A parallel tabu search for the unconstrained binary quadratic programming problem

Jialong Shi, Qingfu Zhang, Bilel Derbel, Arnaud Liefooghe

A Parallel Tabu Search for the Unconstrained Binary Quadratic Programming Problem

Jialong Shi, Qingfu Zhang
Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
jlshi2-c@my.cityu.edu.hk, qingfu.zhang@cityu.edu.hk

Bilel Derbel, Arnaud Liefooghe
Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL
Dolphin, Inria Lille – Nord Europe
F-59000 Lille, France
{bilel.derbel, arnaud.liefooghe}@univ-lille1.fr

***Abstract*—Although several sequential heuristics have been proposed for dealing with the Unconstrained Binary Quadratic Programming (UBQP), very little effort has been made for designing parallel algorithms for the UBQP. This paper propose a novel decentralized parallel search algorithm, called Parallel Elite Biased Tabu Search (PEBTS). It is based on D^{2}TS, a state-of-the-art sequential UBQP metaheuristic. The key strategies in the PEBTS algorithm include: (i) a lazy distributed cooperation procedure to maintain diversity among different search processes and (ii) finely tuned bit-flip operators which can help the search escape local optima efficiently. Our experiments on the Tianhe-2 supercomputer with up to 24 computing cores show the accuracy of the efficiency of PEBTS compared with a straightforward parallel algorithm running multiple independent and non-cooperating D^{2}TS processes.**

I. Introduction

The Unconstrained Binary Quadratic Programming (UBQP) problem is a well-known NP hard problem [1]. Given a collection of items such that each pair is associated with a profit value that can be positive, negative or zero, the UBQP problem seeks a subset of items that maximizes the sum of their paired values. The value of a pair is summed up only if the two corresponding items are selected. Besides its theoretical interest [2], UBQP is often a bottleneck on a wide variety of application fields [1], hence making it particularly appealing to study and solve. Randomized heuristics, including metaheuristics [3] like Tabu Search (TS) and Evolutionary Algorithm (EA) are widely used to handle the UBQP. Very little effort has been made on parallel UBQP algorithms. It is still a very challenging issue to design efficient UBQP metaheuristics which can make the best use of modern and massively parallel multi-processors computing platforms. In this paper, we propose and study a parallel TS algorithm, called Parallel Elite Biased Tabu Search (PEBTS), as a first step towards the design of effective and efficient parallel search heuristics for the UBQP problem. The proposed PEBTS algorithm is based on a state-of-the-art sequential UBQP metaheuristic, namely, the Diversification-driven Tabu Search (D^{2}TS) proposed by Glover at al. [4]. In the proposed PEBTS, a good trade-off between the diversification and intensification in the TS is maintained in a distributed manner. Different search processes start from different initial solutions and conduct their search concurrently. The parallel processes are mapped to the computing cores and organized in a torus topology. Each core provides its current best solution asynchronically to the four neighboring cores within the torus, and uses the best solution it found or received to guide its search process. This is achieved by leveraging existing bit-flip variation operators, the core of the TS search procedures, and by renovating their design components in order to enable an effective cooperative search among the different parallel processes. In fact, through the designed parallel cooperation strategy, we are able to show that the search efficiency can be improved and that the diversity of solutions maintained distributively over all the PEBTS processes can accurately be maintained.

The rest of this paper is organized as follows. In Section II, we provide some background and related work on the UBQP problem. For completeness, in Section III, we review the state-of-the-art D^{2}TS sequential procedure in a nutshell. In Section IV, we describe the proposed PEBTS algorithm and its general design components. In Section V, we report our experimental findings when deploying our algorithm using the Tianhe-2 supercomputer. In Section VI, we conclude the paper.

II. Unconstrained Binary Quadratic Programming

The UBQP problem can be formalized as follows [5].

$$\begin{aligned} \text{maximize} \quad & f(x) = x'Qx = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j \\ \text{subject to} \quad & x \in \{0,1\}^n \end{aligned}$$

where $Q = [q_{ij}]$ is a $n \times n$ matrix, and x is a vector of n binary (0-1) variables. In recent years, the UBQP problem has received a growing interest, and appears in many areas such as financial analysis [6], social psychology [7], machine scheduling [8], computer aided design [9] and cellular radio channel allocation [10]. This problem is also related to a wide range of combinatorial optimization problems from graph theory, such as maximum cliques, maximum cuts, maximum vertex packing and minimum coverings [11], [12]. A recent survey of the UBQP can be found in [1]. Some related research developments are reviewed in the following.

Katayama and Narihisa [13] proposed the Parthenogenetic Algorithm (PA). PA can be seen as an Iterated Local Search (ILS), where the perturbation is performed by means

of a mutation operator. Merz and Katayama [14] addressed the fitness landscape of UBQP instances. They also proposed a Memetic Algorithm (MA) in which an EA procedure is combined with a *k-opt* local search. Palubeckis [15] proposed the Multi-start Tabu Search (MST), in which the initial solution is constructed independently at each iteration, and improved by means of TS. Later in [16], an Iterated Tabu Search (ITS) algorithm, is proposed where the initial solution from which TS starts the search is generated by perturbing the resulting solution found by TS at the previous round. Glover et al. [4] proposed the Diversification-Driven Tabu Search (D^2TS), where a population of elite solutions is managed using an internal memory. At each iteration of D^2TS, a solution selected at random within the elite population, it is then perturbed based on a memory-based strategy, and improved by means of TS. The resulting solution is then used to update the elite population.

The TS is actually the core component of a number of other metaheuristics. For instance, Lü et al. [17] designed a Hybrid Metaheuristic Approach (HMA), in which TS is incorporated into an EA. In HMA, offspring solutions are generated by means of a path-relinking procedure, and improved by means of TS. The population updating mechanism is based on a distance-and-quality goodness score function. Wang et al. [18] presented the Backbone Guided Tabu Search (BGTS). Backbone variables are the ones having the same value in all optimal solutions. BGTS divides the variables into a backbone set and a free set, both of which change members at each iteration. Fixing or freeing a variable is determined by the result of TS. The initial solution of TS is generated by fixing the backbone variables and randomly assigning values to free variables. A Backbone Multilevel Memetic Algorithm (BMMA) is also designed in [19], which includes a backbone-based coarsening phase, an uncoarsening phase, and a memetic refinement phase. The coarsening phase creates coarser sub-problems. Then, the memetic refinement and the uncoarsening phases try to solve the sub-problems from simple to complex. In another work from Wang et al. [20], a path-relinking based algorithm is proposed. In this algorithm, an elite population is iteratively updated by a path-relinking phase and a TS phase. Recently, Wang et al. [21] introduced a GRASP-Tabu Search (GRASP-TS) algorithm and an enhanced variant based on a population management strategy.

All of the aforementioned metaheuristics are intrinsically *sequential*. They cannot fully benefit from the computational resources available in multi-processor computers. In [22], an island model approach using a basic master-slave centralized migration strategy is proposed. However, it does not incorporate TS, which has been shown to be a state-of-the-art component when tackling the UBQP problem, as mentioned before. To our best knowledge, there does not exist any other parallel algorithms for the UBQP problem. In this paper, we intend to fill this gap by proposing a parallel TS-based algorithm. This parallel algorithm takes inspiration from the state-of-the-art sequential D^2TS, as detailed in the next section for the sake of completeness.

III. Diversification-Driven Tabu Search

Proposed by Glover et al. [4], D^2TS maintains a population of at most R elite solutions, denoted $EliteSol$, in the algorithm internal memory. A vector of size n (the problem size), denoted $EliteFreq$, is used to record the total number of times each variable is assigned a value of 1 in $EliteSol$. Besides, D^2TS also maintains another vector of size n, denoted $FlipFreq$, in order to record the number of times each variable has been flipped. At each iteration, D^2TS first selects *at random* a solution from $EliteSol$, for which a memory-based perturbation operator is applied. The perturbed solution is then improved by a simple TS procedure. After the TS improvement phase, the resulting elite solution is used to update $EliteSol$, if it is not already included there-in. If $EliteSol$ has reached its full capacity (R), and if the new solution is better than the worst one in $EliteSol$, then the worst one is replaced by the new one. D^2TS starts from a randomly generated solution and stops once the computational budget is exhausted. The main ingredient of D^2TS are depicted in the pseudo-code of Algorithm 1 and discussed briefly in next paragraphs.

In D^2TS, the TS procedure is based on 1-Flip moves. At each move, TS checks the flip of variables that are not contained in the tabu list, and selects the move that leads to the best neighboring solution. The flipped variable in the selected neighboring solution is used to update the tabu list, and is forbidden to be flipped until a number of $TabuTenure(i)$ moves have elapsed. Following [4], $TabuTenure(i)$ is set as:

$$TabuTenure(i) = c + rand(10), \tag{1}$$

where c is a predefined constant and $rand(10)$ is a random integer in $[1, 10]$. Notice that a fast move evaluation procedure is proposed in [23] in order to calculate the move gain of a given neighbor, and hence its fitness value. In addition, a simple aspiration criterion is applied: if a move leads to a solution better than the current best solution $x^\star$, this move will be permitted in spite of being tabu. The TS terminates when the current best solution $x^\star$ is unchanged for α moves. Here α is called *improvement cutoff* and is a predefined constant.

The perturbation step is based on $EliteFreq$ and $FlipFreq$. It first requires to compute the *score* of each variable i:

$$Score(i) = \frac{EliteFreq(i)(r - EliteFreq(i))}{r^2} + \beta(1 - \frac{FlipFreq(i)}{max\_Freq}), \tag{2}$$

where β is a predefined constant and max_Freq is the largest value in $FlipFreq$. After the scoring step, the perturbation operator sorts all the variables in the non-increasing order of their scores, and selects γ different variables to be flipped. At each round of selection, the j-th highly-scored variable has a probability $P_j = \frac{j^{-\lambda}}{\sum_{i=1}^{n} i^{-\lambda}}$ to be flipped, where γ and λ are predefined constants.

Based on this sequential D^2TS algorithm and its components, we are able to introduce our proposed parallel metaheuristic for UBQP, which is the purpose of the next section.

Algorithm 1: Diversification-Driven Tabu Search (D^2TS)

Input: $Q, \alpha, c, R, \beta, \lambda, \gamma$
Output: the best binary n-vector $x^\star$ found so far

$EliteSol \leftarrow \{\}; r \leftarrow 0; EliteFreq(i) \leftarrow 0, i = 1, \ldots, n$;
$x \leftarrow$ randomly generate an initial solution;
while $r < R$ **do**
  $x' \leftarrow$ TabuSearch(x, Q, α, c);
  if x' *is not in* $EliteSol$ **then**
    $EliteSol \leftarrow EliteSol + \{x'\}$;
    $r \leftarrow r + 1$;
    $EliteFreq = EliteFreq + x'$;
  $x \leftarrow$ randomly select a solution from $EliteSol$;
  $x \leftarrow$ Perturbation$(x, EliteSol, EliteFreq, \beta, \lambda, \gamma)$;
while *Stopping criterion is not met* **do**
  $x \leftarrow$ randomly select a solution from $EliteSol$;
  $x \leftarrow$ Perturbation$(x, EliteFreq, FlipFreq, \beta, \lambda, \gamma)$;
  $x' \leftarrow$ TabuSearch(x, Q, α, c);
  $x_w \leftarrow$ the worst solution in $EliteSol$;
  if x' *is not in* $EliteSol$ *and* $f(x') > f(x_w)$ **then**
    $EliteSol = EliteSol + \{x'\} - \{x_w\}$;
    $EliteFreq = EliteFreq + x' - x_w$;

IV. Parallel Elite Biased Tabu Search

Due to the stochastic nature of metaheuristics, the time it takes to reach a target function value is a random variable. Hence, running multiple algorithm processes in parallel can reduce the hitting time. In addition, many experimental studies from the literature prove that the cooperation among different processes can achieve even better efficiency. Therefore, designing a parallel cooperative metaheuristic for the UBQP can be a good alternative to leverage existing sequential start-of-the-art algorithms. Before going into the details of our parallel algorithm, we shall first briefly review existing work on parallel metaheuristics to better position our contribution.

A. Overview of Parallel Metaheuristics

Some recent surveys of parallel metaheuristics can be found in [24]–[26]. Generally speaking, the class of metaheuristic approaches can be divided into two categories: (i) population-based metaheuristics such as EA, and (ii) trajectory-based metaheuristics such as TS. The parallelization of population-based metaheuristics can either aim at: (1) parallelizing the variation operators applied to the individuals, or (2) parallelizing the evolution of different individual in the population. For trajectory-based metaheuristics, three mainstream strategies can be reported.

- *Parallel single solution evaluation scheme*. This scheme requires that the flow of the evaluation function can be run in a parallel way and aims at speeding up costly fitness function evaluations. A master-slave approach is generally applied, where the master executes the main procedure of the algorithm, whereas the slaves are in charge of running the parallel tasks when one new single solution is to be evaluated in parallel.
- *Parallel multiple solution evaluation scheme*. In this scheme, multiple candidate solution are evaluated in parallel, typically when exploring the neighborhood of the current solution evolved by the metaheuristic.
- *Parallel multistart scheme*. In this scheme, different processes start from different initial solutions and produce different search trajectories. They may have different configurations. During the search process, these processes can communicate with each other in order to exchange useful information. The parallel TS algorithm proposed in this paper belongs to this third category.

It is important to notice that parallel TS algorithms exist for other problem classes. In fact, De Falco et al. [27] proposed a simple parallel variant of TS based on exchanging best solutions. Al-Yamani et al. [28] developed a heterogeneous parallel TS algorithm for the VLSI placement problem, which integrates the first and third schemes sketched previously. Bortfeldt et al. [29] designed a distributed parallel TS metaheuristic for the container loading problem, in which each process periodically adopts solution from its predecessor and restarts from the adopted solution. Attanasio et al. [30] also proposed a distributed parallel TS metaheuristic for the dynamic multi-vehicle dial-a-ride problem, in which two different cooperation strategies are investigated. Banos et al. [31] presented a parallel metaheuristic for the graph partitioning problem which hybridizes Simulated Annealing (SA) and TS. Blazewicz et al. [32] proposed a master-slave parallel TS algorithm for the two-dimensional cutting problem. Le Bouthillier and Crainic [33] proposed a cooperative parallel metaheuristic for the vehicle routing problem with time windows, in which TS processes and EA processes are executed in parallel. Talbi and Bachelet [34] proposed a parallel metaheuristic for the quadratic assignment problem, which uses TS as the main search agent. Maischberger [35] proposed a synchronous distributed parallel metaheuristic for the vehicle routing problems, in which each process executes ILS extended with TS.

It is worth noticing that the parallel algorithm proposed in this paper is different from the aforementioned parallel algorithms, essentially because it uses a novel cooperation strategy and, more importantly, a finely-tuned UBQP-dedicated bit-flip perturbation operator.

B. Proposed Approach

The proposed parallel TS algorithm for UBQP is called Parallel Elite Biased Tabu Search (PEBTS). In PEBTS, m processes following the same search strategy are run in parallel, starting from different initial solutions. The basic procedure of each process repeatedly alternates between a TS phase and a perturbation phase until the stopping condition is satisfied. Each process has four neighbors and communicates with its neighbors aperiodically. The messages transmitted among processes are the newest best-found solutions. In the meantime, each process keeps one elite solution x_e in its

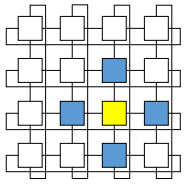

Fig. 1. A 4×4 torus topology of the PEBTS processes

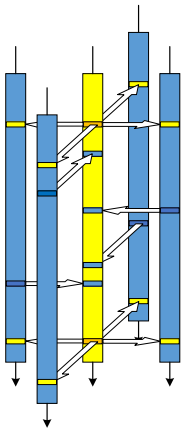

Fig. 2. A sketch of the communication among the PEBTS processes

memory, corresponding to the best one of the solutions it received and found.

Starting from a randomly-generated solution, each PEBTS process performs a TS procedure. Similar to D^{2}TS, when the best solution $x^\star$ remains unchanged for α moves, the TS procedure terminates. However, after this TS phase, the PEBTS process directly perturbs the current best solution $x^\star$ and uses the perturbed solution as the initial solution of the next round of TS. This strategy is different from the perturbation strategy in D^{2}TS, which randomly selects a solution from the $EliteSol$ set to be perturbed, i.e., there is no elite set maintained explicitly in PEBTS.

The parallel processes of PEBTS are organized following a torus topology. For instance, in Fig. 1, the four neighbors of the yellow process are colored in blue. During the search, each process only communicates with its neighbors. Specifically, when a process finds a new best solution $x^\star$, it sends this solution to all of its neighbors. Meanwhile, each process periodically checks whether there are new solutions coming from its neighbors, and receive those solutions if they exist. Notice that, although a process may receive better solutions from its neighbors, it still sends the one it found itself to its neighbors. Fig. 2 sketches the communication between one PEBTS process and its neighbors. Let S_r denote the set of solutions that a process receives from its neighbors. In PEBTS, each process maintains one elite solution x_e in its memory, which is the best one from $S_r \cup \{x^\star\}$. The elite solution x_e guides both the TS phase and the perturbation phase.

During the TS phase, when a new variable is added to the tabu list, if the same value is taken by this variable within the current solution and x_e, its $TabuTenure$ will be added an extra constant c'. Accordingly, we set:

$$TabuTenure(i) = \begin{cases} c + c' + rand(10) & \text{if } x_i = x_{e,i} \\ c + rand(10) & \text{otherwise,} \end{cases} \quad (3)$$

where c' is a predefined constant, $x_{e,i}$ is the value of variable i in x_e. By adopting this rule at every parallel process, when a variable is flipped to the same value it takes in x_e, it will

Algorithm 2: Parallel Elite Biased Tabu Search (PEBTS)

Input: $Q, \alpha, c, c', \beta, \lambda, \gamma$
Output: the best binary n-vector $x^\star$ found so far

1 $x \leftarrow$ randomly generate an initial solution;
2 $k \leftarrow 0$; $x^\star \leftarrow x$; $TabuTenure(i) \leftarrow 0, i = 1, \ldots, n$;
3 **while** *Stopping criterion is not met* **do**
4 **if** $x^\star$ *has updated* **then** SendToNeighbors($x^\star$) ;
5 $S_r \leftarrow$TryToReceive();
6 $x_e \leftarrow$SelectBestSolution($S_r \cup \{x^\star\}$);
7 **for** *each* i *that* $TabuTenure(i) > 0$ **do**
8 $TabuTenure(i) \leftarrow TabuTenure(i) - 1$
9 $\Delta f' \leftarrow -\infty$;
10 $\Delta f_i \leftarrow$ the move value of flipping $x_i, i = 1, \ldots, n$;
11 **for** *each* i *such that* $TabuTenure(i) = 0$ *or* $f(x) + \Delta f_i > f(x^\star)$ **do**
12 **if** $\Delta f' < \Delta f_i$ **then** $\Delta f' \leftarrow \Delta f_i$; $i' \leftarrow i$
13 Flip the variable i' in x;
14 $FlipFreq(i') \leftarrow FlipFreq(i') + 1$;
15 **if** $x_{i'} = x_{e,i'}$ **then**
16 $TabuTenure(i') \leftarrow c + c' + rand(10)$;
17 **else** $TabuTenure(i') \leftarrow c + rand(10)$;
18 **if** $f(x) > f(x^\star)$ **then**
19 $x^\star \leftarrow x$; $k \leftarrow 0$
20 **else** $k \leftarrow k + 1$;
21 **if** $k > \alpha$ **then**
22 $x \leftarrow x^\star$;
23 $x \leftarrow$EliteBiasedPerturbation($x, x_e, FlipFreq, \beta, \lambda, \gamma$);
24 $k \leftarrow 0$; $TabuTenure(i) \leftarrow 0; i = 1, \ldots, n$;

be forbidden to flip back for a longer tenure. This will tend to let the current solution of a parallel process to be more similar to its elite solution x_e, and hence to encourage the search direction to be attracted by x_e.

The perturbation operator is also influenced by x_e. In PEBTS, the score of variable i computed at every parallel process is defined as follows:

$$Score(i) = |x_i - x_{e,i}| + \beta\Big(1 - \tfrac{FlipFreq(i)}{max\_Freq}\Big). \quad (4)$$

A number of variables are then selected and flipped according to this modified score function, following the same probability distribution as defined previously in sequential D^{2}TS. From the left term of Eq. (4), we can see that the variables that take different values in the current solution and in x_e tend to have a higher score. As a consequence, they are more likely to be selected and flipped, and the perturbed solution will tend to get more similar to x_e.

The entire procedure of a PEBTS process is shown in Algorithm 2. Lines 4-6 correspond the communication phase, Lines 7-20 to the TS phase, and Lines 21-24 to the perturbation phase.

C. Design Principles of PEBTS and Discussion

Compared to D^{2}TS, PEBTS has the following design features:

- Each PEBTS process restarts from the best solution $x^\star$ that it has found locally (D^{2}TS restarts with one solution from global $EliteSol$).
- Each PEBTS process sends its own best solution $x^\star$ to its neighboring processes whenever there is a change, and possibly receives new solutions from its neighbors.
- In PEBTS, both the TS phase and the perturbation phase are guided by the elite solution x_e, which is the best one from received solutions and $x^\star$.

In order to maintain diversity, D^{2}TS keeps R solutions in $EliteSol$, and restarts from one of them at each iteration. However, in PEBTS, there are m parallel processes which start from different initial solutions and search in different areas of the search space. Notice that the elite solution x_e maintained locally by each parallel process is *not* used as a restarting point just before the perturbation phase. Instead, the local best solution $x^\star$ is considered for perturbation. As such, we prevent x_e to flood the distributed search and we expect the diversification of PEBTS to be intrinsically maintained. Notice also that this allows us to get rid of explicitly maintaining any $EliteSol$ set in PEBTS, which is likely to be implicitly achieved due to its distributed nature. This is the reason why we adopted the first design feature in PEBTS.

The goal of the second feature is to let different processes share information with each other. In fact, we believe that the most useful information is hidden in the best solutions found distributively by the PEBTS processes. On the other hand, since communication with a restricted number of other processes (namely, four) only happens when a new best solution is found, the distributed protocol underlying PEBTS tends to reduce the overall network communication load, and in the meantime to enables an accurate level of solution diversity.

The third design feature reflects how the PEBTS processes benefit from the information hidden in the received elite solutions. According to the fitness-distance correlation analysis of Merz and Katayama [14], high-quality UBQP solutions are typically contained in a small fraction of the search space, and there exists a high correlation between the fitness of high-quality solutions and the (Hamming) distance to the optimum. The higher the quality of a solution, the more likely it is close to the optimum. Considering these characteristics of UBQP, each PEBTS process is designed to use solely one elite solution x_e, which is the best one from the set $S_r \cup \{x^\star\}$. After getting a new x_e, a PEBTS parallel process does not update its current solution to x_e, because we do not want the process to forget the current region of the search space where it is actually operating. Instead, we want to encourage the different parallel PEBTS processes to search in diverse regions of the search space. Therefore, when a process is given a new x_e, it does not directly jump to x_e, but instead it is simply *attracted* by x_e. Here the "attraction" of x_e is completed by adaptively pushing the resulting solutions of the TS and perturbation phases to be more similar to x_e, as described in Section IV-B.

V. Experimental Analysis

In order to investigate the accuracy and the relative performance of the proposed parallel algorithm, we consider the following two sets of instances. The first set contains the 10 UBQP instances of size $n = 2\,500$ in the ORLIB [36] repository. These instances are named $\{\texttt{b2500.1}, \ldots, \texttt{b2500.10}\}$, and all have a density of 0.1, where the density refers to the proportion of non-zero numbers in the matrix Q. The second set of instances contains the 21 large instances named $\{\texttt{p3000.1}, \ldots, \texttt{p7000.3}\}$, with a problem size ranging from $n = 3\,000$ to $7\,000$, and with densities ranging from 0.5 to 1.0. Both instance sets have known optima and are widely used in the literature; see e.g. [4], [14]–[21]. They are commonly believed to be very challenging. In our experiments, algorithms are implemented in GNU C++ with the `-O2` optimization option. The fast move evaluation method from [23] is used in our implementation. The communication between different processes is achieved using MPI. The computer platform is the Tianhe-2 supercomputer. Notice that Tianhe-2 is one of the world's top-ranked supercomputers. It is equipped with 17 920 computer nodes, each comprising two Intel Xeon E5-2692 12C (2.200 GHz) processors. The operating system of Tianhe-2 is based on Ubuntu.

A. Impact of Cooperation

One notable feature of the proposed PEBTS algorithm is the cooperation mechanism designed to coordinate the actions of the parallel search processes. In order to evaluate the benefit of the underlying cooperative parallel search, we consider the aforementioned first set of instances and we conduct the following set of experiments, as detailed in the next paragraphs.

For each instance, PEBTS is compared to a simple parallel algorithm, called PITS, in which several parallel *independent* TS processes are executed without any cooperation mechanism. More precisely, the search procedure of each PITS process is set to be similar to that of the PEBTS process, except that PITS processes do *not* communicate. Hence, at each PITS process, the elite solution x_e is equal to the current best solution $x^\star$, instead of the best one from $S_r \cup \{x^\star\}$. Moreover, based on some preliminary pilot testing, we were able to observe that the best solution $x^\star$ found by a parallel process has a relatively low quality at the early stages of the search process. Besides, $x^\star$ changes very quickly at the

TABLE I
Parameter settings of the experiment in Section V-A.

Parameters	Description	Values
m	Process number	16
c'	Influence strength of x_e in TS	{0, 6, 12, 18, 24}
c	Tabu tenure constant	$n/100$
α	Improvement cutoff of TS	$20n$
β	Frequency-related weight in perturbation scoring	1
λ	Perturbation selection importance factor	1.2
γ	Perturbation strength	$n/4$
T_{max}	Max runtime	60s
CST	Communication start time	After n TS moves

TABLE II

PEBTS *vs.* PITS ON UBQP INSTANCES OF SIZE $n = 2\,500$. g_{avr} IS THE AVERAGE GAP TO THE OPTIMAL OBJECTIVE VALUE, suc IS THE SUCCESS RATE OVER THE 100 RUNS, t_{avr} IS THE AVERAGE RUNTIME AND t_{mdn} IS THE MEDIAN RUNTIME (IN SECONDS). FOR EACH ALGORITHM THE LAST ROW ("AVERAGE") SUMMARIZES THE ALGORITHM'S AVERAGE PERFORMANCE OVER ALL THE CONSIDERED INSTANCES.

Instances	PEBTS (c'=0)				PEBTS (c'=6)				PEBTS (c'=12)				PEBTS (c'=18)				PEBTS (c'=24)			
	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}
b2500.1	0.0	100	0.19	0.16	0.0	100	0.17	0.16	0.0	100	0.16	0.15	0.0	100	0.15	0.14	0.0	100	0.19	0.15
b2500.2	0.0	100	1.38	1.16	0.0	100	1.15	0.82	7.2	98	3.92	0.90	78.3	70	19.77	0.78	131.6	52	30.11	16.80
b2500.3	0.0	100	1.06	0.29	0.0	100	0.28	0.21	5.3	98	1.43	0.22	5.3	98	1.39	0.17	29.7	89	6.77	0.17
b2500.4	0.0	100	0.11	0.10	0.0	100	0.11	0.10	0.0	100	0.10	0.10	0.0	100	0.09	0.09	0.0	100	0.09	0.09
b2500.5	0.0	100	0.16	0.15	0.0	100	0.16	0.15	0.0	100	0.15	0.13	0.0	100	0.17	0.12	0.0	100	0.26	0.11
b2500.6	0.0	100	0.26	0.23	0.0	100	0.23	0.21	0.0	100	0.18	0.17	0.0	100	0.16	0.15	0.0	100	0.15	0.14
b2500.7	0.0	100	0.55	0.49	0.0	100	0.47	0.41	0.0	100	0.33	0.29	0.0	100	0.30	0.28	7.3	99	0.88	0.25
b2500.8	0.0	100	0.18	0.14	0.0	100	0.17	0.13	0.1	99	0.75	0.13	1.0	93	4.34	0.13	1.5	89	6.73	0.13
b2500.9	0.0	100	0.63	0.44	0.0	100	1.02	0.33	1.7	88	9.94	0.23	3.2	77	14.79	0.19	4.1	71	18.00	0.18
b2500.10	0.0	100	0.50	0.33	0.0	100	0.46	0.36	0.0	100	0.46	0.27	13.4	97	3.84	0.26	40.4	89	8.25	0.28
Average	0.0	100	0.50	0.35	0.0	100	0.42	0.29	1.4	98.3	1.74	0.26	10.1	93.5	4.50	0.23	21.5	88.9	7.14	1.83
Instances	PITS (c'=0)				PITS (c'=6)				PITS (c'=12)				PITS (c'=18)				PITS (c'=24)			
	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}
b2500.1	0.0	100	0.17	0.16	0.0	100	0.19	0.17	0.0	100	0.18	0.17	0.0	100	0.20	0.17	0.0	100	0.24	0.19
b2500.2	0.0	100	1.77	1.17	0.0	100	2.13	1.03	6.1	97	6.77	2.28	24.3	88	12.72	4.42	40.2	83	20.29	8.90
b2500.3	0.0	100	0.80	0.31	0.0	100	0.42	0.28	0.0	100	0.41	0.29	0.0	100	0.60	0.26	2.7	99	1.16	0.34
b2500.4	0.0	100	0.11	0.11	0.0	100	0.11	0.10	0.0	100	0.11	0.10	0.0	100	0.11	0.10	0.0	100	0.12	0.11
b2500.5	0.0	100	0.15	0.14	0.0	100	0.16	0.14	0.0	100	0.16	0.14	0.0	100	0.18	0.15	0.0	100	0.19	0.16
b2500.6	0.0	100	0.26	0.23	0.0	100	0.26	0.25	0.0	100	0.25	0.22	0.0	100	0.28	0.24	0.0	100	0.31	0.29
b2500.7	0.0	100	0.57	0.52	0.0	100	0.55	0.46	0.0	100	0.64	0.59	0.0	100	0.80	0.60	0.0	100	1.41	0.82
b2500.8	0.0	100	0.23	0.14	0.0	100	0.19	0.17	0.0	100	0.17	0.14	0.0	100	0.21	0.17	0.0	100	0.26	0.17
b2500.9	0.0	100	0.77	0.26	0.0	100	0.65	0.35	0.0	100	0.74	0.30	0.7	95	3.68	0.28	0.4	97	2.69	0.29
b2500.10	0.0	100	0.56	0.41	0.0	100	0.66	0.38	0.0	100	1.49	0.54	3.4	99	2.86	0.56	6.9	98	6.44	0.94
Average	0.0	100	0.54	0.34	0.0	100	0.53	0.33	0.6	99.7	1.09	0.48	2.8	98.2	2.16	0.69	5.0	97.7	3.31	1.22

early stages, which may cause a heavy communication load in PEBTS. As a consequence, we configured each PEBTS process so that it only starts to communicate (by sending and receiving $x^\star$) with its neighbors after a predefined number of TS moves, which is set empirically to n (the problem size).

The parameter setting is summarized in Table I. We set the number of processes to $m = 16$. Each parallel process starts from a randomly and independently-generated solution. The termination criteria is set to a maximum runtime, namely 60 seconds, and each considered algorithm configuration is executed for 100 independent runs on each benchmark instance. Note that the runtime measured here is a wall-clock time. Since the TS phase constitutes the core stage of PEBTS and PITS, we also study the impact of the c' parameter, which is used to control the attraction strength of x_e during the TS phase, i.e., $c' \in \{0, 6, 12, 18, 24\}$ for both PEBTS and PITS. Notice that when $c' = 0$, x_e does not have any impact on TS, it only influences the perturbation phase. All other parameter values are the ones recommended in [4], except for β. In fact, considering that the left term of the original scoring formula (Eq. 2) ranges from 0 to $1/4$, and that the left term of the new scoring formula (Eq. 4) ranges from 0 to 1, we set $\beta = 1$ in PEBTS.

The results of this first set experiments are reported in Table II, where three performance measures are shown: (i) the average percentage gap to the optimum, (ii) the success rate, which is the percentage of runs where the optimum was found, and (iii) the (average and median) time to hit the optimum. Interestingly, we observe that higher values of c' have a negative effect on both competing algorithms, and more importantly, that PEBTS is no longer able to beat PITS. This clearly indicates that cooperation among the parallel processes when solving UBQP instances can only be beneficial when carefully tuned in order to maintain a good diversity balance.

TABLE III

PEBTS *vs.* PITS ($c' = 6$) ON UBQP INSTANCES OF SIZE $n = 2\,500$.

Instance	Median Runtime (s)		P-value of
	PEBTS	PITS	Mann-Whitney U-test
b2500.1	**0.16**	0.17	0.2774
b2500.2	**0.82**(+)	1.03	0.0337
b2500.3	**0.21**(+)	0.28	0.0214
b2500.4	0.10	0.10	0.8496
b2500.5	0.15	**0.14**	0.3584
b2500.6	**0.21**(+)	0.25	0.0267
b2500.7	**0.41**	0.46	0.2288
b2500.8	**0.13**(+)	0.17	0.0450
b2500.9	**0.33**	0.35	0.9018
b2500.10	**0.36**	0.38	0.6521

Actually, both competing algorithms are able to perform at their best for $c' = 6$, and PEBTS is able to significantly outperform PITS. In fact, the results of a Mann-Whitney U-test with respect to the running time of PEBTS and PITS for $c' = 6$ are shown in Table III, where bold font means better median runtime value and a "(+)" means that the difference is statistically significant with a 5% level. In summary, PEBTS performs significantly better than PITS on 4 instances, whereas it is not worse than PITS on the remaining ones. These first results prove that the cooperation among different *finely-tuned* PEBTS processes globally improves the overall efficiency.

B. Comparison with Multi-Start Parallel Variant of D^{2}TS

The other notable feature of the PEBTS algorithm is the fact that no explicit elite set is maintained within each parallel search process. In order to evaluate the relevance of such a design choice, we conduct a second set of experiments where the performance of PEBTS is compared against a multi-start parallel algorithm, called Parallel Independent D^{2}TS (PID^{2}TS), where multiple independent D^{2}TS are executed in parallel on the available processes. Hence PID^{2}TS is a parallel variant of D^{2}TS which is faster than the sequential D^{2}TS.We

TABLE IV
PARAMETER SETTINGS OF THE EXPERIMENT IN SECTION V-B.

Parameters	PEBTS	PID^{2}TS
m	24	24
R	-	8
c'	$n/400$	-
c	$n/100$	$n/100$
α	$20n$	$20n$
β	1	0.3
λ	1.2	1.2
γ	$n/4$	$n/4$
T_{max}	{60s,120s,360s,720s}	{60s,120s,360s,720s}
CST	After n TS moves	-

consider the second set of UBQP instances and the parameter setting depicted in Table IV. Notice that the maximum runtime is set to 60, 120, 360, 720 and 720 seconds, respectively, for instances with 3 000, 4 000, 5 000, 6 000 and 7 000 variables. For both algorithms, we perform 20 runs using $m = 24$ parallel processes. The parameter values of D^{2}TS are set as in the original paper [4]. The parameter values for PEBTS are seemingly the same, expect for the value of c', which was shown to have a high impact in the previous section. In fact, by performing a linear extrapolation from our first set of experiments, we empirically choose c' as a function of the problem size n, i.e., $c' = n/400$. Finally, we also included the baseline PITS algorithm discussed in the previous section for completeness.

Our experimental findings are summarized in Table V. Overall, PEBTS achieves the highest average success rate suc and the shortest average runtime t_{avr}, while PID^{2}TS achieves the lowest average gap to the optimum g_{avr}. Specifically, on the instances with 3 000 and 4 000 variables, PEBTS typically reaches the optimal function value faster than PITS and PID^{2}TS. On the instances with 5 000 and 6 000 variables, PID^{2}TS typically gets better solutions, but PEBTS is much faster than PID^{2}TS and PITS. On the instances with 7 000 variables, the success rate of PEBTS is the highest one, and approximately twice larger than the one from PID^{2}TS, while still achieving a better runtime.

Notice that the performance of PEBTS and PID^{2}TS is compared using 3 metrics: g_{avr}, suc and t_{avr}. As a consequence, we can use the Pareto-dominance relation to better judge which one is better. Indeed, if algorithm A is not worse than algorithm B on all 3 metrics, and A is better than B on at least one metric, then we state that A outperforms B. In Table V, PEBTS outperforms PID^{2}TS on 10 instances while PID^{2}TS outperforms PEBTS on only 2 instances. Based on the above analysis, we state that the proposed parallel cooperative approach performs better than PID^{2}TS in most cases.

VI. CONCLUSION

In this paper, we proposed a parallel metaheuristic called PEBTS, which to our best knowledge is the first cooperative parallel tabu search designed for the UBQP problem. We conducted two sets of experiments in order to fairly evaluate the behavior of the proposed algorithm when compared against independent parallel TS processes operating independently. Our findings reveal that cooperation can indeed speedup the overall search procedure, while increasing the probability to hit the optimum. It is our hope that these results can lead to the developments of further highly effective parallel algorithms dedicated to the challenging UBQP problem. One particularly interesting alternative is to consider a distributed elite memory that is maintained explicitly and cooperatively by the parallel processes involved in the search. However, this implies to design advanced adaptive operators and control rules that can operate distributively in order to maintain a good balance of solution diversity among the parallel processes.

Acknowledgments. The authors gratefully acknowledge Jin-Kao Hao for fruitful discussions about UBQP. The work described in this paper was supported by a grant from ANR/RCC Joint Research Scheme sponsored by the Research Grants Council of the Hong Kong Special Administrative Region, China and France National Research Agency (Project No. A-CityU101/16).

REFERENCES

[1] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," *Journal of Combinatorial Optimization*, vol. 28, no. 1, pp. 58–81, 2014.

[2] M. R. Garey and D. S. Johnson, "A guide to the theory of np-completeness," *WH Freemann, New York*, 1979.

[3] F. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2003.

[4] F. Glover, Z. Lü, and J.-K. Hao, "Diversification-driven tabu search for unconstrained binary quadratic problems," *4OR*, vol. 8, no. 3, pp. 239–253, 2010.

[5] A. Liefooghe, S. Verel, and J.-K. Hao, "A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming," *Applied Soft Computing*, vol. 16, pp. 10–19, 2014.

[6] R. McBride and J. Yormark, "An implicit enumeration algorithm for quadratic integer programming," *Management Science*, vol. 26, no. 3, pp. 282–296, 1980.

[7] F. Harary *et al.*, "On the notion of balance of a signed graph." *The Michigan Mathematical Journal*, vol. 2, no. 2, pp. 143–146, 1953.

[8] B. Alidaee, G. A. Kochenberger, and A. Ahmadian, "0-1 quadratic programming approach for optimum solutions of two scheduling problems," *International Journal of Systems Science*, vol. 25, no. 2, pp. 401–408, 1994.

[9] J. Krarup and P. M. Pruzan, "Computer-aided layout design," in *Mathematical programming in use*. Springer, 1978, pp. 75–94.

[10] P. Chardaire and A. Sutter, "A decomposition method for quadratic zero-one programming," *Management Science*, vol. 41, no. 4, pp. 704–712, 1995.

[11] P. M. Pardalos and G. P. Rodgers, "Computational aspects of a branch and bound algorithm for quadratic zero-one programming," *Computing*, vol. 45, no. 2, pp. 131–144, 1990.

[12] P. M. Pardalos and J. Xue, "The maximum clique problem," *Journal of global Optimization*, vol. 4, no. 3, pp. 301–328, 1994.

[13] K. Katayama and H. Narihisa, "On fundamental design of parthenogenetic algorithm for the binary quadratic programming problem," in *Congress on Evolutionary Computation, CEC*. IEEE, 2001, pp. 356–363.

[14] P. Merz and K. Katayama, "Memetic algorithms for the unconstrained binary quadratic programming problem," *BioSystems*, vol. 78, no. 1, pp. 99–118, 2004.

[15] G. Palubeckis, "Multistart tabu search strategies for the unconstrained binary quadratic optimization problem," *Annals of Operations Research*, vol. 131, no. 1-4, pp. 259–282, 2004.

[16] ——, "Iterated tabu search for the unconstrained binary quadratic optimization problem," *Informatica*, vol. 17, no. 2, pp. 279–296, 2006.

[17] Z. Lü, F. Glover, and J.-K. Hao, "A hybrid metaheuristic approach to solving the ubqp problem," *European Journal of Operational Research*, vol. 207, no. 3, pp. 1254–1262, 2010.

TABLE V

PEBTS COMPARED AGAINST PITS AND PID^{2}TS ON UBQP INSTANCES WITH SIZE $n \in \{3\,000, 4\,000, \ldots, 7\,000\}$. g_{avr} IS THE AVERAGE GAP TO THE OPTIMAL OBJECTIVE VALUE, suc IS THE SUCCESS RATE OVER THE 100 RUNS, t_{avr} IS THE AVERAGE RUNTIME AND t_{mdn} IS THE MEDIAN RUNTIME (IN SECONDS), $dens$ IS THE DENSITY AND f^* IS THE KNOWN OPTIMAL FUNCTION VALUE FOR THE CORRESPONDING UBQP INSTANCE. THE BEST METRIC VALUES ARE MARKED BY BOLD FONT. FOR EACH ALGORITHM THE LAST ROW ("AVERAGE") SUMMARIZES THE ALGORITHM'S AVERAGE PERFORMANCE OVER ALL THE CONSIDERED INSTANCES.

Instances	$dens$	f^*	PITS				PID^{2}TS				PEBTS			
			g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}	g_{avr}	suc	t_{avr}	t_{mdn}
p3000.1	0.5	3931583	0.0	20	3.07	2.44	0.0	20	3.76	2.24	0.0	20	**1.90**	**1.74**
p3000.2	0.8	5193073	0.0	20	2.84	2.43	0.0	20	2.46	1.98	0.0	20	**1.97**	**1.69**
p3000.3	0.8	5111533	7.3	19	7.44	3.77	0.0	20	8.16	6.61	0.0	20	**5.13**	**3.73**
p3000.4	1.0	5761822	0.0	20	5.47	5.22	0.0	20	**3.12**	**2.94**	0.0	20	3.87	4.09
p3000.5	1.0	5675625	0.0	20	20.36	18.29	1.4	19	19.45	14.64	0.0	20	**9.37**	**7.24**
p4000.1	0.5	6181830	0.0	20	3.36	3.37	0.0	20	4.17	3.83	0.0	20	**2.37**	**2.26**
p4000.2	0.8	7801355	0.0	20	21.57	22.28	0.0	20	16.71	14.63	0.0	20	**11.96**	**10.41**
p4000.3	0.8	7741685	0.0	20	13.08	10.55	0.0	20	**12.04**	9.97	2.7	19	12.21	**5.70**
p4000.4	1.0	8711822	0.0	20	21.91	20.34	0.0	20	29.73	18.18	0.0	20	**12.71**	**10.90**
p4000.5	1.0	8908979	35.2	17	45.80	35.27	0.0	20	33.26	23.99	0.0	20	**21.78**	**17.20**
p5000.1	0.5	8559680	408.6	4	305.73	360.00	**276.1**	**6**	**286.13**	360.00	508.4	2	326.76	360.00
p5000.2	0.8	10836019	228.7	**13**	**192.19**	**182.88**	**186.2**	12	209.20	229.97	291.0	10	192.53	206.69
p5000.3	0.8	10489137	185.8	5	315.20	360.00	**73.8**	10	249.09	318.00	88.5	**15**	**145.91**	**91.82**
p5000.4	1.0	12252318	558.3	3	338.18	360.00	**378.6**	**7**	272.10	360.00	582.5	6	**268.59**	360.00
p5000.5	1.0	12731803	51.2	14	187.12	153.43	**0.0**	**20**	104.44	78.82	51.3	19	**78.10**	**48.04**
p6000.1	0.5	11384976	169.8	6	594.23	720.00	**126.8**	**8**	572.03	720.00	139.4	7	**488.83**	720.00
p6000.2	0.8	14333855	200.2	6	623.08	720.00	**115.8**	**8**	516.52	720.00	169.1	7	**497.18**	720.00
p6000.3	1.0	16132915	1428.1	1	713.34	720.00	**490.4**	9	548.92	720.00	1101.3	**11**	**387.90**	**231.09**
p7000.1	0.5	14478676	750.5	1	688.37	720.00	621.5	3	684.84	720.00	**544.0**	**7**	**498.60**	720.00
p7000.2	0.8	18249948	1078.9	1	689.48	720.00	**546.2**	1	705.89	720.00	669.4	**4**	**590.89**	720.00
p7000.3	1.0	20446407	349.3	4	653.66	720.00	189.8	7	558.25	720.00	**44.5**	**17**	**272.94**	**222.89**
Average			259.6	12.1	259.31	279.06	**143.2**	13.8	230.49	274.57	199.6	**14.5**	**182.45**	**212.64**

[18] Y. Wang, Z. Lü, F. Glover, and J.-K. Hao, "Backbone guided tabu search for solving the ubqp problem," *Journal of Heuristics*, vol. 19, no. 4, pp. 679–695, 2013.

[19] ——, "A multilevel algorithm for large unconstrained binary quadratic optimization," in *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. Springer, 2012, pp. 395–408.

[20] ——, "Path relinking for unconstrained binary quadratic programming," *European Journal of Operational Research*, vol. 223, no. 3, pp. 595–604, 2012.

[21] ——, "Probabilistic grasp-tabu search algorithms for the ubqp problem," *Computers & Operations Research*, vol. 40, no. 12, pp. 3100–3107, 2013.

[22] I. Borgulya, "A parallel evolutionary algorithm for unconstrained binary quadratic problems," in *the 10th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2008, pp. 603–604. [Online]. Available: http://doi.acm.org/10.1145/1389095.1389213

[23] F. Glover and J.-K. Hao, "Efficient evaluations for solving large 0-1 unconstrained quadratic optimisation problems," *International Journal of Metaheuristics*, vol. 1, no. 1, pp. 3–10, 2010.

[24] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: recent advances and new trends," *International Transactions in Operational Research*, vol. 20, no. 1, pp. 1–48, 2013.

[25] T. G. Crainic, "Parallel meta-heuristic search," Interuniversity Research Center on Enterprise Networks, Logistics and Transportation, Tech. Rep. CIRRELT-2015-42, 2015.

[26] D. Sudholt, "Parallel evolutionary algorithms," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 929–959.

[27] I. De Falco, R. Del Balio, E. Tarantino, and R. Vaccaro, "Improving search by incorporating evolution principles in parallel tabu search," in *IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*. IEEE, 1994, pp. 823–828.

[28] A. Al-Yamani, S. M. Sait, and H. R. Barada, "HPTS: heterogeneous parallel tabu search for vlsi placement," in *Congress on Evolutionary Computation, CEC*. IEEE, 2002, pp. 351–355.

[29] A. Bortfeldt, H. Gehring, and D. Mack, "A parallel tabu search algorithm for solving the container loading problem," *Parallel Computing*, vol. 29, no. 5, pp. 641–662, 2003.

[30] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte, "Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem," *Parallel Computing*, vol. 30, no. 3, pp. 377–387, 2004.

[31] R. Banos, C. Gil, J. Ortega, and F. G. Montoya, "A parallel multilevel metaheuristic for graph partitioning," *Journal of Heuristics*, vol. 10, no. 3, pp. 315–336, 2004.

[32] J. Blazewicz, A. Moret-Salvador, and R. Walkowiak, "Parallel tabu search approaches for two-dimensional cutting," *Parallel processing letters*, vol. 14, no. 01, pp. 23–32, 2004.

[33] A. Le Bouthillier and T. G. Crainic, "A cooperative parallel meta-heuristic for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 32, no. 7, pp. 1685–1708, 2005.

[34] E.-G. Talbi and V. Bachelet, "Cosearch: A parallel cooperative metaheuristic," *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 1, pp. 5–22, 2006.

[35] J.-F. Cordeau and M. Maischberger, "A parallel iterated tabu search heuristic for vehicle routing problems," *Computers & Operations Research*, vol. 39, no. 9, pp. 2033–2050, 2012.

[36] J. E. Beasley, "Obtaining test problems via internet," *Journal of Global Optimization*, vol. 8, no. 4, pp. 429–433, 1996.