayton and inverse Brayton cycle using advanced optimization algorithms, Engineering Optimization.
Rao, R.V. & Patel, V. (2012d) Multi-objective optimization of two stage thermoelectric cooler using a modified teaching-learning-based-optimization algorithm, Engineering Applications of Artificial Intelligence.
Rao, R.V. & Patel, V. (2012e) Comparative performance of an elitist teaching-learning-based optimization algorithm for solving unconstrained optimization problems, International Journal of Industrial Engineering Computations 4.
Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009) GSA: a gravitational search algorithm. Information sciences, 179(13), 2232-2248.
Rossin, D.F., Springer, M.C., & Klein, B.D. (1999) New complexity measures for the facility layout problem: An empirical study using traditional and neural network analysis, Computers and Industrial Engineering, vol. 36, no. 3, pp. 585-602.
Steinberg, L., (1961) The backboard wiring problem: A placement algorithm, SIAM Review, vol.3, no. 1, pp. 37-50.
Storn, R., & Price, K. (1997) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4), 341-359.
Stützle, T. & Dorigo, M. (1999) ACO algorithms for the quadratic assignment problem. In: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas for Optimization. McGraw-Hill, pp. 33-50.
Stützle, T. (2006) Iterated local search for the quadratic assignment problem. European Journal of Operational Research, 174(3), 1519-1539.
Taillard, E. (1991) Robust taboo search for the quadratic assignment problem, Parallel Computing, vol. 17, no. 4-5, pp. 443-455.
Talbi, E-G., Hafidi,Z., & Geib, J-M. (1997) Parallel adaptive tabu search for large optimization problems, In: MIC'97-2nd Metaheuristics International Conference, Sophia Antipolis, France.
Talbi, E-G. & Bachelet, V. (2006) COSEARCH: A parallel cooperative metaheuristic, Journal of Mathematical Modeling and Algorithms, vol. 5, no. 1, pp. 5-22.
Tate, D.M. & Smith, A.E. (1995) A genetic approach to the quadratic assignment problem, Computers and Operations Research, vol. 22, no. 1, pp. 73-83, .
Tosun, U., Dokeroglu, T., & Cosar, A. (2013) A New Robust Island Parallel Genetic Algorithm for the Quadratic Assignment Problem, International Journal of Production Research, Volume 51, Issue 14, 4117-4133.
Tseng, L. & Liang, S. (2005) A hybrid metaheuristic for the quadratic assignment problem, Computational Optimization and Applications, vol. 34, no. 1, pp. 85-113.
Tsutsui, S. & Fujimoto, N. (2009) Solving Quadratic Assignment Problems by Genetic Algorithms with GPU Computation: a Case Study,

GECCO, Montreal Quebec, Canada.
Waghmare, G. (2013) Comments on A Note on Teaching-Learning-Based Optimization Algorithm. Information Sciences 229:159-169
Xia, Y. (2010). An efficient continuation method for quadratic assignment problems. Computers & Operations Research, 37(6), 1027-1032.
Zhang, H., Beltran-Royo, C., & Constantino, M. (2010). Effective formulation reductions for the quadratic assignment problem. Computers & Operations Research, 37(11), 2007-2016.

Figures and Tables

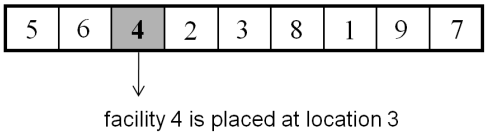

Figure 1: Representation of an individual for the quadratic assignment problem.

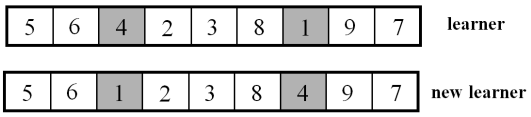

Figure 2: Mutation operator.

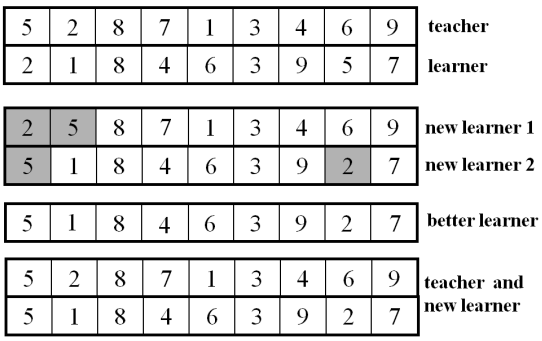

Figure 3: Swap Path Recombination operator (SPX).

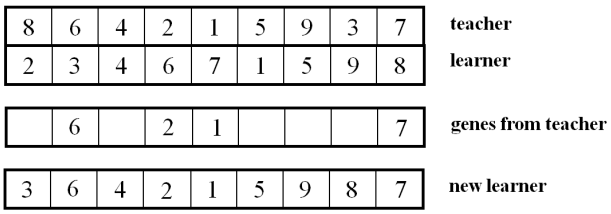

Figure 4: Order Based Recombination operator (OBX).

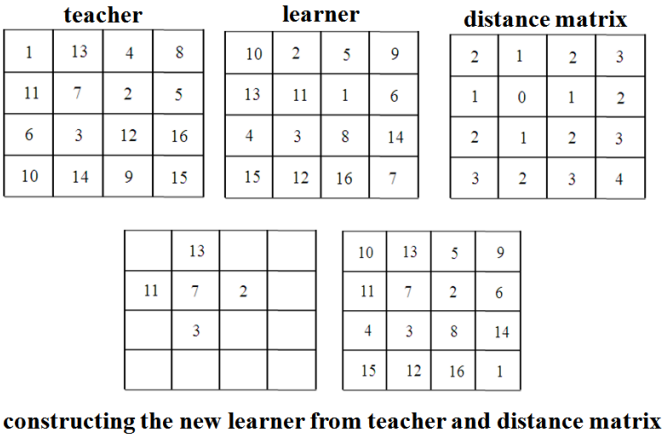

Figure 5: Cohesive Recombination operator (COHX).

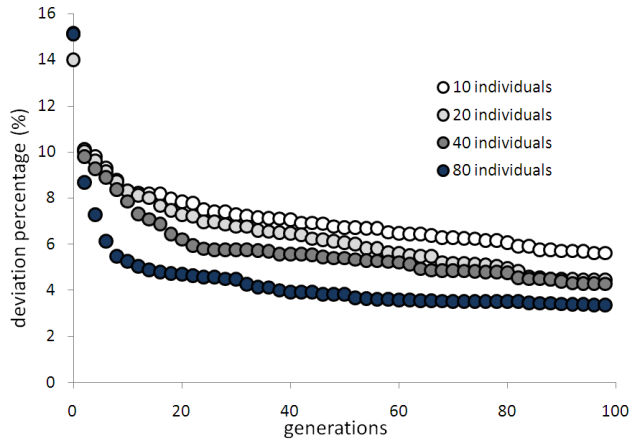

Figure 6: Population tests with Sko100a instance and SPX Recombination operator.

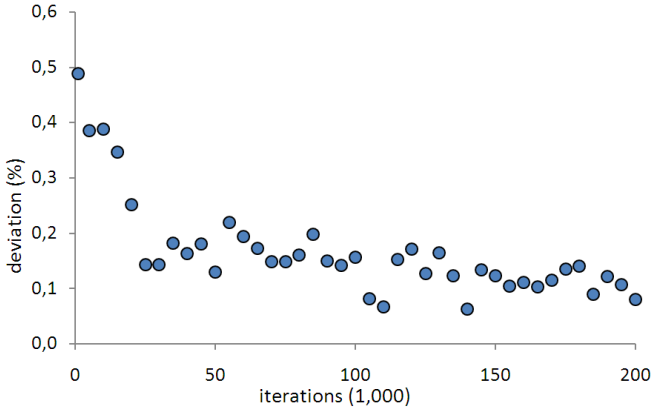

Figure 7: Robust tabu search experiments for randomly generated 40 individuals with increasing number of iterations for Sko100a instance.

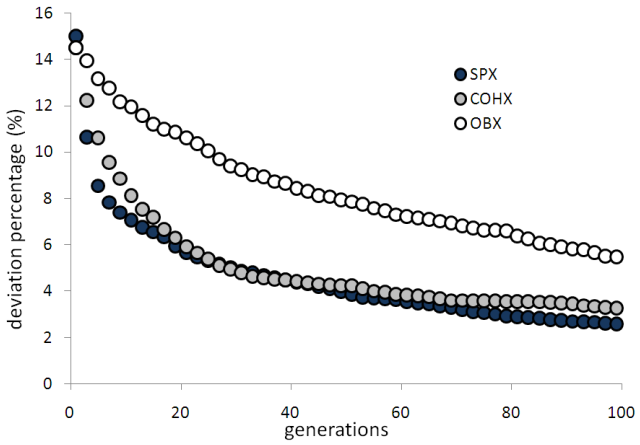

Figure 8: Generation tests with Sko100a instance.

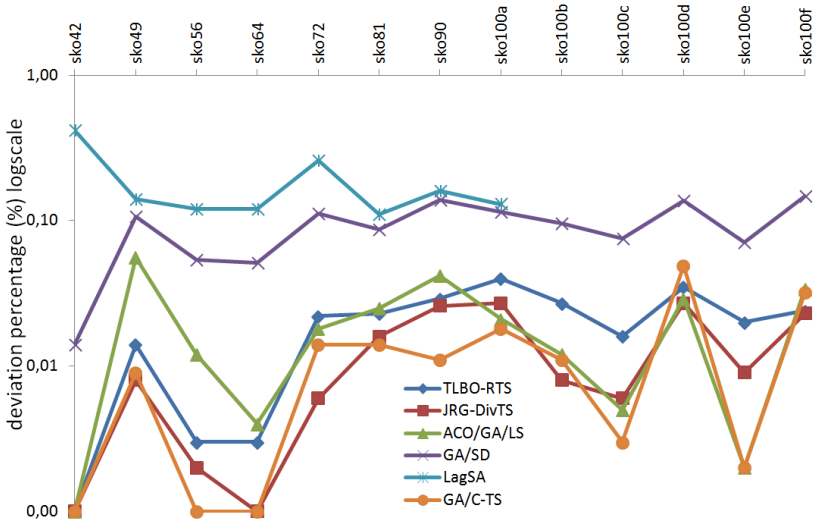

Figure 9: Vizualized results of Table 10 with sko problem instances.

Table 1: Global parameter settings for TLBO phase of the proposed algorithms

Parameter	**setting 1**	**setting 2**	**setting 3**	**setting 4**	**setting 5**
Population Size	20	20	30	50	30
Number of Generations	100	100	100	100	100
RTS phase configuration (see Table 2)	1	2	3	4	5
number of processors	1	1	1	40	50

Table 2: RTS parameter settings used in our experiments

Configuration	maximum # of failures	tabu tenure	aspiration value
1	2,000 x n	lower limit=(9×n)/10 upper limit=(11×n)/10	n×n×2
2	[400,000-1,000,000]	lower limit=(9×n)/10 upper limit=(11×n)/10	n×n×2
3	5,000 x n	lower limit=(9×n)/10 upper limit=(11×n)/10	n×n×2
4	5,000 x n	lower limit=(2×n)/10 upper limit=(18×n)/10	n×n×2
5	2,000 x n	lower limit=(2×n)/10 upper limit=(18×n)/10	n×n×2

Table 3: Results found by the proposed sequential TLBO-RTS algorithms with parameter setting 1.
(20 individuals, 100 generations, 1 processor, and 2,000 x n failures for RTS)

		TLBO-SPX				TLBO-OBX				TLBO-COHX			
Instance	BKS	Found	APD	BPD	min.	Found	APD	BPD	min.	Found	APD	BPD	min.
Esc32a	130	10	0	0	1.2	10	0	0	1.2	10	0	0	1.2
Esc32b	168	10	0	0	1.2	10	0	0	1.2	10	0	0	1.2
Esc32c	642	10	0	0	1.2	10	0	0	1.2	10	0	0	1.2
Esc64a	116	10	0	0	9.2	10	0	0	9.2	10	0	0	9.3
Esc128	64	10	0	0	75.4	10	0	0	75	10	0	0	76.3
Kra30a	88900	10	0	0	1.0	10	0	0	1.0	10	0	0	1.0
Kra30b	91420	10	0	0	1.0	10	0	0	1.0	10	0	0	1.0
Lipa30a	13178	10	0	0	1.0	10	0	0	1.0	10	0	0	1.0
Lipa30b	151426	10	0	0	1.0	10	0	0	1.0	10	0	0	1.0
Lipa40a	31538	10	0	0	2.3	10	0	0	2.3	10	0	0	2.3
Lipa40b	476581	10	0	0	2.3	10	0	0	2.3	10	0	0	2.3
Lipa50a	62093	10	0	0	4.3	10	0	0	4.2	10	0	0	4.4
Lipa50b	1210244	10	0	0	4.3	10	0	0	4.2	10	0	0	4.4
Lipa60a	107218	10	0	0	7.6	10	0	0	7.4	10	0	0	7.7
Lipa60b	2520135	10	0	0	7.6	10	0	0	7.4	10	0	0	7.7
Lipa70a	169755	10	0	0	12.5	10	0	0	12.3	10	0	0	12.7
Lipa70b	4603200	10	0	0	12.5	10	0	0	12.3	10	0	0	12.7
Lipa80a	253195	5	0.254	0	18.8	6	0.207	0	18.5	8	0.103	0	19
Lipa80b	7763962	10	0	0	18.8	10	0	0	18.5	10	0	0	19
Lipa90a	360630	6	0.193	0	25.7	4	0.281	0	25.2	4	0.199	0	26.1
Lipa90b	12490441	10	0	0	25.7	10	0	0	25.2	10	0	0	26.1
Nug30	6124	10	0	0	1.0	10	0	0	0.9	10	0	0	1.0
Sko42	15812	10	0	0	2.6	10	0	0	2.4	10	0	0	2.6
Sko49	23386	2	0.034	0	4.1	1	0.037	0	4.0	5	0.019	0	4.2
Sko56	34458	4	0.008	0	6.2	6	0.005	0	6.0	1	0.017	0	6.3
Sko64	48498	5	0.003	0	9.5	6	0.002	0	9.2	4	0.004	0	9.5
Sko72	66256	0	0.033	0.018	13.6	0	0.024	0.009	13.2	0	0.034	0.018	13.8
Sko90	115534	0	0.046	0.010	25.7	0	0.049	0.035	25.2	0	0.026	0.017	26.1
Sko100a	152002	0	0.048	0.027	36.1	0	0.064	0.050	35.7	0	0,057	0.045	36.5
Sko100b	153890	0	0.028	0.023	36.1	0	0.017	0.005	35.7	0	0,042	0.030	36.5
Sko100c	147862	0	0.019	0.012	36.1	0	0.019	0.003	35.7	0	0,032	0.020	36.5
Sko100d	149576	0	0.041	0.017	36.1	0	0.031	0.013	35.7	0	0,058	0.038	36.5
Sko100e	149150	0	0.026	0.019	36.1	0	0.023	0.013	35.7	0	0,025	0.008	36.5
Sko100f	149036	0	0.062	0.015	36.1	0	0.047	0.031	35.7	0	0.049	0.036	36.5
Tai30a	1818146	10	0	0	1.0	10	0	0	1.0	10	0	0	1.0
Tai30b	637117113	10	0	0	1.0	10	0	0	1.0	10	0	0	1.0
Tai35a	2422002	5	0.150	0	1.5	1	0.244	0	1.4	5	0.099	0	1.5
Tai35b	283315445	10	0	0	1.5	10	0	0	1.4	10	0	0	1.5
Tai40a	3139370	0	0.440	0.219	2.3	0	0.446	0.278	2.2	0	0.515	0.331	2.3
Tai40b	637250948	10	0	0	2.3	10	0	0	2.2	10	0	0	2.3
Tai50a	4938796	0	0.961	0.809	4.4	0	0.892	0.810	4.2	0	0.934	0.645	4.5
Tai50b	458821517	10	0	0	4.4	10	0	0	4.2	10	0	0	4.5
Tai60a	7205962	0	1.098	0.899	7.6	0	1.039	0.928	7.2	0	1.044	0.915	7.7
Tai60b	608215054	10	0	0	7.6	10	0	0	7.2	10	0	0	7.7
Tai64c	1855928	10	0	0	9.2	10	0	0	8.8	10	0	0	9.3
Tai80a	13499184	0	1.124	1.079	18.8	0	1.185	1.082	17.5	0	1.189	1.106	19
Tai80b	818415043	0	0.004	0.001	18.8	0	0.005	0.002	17.5	4	0.007	0	19
Tai100a	21052466	0	1.091	1.020	36.1	0	1.041	0.912	35.7	0	1.143	1.106	36.5
Tai100b	1185996137	0	0.039	0.001	36.1	2	0.027	0	35.7	1	0,032	0	36.5
Tai150b	498896643	0	0.119	0.086	123	0	0.095	0.068	119	0	0.074	0.019	123.8
Tho30	149936	10	0	0	1.0	10	0	0	1.0	10	0	0	1.0
Tho40	240516	7	0.003	0	2.3	10	0	0	2.2	10	0	0	2.3
Tho150	8133398	0	0.060	0.047	123	0	0.060	0.046	119	0	0.073	0.052	123.7
Wil50	48816	10	0	0	4.3	10	0	0	4.1	10	0	0	4.5
Wil100	273038	0	0.010	0.004	36.1	0	0.010	0.004	35.7	0	0.014	0.005	36.5
Average		6.07	0.107	0.078	17.39	6.10	**0.089**	**0.078**	**17.33**	**6.22**	0.105	0.079	17.49

Table 4: Results found for Type 1 problem instances with parameter setting 1 using OBX recombination operator. (20 individuals, 100 generations, 1 processor, and 2,000 x *n* failures for RTS)

Unstructured, randomly generated instances					
Instance	BKS	Found	APD	BPD	min.
Rou12	235528	10	0	0	0.1
Rou15	354210	10	0	0	0.1
Rou20	725522	10	0	0	0.3
Lipa20a	27076	10	0	0	0.3
Lipa20b	13178	10	0	0	0.3
Lipa30a	13178	10	0	0	1.0
Lipa30b	151426	10	0	0	1.0
Lipa40a	31538	10	0	0	2.3
Lipa40b	476581	10	0	0	2.3
Lipa50a	62093	10	0	0	4.3
Lipa50b	1210244	10	0	0	4.3
Lipa60a	107218	10	0	0	7.6
Lipa60b	2520135	10	0	0	7.6
Lipa70a	169755	10	0	0	12.5
Lipa70b	4603200	10	0	0	12.5
Lipa80a	253195	5	0.254	0	18.8
Lipa80b	7763962	10	0	0	18.8
Lipa90a	360630	6	0.193	0	25.7
Lipa90b	12490441	10	0	0	25.7
Tai20a	703482	10	0	0	0.3
Tai25a	1167256	10	0	0	0.3
Tai30a	1818146	10	0	0	1.0
Tai35a	2422002	5	0.150	0	1.5
Tai40a	3139370	0	0.440	0.219	2.3
Tai50a	4938796	0	0.961	0.809	4.4
Tai60a	7205962	0	1.098	0.899	7.6
Tai80a	13499184	0	1.124	1.079	18.8
Tai100a	21052466	0	1.091	1.020	36.1
Average		7.71	**0.190**	**0.144**	7.77

Table 5: Results found for Type 2 problem instances with parameter setting 1 using OBX recombination operator. (20 individuals, 100 generations, 1 processor, and 2,000 x *n* failures for RTS)

Instances with Grid-Distances					
Instance	BKS	Found	APD	BPD	min.
Nug12	578	10	0	0	0.1
Nug14	1014	10	0	0	0.1
Nug15	1150	10	0	0	0.1
Nug16a	1610	10	0	0	0.2
Nug16b	1240	10	0	0	0.2
Nug17	1732	10	0	0	0.2
Nug18	1930	10	0	0	0.2
Nug20	2570	10	0	0	0.3
Nug21	2438	10	0	0	0.3
Nug22	3596	10	0	0	0.4
Nug24	3488	10	0	0	0.5
Nug25	3744	10	0	0	0.5
Nug27	5234	10	0	0	0.7
Nug28	5166	10	0	0	0.8
Nug30	6124	10	0	0	0.9
Sko42	15812	10	0	0	2.4
Sko49	23386	1	0.037	0	4.0
Sko56	34458	6	0.005	0	6.0
Sko64	48498	6	0.002	0	9.2
Sko72	66256	0	0.024	0.009	13.2
Sko90	115534	0	0.049	0.035	25.2
Sko100a	152002	0	0.064	0.050	35.7
Sko100b	153890	0	0.017	0.005	35.7
Sko100c	147862	0	0.019	0.003	35.7
Sko100d	149576	0	0.031	0.013	35.7
Sko100e	149150	0	0.023	0.013	35.7
Sko100f	149036	0	0.047	0.031	35.7
Scr12	31410	10	0	0	0.1
Scr15	51140	10	0	0	0.1
Scr20	110030	10	0	0	0.3
Tho30	149936	10	0	0	1.0
Tho40	240516	10	0	0	2.2
Tho150	8133398	0	0.060	0.046	119
Wil50	48816	10	0	0	4.3
Wil100	273038	0	0.010	0.004	36.1
Average		6,37	**0.011**	**0.00**6	12.65

Table 6: Results found for Type 3 problem instances with parameter setting 1 using OBX recombination operator. (20 individuals, 100 generations, 1 processor, and 2,000 x n failures for RTS)

Real-life instances											
Instance	BKS	Found	APD	BPD	min.	Instance	BKS	Found	APD	BPD	min.
Bur26a	5426670	10	0	0	0.6	Kra30a	88900	10	0	0	1.0
Bur26b	3817852	10	0	0	0.6	Kra30b	91420	10	0	0	1.0
Bur26c	5426795	10	0	0	0.6	Kra32	88700	10	0	0	1.2
Bur26d	3821225	10	0	0	0.6	Ste36a	9526	10	0	0	1.6
Bur26e	5386879	10	0	0	0.6	Ste36b	15852	10	0	0	1.6
Bur26f	3782044	10	0	0	0.6	Ste36c	8239110	10	0	0	1.6
Bur26g	10117172	10	0	0	0.6	Esc16a	68	10	0	0	0.1
Bur26h	7098658	10	0	0	0.6	Esc16b	292	10	0	0	0.1
Els19	17212548	10	0	0	0.2	Esc16c	160	10	0	0	0.1
Had12	1652	10	0	0	0.1	Esc16d	16	10	0	0	0.1
Had14	2724	10	0	0	0.1	Esc16e	28	10	0	0	0.1
Had16	3720	10	0	0	0.1	Esc16f	0	10	0	0	0.1
Had18	5358	10	0	0	0.2	Esc16g	26	10	0	0	0.1
Had20	6922	10	0	0	0.3	Esc16h	996	10	0	0	0.1
Chr12a	9552	10	0	0	0.1	Esc16i	14	10	0	0	0.1
Chr12b	9742	10	0	0	0.1	Esc16j	8	10	0	0	0.1
Chr12c	11156	10	0	0	0.1	Esc32a	130	10	0	0	1.2
Chr15a	9896	10	0	0	0.1	Esc32b	168	10	0	0	1.2
Chr15b	7990	10	0	0	0.1	Esc32c	642	10	0	0	1.2
Chr15c	9504	10	0	0	0.1	Esc32d	200	10	0	0	1.2
Chr18a	11098	10	0	0	0.2	Esc32e	2	10	0	0	1.2
Chr18b	1534	10	0	0	0.2	Esc32g	6	10	0	0	1.2
Chr20a	2192	10	0	0	0.3	Esc32h	438	10	0	0	1.2
Chr20b	2298	10	0	0	0.3	Esc64a	116	10	0	0	9.2
Chr20c	14142	10	0	0	0.3	Esc128	64	10	0	0	75.0
Chr22a	6156	10	0	0	0.4						
Chr22b	6194	10	0	0	0.4						
Chr25a	3796	10	0	0	0.5						
Average								10	**0**	**0**	2.09

Table 7: Results found for Type 4 problem instances with parameter setting 1 using OBX recombination operator. (20 individuals, 100 generations, 1 processor, and 2,000 x n failures for RTS)

Real-life-like instances					
Instance	BKS	Found	APD	BPD	min.
Tai20b	122455319	10	0	0	0.26
Tai25b	344355646	10	0	0	0.55
Tai30b	637117113	10	0	0	1.0
Tai35b	283315445	10	0	0	1.4
Tai40b	637250948	10	0	0	2.2
Tai50b	458821517	10	0	0	4.2
Tai60b	608215054	10	0	0	7.2
Tai80b	818415043	0	0.005	0.002	17.5
Tai100b	1185996137	2	0.027	0	35.7
Average		8	**0.003**	**0**	7.77

Table 8: Parallel TLBO-RTS algorithm results (parameter setting 4 uses 50 individuals, 100 generations, 5,000 x n failures for RTS, and 40 processors, parameter setting 5 uses 30 individuals, 100 generations, 2,000 x n failures for RTS and 50 processors)

Instance	BKS	APD	BPD	min.	par. set.
Lipa80a	253195	**0**	**0**	239.8	4
Lipa90a	360630	**0**	**0**	360.7	4
Sko49	23386	**0**	**0**	54.1	4
Sko56	34458	**0**	**0**	81.6	4
Sko64	48498	**0**	**0**	119.3	4
Sko72	66256	**0**	**0**	170.8	4
Sko90	115534	**0**	**0**	342.8	4
Sko100a	152002	0.003	**0**	594.3	4
Sko100b	153890	0.005	**0**	482.6	4
Sko100c	147862	**0**	**0**	508.5	4
Sko100d	149576	0.009	0.007	509.4	4
Sko100e	149150	0.005	**0**	614.5	4
Sko100f	149036	0.005	**0**	482.6	4
Tai35a	2422002	**0**	**0**	18.3	4
Tai40a	3139370	**0**	**0**	29.0	3
Tai50a	4938796	0.360	0.321	55.0	4
Tai60a	7205962	0.410	0.388	95.3	4
Tai80a	13499184	0.870	0.850	239.5	4
Tai80b	818415043	**0**	**0**	239.0	4
Tai100a	21052466	0.596	0.575	483.3	5
Tai100b	1185996137	**0**	**0**	508.2	4
Tai150b	498896643	0.015	0.011	428.5	5
Tho150	8133398	0.030	0.027	556.6	5
Wil100	273038	**0**	**0**	482.6	4
Average		**0.099**	**0.091**	341.1	

Table 9: Comparison of the TLBO-based hybrid algorithms with state-of-the-art algorithms on Type-1 problem instances with parameter setting 2 (20 individuals, 100 generations, 1 processor, and [400,000-1,000,000] failures for RTS).

		TLBO-RTS		JRG-DivTS		ITS	SC-Tabu		ACO/GA/LS	
Instance	BKS	APD	min.	APD	min.	APD	APD	min.	APD	min.
Tai20a	70382	**0**	5.2	**0**	0.2	**0**	0.246	0.001	-	-
Tai25a	1167256	**0**	8.3	**0**	0.2	**0**	0.239	0.03	-	-
Tai30a	1818146	**0**	12.1	**0**	1.3	**0**	0.154	0.07	0.341	1.4
Tai35a	2422002	**0**	16.7	**0**	4.4	**0**	0.280	0.18	0.487	3.5
Tai40a	3139370	**0.074**	57.5	0.222	5.2	**0.220**	0.561	0.20	0.593	13.1
Tai50a	4938796	0.550	127.6	0.725	10.2	**0.410**	0.889	0.23	0.901	29.7
Tai60a	7205962	0.643	128.5	0.718	25.7	**0.450**	0.940	0.41	1.068	58.5
Tai80a	13499184	0.771	244.8	0.753	52.7	**0.360**	0.648	1.0	1.178	152.2
Tai100a	21052466	1.045	385.7	0.825	142.1	**0.300**	0.977	1.99	1.115	335.6
Average		**0.342**	109.5	**0.360**	26.88	**0.193**	0.548	0.45	0.812	84.9

Table 10: Comparison of the TLBO-based hybrid algorithms with state-of-the-art algorithms on Type-2 instances with parameter setting 3 (30 individuals, 100 generations, 1 processor, and 5,000 x n failures for RTS).

		TLBO-RTS		JRG-DivTS		ACO/GA/LS		GA/SD		LagSA		GA/C-TS	
Instance	BKS	APD	min.	APD	min.	APD	min.	APD	min.	APD	min.	APD	min.
Sko42	15812	**0**	7.7	**0**	4.0	**0**	0.7	0.014	0.16	0.42	3.71	**0**	1.2
Sko49	23386	**0.014**	4.1	**0.008**	9.6	**0.056**	7.6	0.107	0.28	0.14	6.96	**0.009**	2.1
Sko56	34458	**0.003**	18.5	**0.002**	13.2	0.012	9.1	0.054	0.42	0.12	13.6	**0.001**	3.2
Sko64	48498	**0.003**	27.5	**0**	22.0	0.004	17.4	0.051	0.73	0.12	24.15	**0**	5.9
Sko72	66256	0.022	39.2	**0.006**	38.0	**0.018**	70.8	0.112	0.93	0.26	43.1	**0.014**	8.4
Sko81	90998	**0.023**	56.6	**0.016**	56.6	0.025	112.3	0.087	1.44	0.11	66.2	**0.014**	13.3
Sko90	115534	**0.029**	79.4	**0.026**	89.6	0.042	92.1	0.139	2.31	0.16	116.9	**0.011**	22.4
Sko100a	152002	0.040	109.5	0.027	129.2	**0.021**	171.0	0.114	3.42	0.13	170.8	**0.018**	33.6
Sko100b	153890	0.027	109.4	**0.008**	106.6	**0.012**	192.4	0.096	3.47	-	-	**0.011**	34.1
Sko100c	147862	**0.016**	109.6	0.006	126.7	**0.005**	220.6	0.075	3.22	-	-	**0.003**	33.8
Sko100d	149576	**0.035**	109.3	**0.027**	123.5	**0.029**	209.2	0.137	3.45	-	-	0.049	33.9
Sko100e	149150	0.020	109.7	**0.009**	108.8	**0.002**	208.1	0.071	3.31	-	-	**0.002**	30.7
Sko100f	149036	**0.024**	109.5	**0.023**	110.3	0.034	210.9	0.148	3.55	-	-	**0.032**	35.7
Average		**0.019**	69.06	**0.012**	72.1	0.020	117.1	0.093	2.1	0.183	55.7	**0.013**	19.9

Table 11: Comparison of the TLBO-based hybrid algorithms with state-of-the-art algorithms on Type-3 instances with parameter setting 3 (30 individuals, 100 generations, 1 processor, and 5,000 x n failures for RTS).

		TLBO-RTS		JRG-DivTS	ACO/GA/LS	GA/SD	ITS	GA-C/TS	SC-TABU	MOB
Instance	BKS	APD	min.	APD	BPD	APD	APD	APD	APD	
Kra30a	88900	**0**	1.0	**0**	**0**	-	**0**	-	0.714	-
Kra30b	91420	**0**	1.0	**0**	**0**	0.253	**0**	**0**	0.178	-
Kra32	88700	**0**	1.2	**0**	**0**	0.037	-	**0**	-	-
Ste36a	9526	**0**	1.6	**0**	**0**	-	0.04	-	-	-
Ste36b	15852	**0**	1.6	**0**	**0**	0.246	**0**	0.005	-	-
Ste36c	8239110	**0**	1.6	**0**	**0**	0.015	**0**	**0**	-	-
Esc32b	168	**0**	1.2	**0**	**0**	**0**	**0**	0.039	-	-
Esc32c	642	**0**	1.2	**0**	**0**	**0**	**0**	**0**	-	-
Esc32d	200	**0**	1.2	**0**	**0**	**0**	**0**	**0**	-	-
Esc32e	2	**0**	1.2	**0**	**0**	**0**	**0**	**0**	-	**0**
Esc32g	6	**0**	1.2	**0**	**0**	**0**	**0**	-	-	**0**
Esc32h	438	**0**	1.2	**0**	**0**	**0**	**0**	-	-	**0**
Esc64a	116	**0**	9.2	**0**	**0**	**0**	**0**	**0**	-	**0**
Esc128	64	**0**	75.0	**0**	**0**	**0**	0.01	**0**	-	-
Average		**0**	7.1	**0**	**0**	0.045	**0**	0.004	0.446	**0**

Table 12: Comparison of the TLBO-based hybrid algorithms with state-of-the-art algorithms on Type-4 instances with parameter setting 3 (30 individuals, 100 generations, 1 processor, and 5,000 x n failures for RTS).

		TLBO-RTS		JRG-DivTS		ITS		ACO/GA/LS		SC-TABU	
Instance	BKS	APD	min.	APD	min.	APD	min.	APD	min.	APD	min.
Tai20b	122455319	**0**	0.3	**0**	0.2	**0**	0.01	-	-	**0**	0.002
Tai25b	344355646	**0**	0.3	**0**	0.5	**0**	0.01	-	-	0.007	0.010
Tai30b	637117113	**0**	1	**0**	1.3	**0**	0.01	**0**	0.3	**0**	0.034
Tai35b	283315445	**0**	1.5	**0**	2.4	0.02	0.08	**0**	0.3	0.059	-
Tai40b	637250948	**0**	2.3	**0**	3.2	0.01	0.2	**0**	0.6	**0**	0.092
Tai50b	458821517	**0**	4.4	**0**	8.8	0.02	0.5	**0**	2.9	**0.002**	0.23
Tai60b	608215054	**0**	7.6	**0**	17.1	0.04	1.7	**0**	2.8	**0**	0.41
Tai80b	818415043	**0.006**	18.8	0.006	58.2	0.23	3.0	**0**	60.3	**0.003**	1.0
Tai100b	1185996137	0.046	36.1	0.056	118.9	0.14	6.66	**0.01**	698.9	**0.014**	1.98
Average		**0.006**	8.05	0.07	23.4	0.051	1.35	**0.001**	109.4	**0.009**	0.47

Table 13: Comparison of the parallel TLBO-RTS algorithm results with other state-of-the-art parallel algorithms.

Instance	BKS	TLBO-RTS		TB-MTS	TB-COSEARCH	CPTS		QAP-IPGA		PMTS	
		APD	min.	APD	APD	APD	min.	APD	min.	APD	min.
Bur26d	3821225	**0**	9.2	**0**	**0**	**0**	0.4	**0**	14.7	-	-
Nug30	6124	**0**	14.8	**0**	**0**	**0**	1.7	**0**	28.6	-	-
Tai35b	283315445	**0**	22.4	**0**	**0**	**0**	2.4	0.820	33.3	-	-
Ste36c	8239110	**0**	24.1	**0**	**0**	**0**	2.5	**0**	34.7	-	-
Lipa50a	62093	**0**	68.8	**0**	**0**	**0**	11.2	0.840	35.5	-	-
Sko64	48498	**0**	119.3	0.004	0.003	**0**	42.9	0.350	44.1	**0**	0.7
Tai64c	1855928	**0**	117.7	**0**	**0**	**0**	20.6	**0**	44.2	-	-
Tai100a	21052466	**0.596**	483.3	0.814	**0.544**	**0.589**	261.2	0.893	52.1	-	-
Tai100b	1185996137	**0**	508.2	0.397	0.135	**0.001**	241.0	**0**	52.3	-	-
Sko100a	152002	**0.003**	594.3	0.073	0.054	**0**	304.8	0.290	52.4	**0**	16.9
Wil100	273038	**0**	482.6	0.035	0.009	**0**	316.6	**0**	52.7	-	-
Tai150b	498896643	**0.015**	428.5	1.128	0.439	**0.076**	1,549.4	0.790	57.3	-	-
Tho150	8133398	**0.030**	556.6	**0.012**	0.065	**0.013**	1,991.7	0.940	57.9	-	-
Average		**0.049**	263.8	0.162	**0.096**	**0.052**	365.1	0.377	43.1	**0**	8.8