# A Critical Element-Guided Perturbation Strategy for Iterated Local Search

Zhipeng Lü and Jin-Kao Hao

LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France
zhipeng.lui@gmail.com, hao@info.univ-angers.fr

**Abstract.** In this paper, we study the perturbation operator of Iterated Local Search. To guide more efficiently the search to move towards new promising regions of the search space, we introduce a Critical Element-Guided Perturbation strategy (CEGP). This perturbation approach consists of the identification of critical elements and then focusing on these critical elements within the perturbation operator. Computational experiments on two case studies—graph coloring and course timetabling—give evidence that this critical element-guided perturbation strategy helps reinforce the performance of Iterated Local Search.

**Keywords:** iterated local search, perturbation operator, critical element-guided perturbation, graph coloring, course timetabling.

## 1 Introduction

Local search based metaheuristics are known to be an effective technique for solving a large number of constraint satisfaction and combinatorial optimization problems [14]. However, they may sometimes be trapped into a poor local optimum and it becomes extremely difficult to jump out of it even with more computing efforts. Therefore, diversification mechanisms play an important role in designing such kinds of algorithms.

In order to obtain a tradeoff between intensification and diversification in local search metaheuristics, many kinds of high-level diversification mechanisms have been proposed in the literature to avoid the search to fall into local optima. Typical examples include tabu list in Tabu Search [11], random acceptance criteria in Simulated Annealing, perturbation operator in Iterated Local Search [16], multiple neighborhoods in Variable Neighborhood Search [12] and so on. In particular, it is of significance to utilize low-level problem specific knowledge for constructing strong diversification mechanisms.

In this paper, we study the main diversification mechanism of Iterated Local Search (ILS) [16], i.e., the perturbation operator. ILS is a popular metaheuristic which is mainly composed of two basic components: one is a local search procedure and the other is a perturbation operator. When a local optimum solution cannot be improved any more using the local search, a perturbation operator is employed to produce a new solution, from which a new round of local search

C. Cotta and P. Cowling (Eds.): EvoCOP 2009, LNCS 5482, pp. 1–12, 2009.

starts. It is desirable that the perturbation should be able to guide the search to a promising region of the search space.

As for other components of an ILS procedure, it is useful to integrate problem specific knowledge to make a perturbation operator informative and effective. One fundamental question is then what kind of problem knowledge should be used in the design of an effective perturbation operator.

In this paper, we put forward that the structure of the local optimum solution found so far itself can be used for constructing the perturbation operator. For this purpose, we introduce a new perturbation strategy, called Critical Element-Guided Perturbation (CEGP) for Iterated Local Search (Section 2). We illustrate this on two typical hard combinatorial optimization problems—graph coloring (Section 3) and course timetabling (Section 4), showing its importance in the design of a powerful ILS algorithm.

## 2 Critical Element-Guided Perturbation (CEGP)

Iterated Local Search can be described by a simple computing schema (see Section 2.3 for a general template). A fundamental principle of ILS is to exploit the tradeoff between diversification and intensification. Intensification focuses on optimizing the objective function as far as possible within a limited search region while diversification should be able to drive the search to explore new promising regions of the search space.

The diversification mechanism of ILS—perturbation operator—has two aims: one is to jump out of the local optimum trap just visited; the other is to lead the search procedure to a new promising region. A commonly-used perturbation operator is to destruct partially the previous local optimum solution in a *random* way, not necessarily guided by an evaluation function [16]. Zhang and Sun used the idea of estimation of distribution algorithms to construct perturbation, which combines global statistical information and the location information of good individual solutions for generating new trial solutions [20]. However, we think that the solution structure of the previously obtained local optimum itself can be used in the design of more intelligent and informative perturbation operators.

### 2.1 CEGP Procedure

For the two purposes just mentioned previously, one more elaborated perturbation should take into account the specific problem structure. Given a local optimum solution obtained by the local search procedure, if one can identify the contribution of each *element* to the cost function or constraint violations, then it is reasonable that a perturbation by changing the values of these critical *elements* would be helpful to jump out of local optimum trap. In its simplest form, an *element* can be a decision variable. For example, in the knapsack problem, an element might be an object while in university course timetabling problem, an element can be a lecture of a course.

Generally speaking, our critical element-guided perturbation strategy is composed of three phases: 1. ***Scoring***: give each element a score; 2. ***Selection***: choose

a certain number of highly-scored elements; 3. ***Perturbing***: randomly perturb the solution using the chosen critical elements.

In order to score an element $e_i$ in the ***Scoring*** phase, it is indispensable to define a scoring function and its parameters. Generally, the parameters include the current element $e_i$ and those elements which are strongly related to $e_i$. More formally, an element $e_i$ can be scored as $Score(e_i) = h(e_i, \widehat{e_i})$, where $\widehat{e_i}$ are the elements set related to $e_i$ and $h(\cdot)$ is a scoring function. For a specific problem, in order to score an element $e_i$, it is necessary to define its related elements set $\widehat{e_i}$ and the appropriate scoring function $h(\cdot)$ according to the problem specific knowledge. We will show two examples in Sections 3.5 and 4.4.

The ***Selection*** phase consists of choosing a certain number of elements according to their scores. For the ***Selection*** phase, it is implemented in an adaptive and random way, i.e., the higher score an element has, the more possibly this element is chosen. Note that this selection procedure is problem independent as shown in Algorithm 1 line 6.

After a certain number of critical elements are chosen, the ***Perturbing*** phase randomly perturbs the chosen elements. The perturbation operator can employ the moves in the local search procedure or quite different moves. In Sections 3.5 and 4.4, we respectively use these two kinds of perturbation operators.

### 2.2 CEGP Framework

Given a general constraint satisfaction and optimization problem (CSOP) [19] and a local optimum solution, the proposed critical element-guided perturbation operator is described in Algorithm 1, where $\phi$ is a positive real number and in this paper we empirically set $\phi \in [1.5, 3.0]$. One observes that the value of $\phi$ determines the intensity of the selection procedure: the larger the value of $\phi$ is, the higher is the possibility that the high-score elements are selected. Note that the commonly-used random perturbation is a special case of our CEGP strategy with $\phi = 0$. In this case, the selection probability becomes $P(k) = 1/n$. On the other hand, setting $\phi = \infty$ will select always the first $\eta$ elements.

**Algorithm 1.** Critical Element-Guided Perturbation Strategy

1: **Input**: a local optimum solution $s$
2: **Output**: a perturbed solution $s'$
3: **Scoring**: score each element $e_i, i = 1, \ldots, n$: $Score(e_i) = h(e_i, \widehat{e_i})$
4: sort all the elements in a non-increasing order according to their scores
5: determine a perturbation strength $\eta$
6: **Selection**: randomly select $\eta$ elements to be perturbed. The $r$th critical element is selected according to the following probability:

$$P(r) = r^{-\phi} / \sum_{i=1}^{n} i^{-\phi} \tag{1}$$

7: **Perturbing**: randomly perturb the selected $\eta$ elements
8: get a perturbed solution $s'$

It should be noted that the scoring method (Algorithm 1, line 3) is essential in the CEGP strategy. It must be combined with low level problem specific knowledge, as shown in Sections 3.5 and 4.4. In addition, the perturbation strength $\eta$ should also be determined according to the given problem.

### 2.3 ILS with CEGP

Iterated local search starts with an initial solution and performs local search until a local optimum is found. Then, the current local optimum solution is perturbed and another round of local search is performed to the perturbed solution. The perturbation procedure is implemented by the CEGP strategy just described above. Finally, an acceptance criterion is used to decide whether the new local optimum solution is accepted as the initial solution for the next run of local search. Algorithm 2 shows the pseudocode of CEGP-based ILS. The detailed description of the general ILS procedure can be found in [16].

**Algorithm 2.** Iterated Local Search with CEGP Strategy

1: $s_0 \leftarrow$ Initial Solution
2: $s' \leftarrow$ Local Search($s_0$)
3: **repeat**
4: $\quad s^* \leftarrow$ Critical Element-Guided Perturbation($s'$)
5: $\quad s^{*'} \leftarrow$ Local Search($s^*$)
6: $\quad s' \leftarrow$ Acceptance Criterion($s^{*'}$,$s'$)
7: **until** stop condition met

In order to implement the CEGP strategy for a given problem, we just need to define what an element $e_i$ is, how the related elements of each $e_i$ are identified, how the scoring function is designed and what is the perturbation moves for the chosen elements. In the following two sections, we show two case studies of applying the ILS algorithm with the proposed CEGP strategy to two difficult problems—graph coloring and course timetabling.

## 3 Case Study 1: Graph Coloring

### 3.1 Problem Description

Given an undirected graph $G = (V, E)$ with a set $V$ of $n$ vertices and an edge set $E$ as well as the number of colors to be used $k$, a legal $k$-coloring of graph $G$ is a partition of $V$ into $k$ independent sets where an independent set is a subset of nonadjacent vertices of $G$. In a formal way, let $c_i$ be the color of vertex $v_i$ ($c_i \in [1, k]$, $i = 1, \ldots, n$), a legal $k$-coloring of graph $G$ is a coloring $C = \{c_1, \ldots, c_n\}$ such that $\forall\{v_i, v_j\} \in E$, $c_i \neq c_j$.

Graph coloring aims at finding the smallest $k$ for a given graph $G$ (the chromatic number $\chi_G$ of $G$) such that $G$ has a legal $k$-coloring.

### 3.2 General Solution Procedure

The graph coloring problem can be solved from the point of view of constraint satisfaction by solving a series of $k$-coloring problems. We starts from an initial number of $k$ colors ($k = |V|$ is certainly sufficient) and solve the $k$-coloring problem. As soon as the $k$-coloring problem is solved, we decrease $k$ by setting $k$ to $k$-1 and solve again the $k$-coloring problem. This process is repeated until no legal $k$-coloring can be found.

### 3.3 Initial Solution and Evaluation Function

For a $k$-coloring problem with a given $k$, we generate an initial solution by means of a greedy algorithm presented in [10]. It can be considered as an improved version of the famous DSATUR algorithm [2]. Note that this greedy heuristic generally generates an illegal $k$-coloring.

Once an initial solution is obtained where each vertex has been assigned a color, our CEGP-based ILS algorithm is used to minimize the number of edges having both endpoints with a same color (or the conflict number). Therefore, our evaluation function is just the number of conflicts $f(C)$ such that

$$f(C) = \sum_{\{v_i,v_j\}\in E} \delta_{ij} \tag{2}$$

where

$$\delta_{ij} = \begin{cases} 1, \text{ if } c_i = c_j; \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

Accordingly, any coloring $C$ with $f(C) = 0$ corresponds to a legal $k$-coloring which presents a solution to the $k$-coloring problem.

### 3.4 Local Search Procedure

In this paper, we employ the Tabu Search algorithm presented in [8] as our local search procedure. This TS algorithm is an improved version of the TABUCOL algorithm in [13]. Here a neighborhood of a given configuration is obtained by changing the color $c_i$ of a conflicting vertex $v_i$ to another color $c_j$ ($c_j \neq c_i$), denoted by $(v_i, c_j)$. The cost deviation of the move $(v_i, c_j)$ is denoted by $\Delta f_{(v_i,c_j)}(C)$. More details can be found in [8].

For the tabu list, once a move $(v_i, c_j)$ is performed, vertex $v_i$ is not allowed to receive again the color $c_i$ for the next $tt$ iterations. The tabu tenure $tt$ is empirically determined by $tt = f + random(10)$ where $f$ is the conflict number of the current solution and $random(10)$ takes a random number in $\{1, \ldots, 10\}$. The stop condition of our tabu search is just the maximal number of iterations during which the best solution has not been improved. In this work, we set this number to be 1,000,000 for all the tested instances.

### 3.5 Perturbation

Once the local search procedure stops with a local optimum solution, a Critical Element-Guided Perturbation operator is performed to reconstruct the obtained local optimum solution. Given the general CEGP strategy described in Section 2, one just needs to know how to score each vertex (element) and what is the perturbation operator for the chosen vertices.

Let us first consider the ***Scoring*** phase. For a vertex (an element) $v_i$, its related element set $\widehat{v_i}$ is defined as the set of $v_i$'s adjacent vertices, i.e., $\widehat{v_i} = \{v_j | \{v_i, v_j\} \in E\}$. Based on $\widehat{v_i}$, the following three sets can be derived:

$$V_i^1 = \{v_j | v_j \in \widehat{v_i}, c_i = c_j\} \tag{4}$$

$$V_i^2 = \{v_j | v_j \in \widehat{v_i}, |V_j^1| > 0\} \tag{5}$$

$$K_i = \{c_j | c_j \neq c_i, \Delta f_{(v_i, c_j)}(C) = 0\} \tag{6}$$

Given these notations, the score of a vertex $v_i$ is calculated as:

$$Score(v_i) = h(v_i, \widehat{v_i}) = \omega_1 \cdot |V_i^1| + \omega_2 \cdot |V_i^2| + \omega_3 \cdot |K_i| \tag{7}$$

where $\omega_1$, $\omega_2$ and $\omega_3$ are the associated weights for these three kinds of scores and we set empirically $\omega_1 = 5$ and $\omega_2 = \omega_3 = 1$ respectively.

In the above formulations, $|V_i^1|$ denotes the total number of vertices conflicting with vertex $v_i$: $|V_i^1| = 0$ means that vertex $v_i$ is conflict-free while $|V_i^1| > 0$ implies that vertex $v_i$ is conflicting. $|V_i^2|$ denotes the number of $v_i$'s adjacent vertices which themselves are conflicting. It is easy to observe that $V_i^1$ is a subset of $V_i^2$. The larger $|V_i^1|$ and $|V_i^2|$ are, the higher score vertex $v_i$ has. The rationale behind this is that a conflicting vertex should naturally change its color while the vertex adjacent to a number of conflicting vertices should also be recolored since it would be impossible for the conflicting vertex to become conflict-free in the next round of the local search if its adjacent vertices are not reassigned.

Furthermore, it is reasonable to give a higher priority to a vertex with a large number of *side walks*, where a *side walk* of vertex $v_i$ denotes a move $(v_i, c_j)$ $(c_j \neq c_i)$ that will not change the total cost function, i.e., $\Delta f_{(v_i, c_j)}(C) = 0$. $|K_i|$ represents the number of *side walks* for vertex $v_i$. Note that changing the color of a vertex having a large number of *side walks* will not worsen the solution quality to a large extent, thus the reassignment of these vertices might help the search to jump out of local optimum solution while keeping the solution quality at a good level.

Once all the vertices are scored in accordance with Eq. (7), they are sorted in a decreasing order according to their scores. The perturbation strength is empirically determined by $\eta = 0.33 \cdot n + random(100)$. We observed that a weaker perturbation strength did not allow the search to escape from the local optima. According to Eq. (1) in Algorithm 1, $\eta$ vertices are randomly chosen. After that, we remove all these $\eta$ vertices and reassign them using the greedy heuristic as shown in Section 3.3. This is the ***Perturbing*** phase in our CEGP strategy. Thus, a perturbed solution is obtained, from which a new round of local search starts.

### 3.6 Experimental Results and Comparisons

To evaluate the efficiency of this CEGP-based ILS algorithm, we carry out experiments on a set of 23 non-trivial DIMACS coloring benchmarks. We contrast the results of our CEGP-based ILS algorithm with the uniformly random perturbation strategy (URP) in order to highlight the impact of the CEGP strategy. To make the comparison as fair as possible, the only difference between URP-based and CEGP-based ILS algorithms is that the perturbed vertices are selected in a blindly uniform way with the URP strategy, as described in Section 2.2. Moreover, we compare our coloring results with those of some best performing reference algorithms. Our algorithm is run on a PC with 3.4GHz CPU and 2.0Gb RAM. To obtain our computational results, the total CPU time limit for each instance is limited to 8 hours. Note that the time limit for the reference algorithms is from several hours to several tens of hours.

Table 1 gives our computational results. Column 2 presents the best known $k^*$ ever reported in the literature. Columns 3 and 4 give the results of the CEGP-based and URP-based ILS algorithms respectively, together with the CPU time in minutes (in brackets). Columns 5 to 11 give the results of seven reference algorithms, including four local search algorithms [1,3,6,13] as well as three hybrid algorithms [7,8,9].

One finds that the CEGP-based ILS algorithm obtains smaller $k$ than the URP-based ILS algorithm for 4 out of 23 instances while larger $k$ for only 1 instance, which would suggest the effectiveness of the proposed CEGP strategy. Furthermore, the CEGP-based ILS algorithm obtains comparable results with these famous reference algorithms.

**Table 1.** Computational results of CEGP-based ILS algorithm on graph coloring problem

| | | ILS | | Local Search Algorithms | | | | Hybrid Algorithms | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Instances | $k^*$ | CEGP(t) | URP(t) | [1] | [3] | [6] | [13] | [7] | [8] | [9] |
| dsjc250.5 | 28 | 28 (1) | 28 (2) | — | 28 | 28 | — | 29 | 28 | 28 |
| dsjc500.1 | 12 | 12 (1) | 12 (1) | 12 | 12 | 13 | — | — | — | 12 |
| dsjc500.5 | 48 | 48 (76) | 48 (98) | 49 | 49 | 50 | 51 | 49 | 48 | 48 |
| dsjc500.9 | 126 | 126 (10) | 126 (16) | 127 | 126 | 127 | — | — | — | 126 |
| dsjc1000.1 | 20 | 21 (3) | 21 (4) | 20 | — | 21 | — | — | 20 | 20 |
| dsjc1000.5 | 83 | 87 (35) | 88 (29) | 89 | 89 | 90 | 94 | 84 | 83 | 84 |
| dsjc1000.9 | 224 | 224 (46) | 225 (32) | 228 | — | 226 | — | — | 224 | 224 |
| r125.5 | 35 | 36 (6) | 36 (8) | — | — | 36 | — | — | — | — |
| r250.5 | 65 | 65 (4) | 65 (4) | 66 | — | 66 | — | 69 | — | — |
| r1000.1c | 98 | 98 (7) | 98 (6) | 98 | — | 98 | — | — | — | — |
| r1000.5 | 234 | 253 (38) | 252 (51) | 249 | — | 242 | — | — | — | — |
| dsjr500.1c | 85 | 85 (14) | 85 (19) | 85 | — | — | — | — | — | 86 |
| dsjr500.5 | 122 | 125 (29) | 125 (38) | 126 | 124 | — | — | 130 | — | 127 |
| le450_15c | 15 | 16 (1) | 16 (1) | 15 | 15 | — | 18 | 15 | 15 | 15 |
| le450_15d | 15 | 15 (24) | 15 (34) | 15 | 15 | — | 18 | 15 | — | 15 |
| le450_25c | 25 | 25 (28) | 26 (1) | 25 | 26 | — | — | 25 | 26 | 26 |
| le450_25d | 25 | 25 (34) | 26 (1) | 25 | 26 | — | — | 25 | — | 26 |
| flat300_26_0 | 26 | 26 (3) | 26 (3) | — | 26 | 26 | 32 | 26 | — | 26 |
| flat300_28_0 | 28 | 30 (18) | 30 (15) | 28 | 31 | 31 | 32 | 33 | 31 | 31 |
| flat300_50_0 | 50 | 50 (6) | 50 (8) | 50 | — | 50 | — | 90 | — | 50 |
| flat300_60_0 | 60 | 60 (10) | 60 (17) | 60 | — | 60 | — | 90 | — | 60 |
| flat300_76_0 | 82 | 87 (46) | 87 (69) | 88 | — | 89 | 93 | 84 | 83 | 84 |
| latin_square_10 | 98 | 100 (37) | 100 (34) | — | — | — | — | — | — | 104 |

## 4 Case Study 2: Course Timetabling

The course timetabling problem consists of scheduling all lectures of a set of courses into a weekly timetable, where each lecture of a course must be assigned a period and a room in accordance with a given set of constraints. In this problem, all hard constraints must be strictly satisfied and the weighted soft constraint violations should be minimized. In this paper, we study the curriculum-based course timetabling problem (CB-CTT), which is one of the three topics of the second international timetabling competition (ITC–2007)[1].

### 4.1 Problem Description

In the CB-CTT problem, a feasible timetable is one in which all lectures are scheduled at a timeslot and a room, such that the hard constraints $H_1$-$H_4$ (see below) are satisfied. In addition, a feasible timetable satisfying the four hard constraints incurs a penalty cost for the violations of the four soft constraints $S_1$-$S_4$ (see below). Then, the objective of CB-CTT is to minimize the weighted soft constraint violations in a feasible solution. The four hard constraints and four soft constraints are:

- $H_1$. **Lectures:** Each lecture of a course must be scheduled in a distinct period and a room.
- $H_2$. **Room occupancy:** Any two lectures cannot be assigned in the same period and the same room.
- $H_3$. **Conflicts:** Lectures of courses in the same curriculum or taught by the same teacher cannot be scheduled in the same period, i.e., no period can have an overlapping of students nor teachers.
- $H_4$. **Availability:** If the teacher of a course is not available at a given period, then no lectures of the course can be assigned to that period.
- $S_1$. **Room capacity:** For each lecture, the number of students attending the course should not be greater than the capacity of the room hosting the lecture.
- $S_2$. **Room stability:** All lectures of a course should be scheduled in the same room. If this is impossible, the number of occupied rooms should be as few as possible.
- $S_3$. **Minimum working days:** The lectures of a course should be spread into the given minimum number of days.
- $S_4$. **Curriculum compactness:** For a given curriculum, a violation is counted if there is one lecture not adjacent to any other lecture belonging to the same curriculum within the same day, which means the agenda of students should be as compact as possible.

[1] http://www.cs.qub.ac.uk/itc2007/

### 4.2 Initial Solution, Search Space and Evaluation Function

Starting from an empty timetable, we generate first an initial feasible solution by means of a greedy graph coloring heuristic. We simply mention that for all the 21 competition instances, this greedy heuristic can easily obtain feasible solutions. Once a feasible timetable that satisfies all the hard constraints is reached, our ILS algorithm is used to minimize the soft constraint violations while keeping hard constraints satisfied. Therefore, the search space of our algorithm is limited to the feasible timetables. The evaluation function of our algorithm is just the weighted soft constraint violations as defined for the ITC–2007.

### 4.3 Local Search Procedure

For this problem, we use a Tabu Search algorithm with two distinct neighborhoods defined by two *move*s denoted as *SimpleSwap* and *KempeSwap*. *SimpleSwap* move consists in exchanging the hosting periods and rooms assigned to two lectures of different courses while a *KempeSwap* move produces a new feasible assignment by swapping the period labels assigned to the courses belonging to two specified Kempe chains. Our Tabu Search algorithm explores these two neighborhoods in a token-ring way. More details about these neighborhoods and the TS algorithm are given in [17].

### 4.4 Perturbation

For the ***Scoring*** phase of this problem, an element is just one lecture of a course and the related elements of a lecture are the lectures involved in the calculation of the lecture's soft constraint violations. When the current TS phase terminates with a local optimum solution, the scoring function of a lecture is just the weighted sum of soft constraint violations involving the lecture.

Then, all the lectures are ranked in a decreasing order according to their scores and a number of lectures are randomly selected in accordance with Eq. (1). Finally, the ***Perturbing*** phase consists of randomly selecting a series of *SimpleSwap* or *KempeSwap* moves involving the chosen lectures. Thus, a perturbed solution is obtained from which a new round of Tabu Search starts.

### 4.5 Experimental Results

In this section, we report computational results on the set of 21 competition instances using two formulations of the CB-CTT problem [4]. The first formulation is previously studied by Di Gaspero *et al* in [5] and the second one is just the topic of the ITC–2007. Like in Section 3.5, we contrast the results of our CEGP-ILS algorithm with that of URP-ILS. We also compare our results with the best known results obtained by other algorithms from the literature. To obtain our computational results, each instance is solved 100 times independently and each ILS run is given a maximum of 2,000,000 local search steps.

Table 2 shows the best results of the CEGP-based and URP-based ILS algorithms on the 21 competition instances for both formulations as well as the

**Table 2.** Computational results and comparison on the 21 course timetabling competition instances

| | Old Formulation | | | | | ITC-2007 Formulation | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | best | CEGP(t) | URP(t) | [4] | [15] | best | CEGP(t) | URP(t) | [4] | [15] | [18] |
| comp01 | 4 | 4 (0) | 4 (0) | 4 | — | 5 | 5 (0) | 5 (0) | 5 | — | 5 |
| comp02 | 24 | **20** (35) | 22 (28) | 24 | — | 33 | **29** (64) | 35 (52) | 56 | 33 | 35 |
| comp03 | 39 | **38** (29) | 40 (20) | 39 | — | 66 | 66 (35) | 68 (28) | 79 | — | 66 |
| comp04 | 18 | 18 (36) | 18 (48) | 18 | — | 35 | 35 (9) | 35 (13) | 38 | — | 35 |
| comp05 | 240 | **219** (8) | 224 (4) | 240 | — | 298 | **292** (3) | 310 (5) | 316 | — | 298 |
| comp06 | **16** | 18 (72) | 20 (60) | 25 | 16 | 37 | 37 (99) | 38 (112) | 55 | — | 37 |
| comp07 | 3 | 3 (28) | 3 (36) | 7 | 3 | **7** | 13 (67) | 14 (75) | 26 | — | 7 |
| comp08 | 20 | 20 (18) | 21 (12) | 22 | 20 | **38** | 39 (51) | 41 (40) | 42 | — | 38 |
| comp09 | 59 | **54** (42) | 56 (50) | 59 | — | 99 | **96** (29) | 102 (38) | 104 | 99 | 100 |
| comp10 | **2** | 3 (15) | 4 (17) | 6 | 2 | **7** | 10 (14) | 14 (10) | 19 | — | 7 |
| comp11 | 0 | 0 (0) | 0 (0) | 0 | — | 0 | 0 (0) | 0 (0) | 0 | — | 0 |
| comp12 | 241 | **239** (45) | 242 (31) | 241 | — | 320 | **310** (42) | 315 (61) | 342 | — | 320 |
| comp13 | 33 | **32** (24) | 32 (17) | 36 | 33 | 60 | **59** (70) | 60 (78) | 72 | 60 | 61 |
| comp14 | 28 | **27** (34) | 28 (38) | 29 | 28 | 51 | 51 (66) | 51 (48) | 57 | 51 | 53 |
| comp15 | 39 | **38** (14) | 38 (19) | 39 | — | 70 | **68** (156) | 69 (145) | 79 | — | 70 |
| comp16 | 21 | **16** (95) | 18 (76) | 21 | — | 28 | **23** (171) | 30 (176) | 46 | 28 | 30 |
| comp17 | 41 | **34** (42) | 36 (48) | 41 | — | 70 | **69** (47) | 72 (68) | 88 | — | 70 |
| comp18 | 37 | **34** (21) | 34 (38) | 37 | — | 75 | **65** (125) | 68 (118) | 75 | — | 75 |
| comp19 | 33 | **32** (70) | 33 (52) | 33 | — | 57 | 57 (164) | 57 (143) | 64 | — | 57 |
| comp20 | 14 | **11** (16) | 11 (9) | 14 | — | **17** | 22 (146) | 30 (130) | 32 | 17 | 22 |
| comp21 | 56 | **52** (31) | 53 (46) | 56 | — | **89** | 93 (82) | 96 (70) | 107 | — | 89 |

previous best known results available in the literature. The average CPU time for our CEGP-ILS and URP-ILS algorithms is also indicated in brackets (in minutes). The reference algorithms include the tabu search algorithm in [4], the integer programming algorithm in [15] and the hybrid algorithm in [18]. Interestingly, these reference best known results are from a web site maintained by the organizers of ITC–2007 (track 3)[2], which provides a complete description about the CB-CTT problem and the continuously updated best known results uploaded by researchers (the column "best" in Table 2).

From Table 2, one easily observes that the CEGP-based ILS algorithm reaches quite competitive results. First of all, it performs better than the URP-based ILS algorithm for the majority of the 21 competition instances and no worse result is observed for any instance. In order to testify the influence of the proposed CEGP perturbation operator, we performed a 95% confidence t-test to compare CEGP-ILS with URP-ILS for both formulations. We found that for 13 (respectively 15) out of the 21 instances of the old formulation (respectively ITC-2007 formulation), the difference of the computational results obtained by CEGP-ILS and URP-ILS is statistically significant. In addition, the CEGP-based ILS algorithm improves the previous best known solutions for 14 and 9 out of the 21 instances respectively for the two formulations, showing the strong search power of the CEGP-based ILS algorithm.

## 5 Conclusion and Discussion

The purpose of this paper is to investigate the diversification scheme of Iterated Local Search. To this end, we proposed a general Critical Element-Guided

[2] http://tabu.diegm.uniud.it/ctt/index.php

Perturbation strategy for jumping out of local optimum solution. The essential idea of this perturbation strategy lies in identifying the critical elements in the local optimum solution and adaptively perturbing the solution using these critical elements. This perturbation approach provides a mechanism for diversifying the search more efficiently compared with the commonly-used uniformly random perturbation strategy.

An ILS algorithm with this CEGP strategy was tested on two case studies—graph coloring and course timetabling, showing clear improvements over the traditional blindly uniform perturbation strategy. The results also show that the CEGP-based ILS algorithm competes well with other reference algorithms in the literature.

The practical effectiveness of this strategy depends mainly on the problem-specific scoring method and the perturbation moves that will be performed. For constrained problems, the score for each element can be simply based on the number of violations involved. For optimization problems, such as Job Shop Scheduling (JSS) and TSP problems, the problem-specific knowledge must be explored in order to score each element. For JSS, the elements in the bottleneck machine or the critical path should reasonably have high scores. For TSP, the scores for elements might be marked according to the tour length involved.

To conclude, we believe that the Critical Element-Guided Perturbation strategy helps design high performance ILS algorithm. At the same time, it should be clear that for a given problem, it is indispensable to realize specific adaptations by considering problem-specific knowledge in order to obtain high efficiency.

## Acknowledgment

The authors would like to thank the anonymous referees for their helpful comments. This work was partially supported by "*Angers Loire Métropole*" and the Region of "*Pays de la Loire*" within the MILES and RADAPOP Projects.

## References

1. Blöchliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers and Operations Research 35(3), 960–975 (2008)
2. Brélaz, D.: New methods to color the vertices of a graph. Communications of the ACM 22(4), 251–256 (1979)
3. Chiarandini, M., Stützle, T.: An application of iterated local search to graph coloring. In: Johnson, D.S., Mehrotra, A., Trick, M.A. (eds.) Proceedings of the Computational Symposium on Graph Coloring and its Generalizations, Ithaca, New York, USA, pp. 112–125 (2002)
4. De Cesco, F., Di Gaspero, L., Schaerf, A.: Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, and results. In: Proceedings of the 7th PATAT Conference (2008), `http://tabu.diegm.uniud.it/ctt/DDS2008.pdf`
5. Di Gaspero, L., Schaerf, A.: Neighborhood portfolio approach for local search applied to timetabling problems. Journal of Mathematical Modeling and Algorithms 5(1), 65–89 (2006)

6. Dorne, R., Hao, J.K.: Tabu search for graph coloring, T-colorings and set Tcolorings. In: Voss, S., Martello, S., Osman, I.H., Roucairol, C. (eds.) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, ch. 6, pp. 77–92. Kluwer, Dordrecht (1998)
7. Fleurent, C., Ferland, J.A.: Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In: Johnson, D.S., Trick, M.A. (eds.) DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 619–652. American Mathematical Society, Providence (1996)
8. Galinier, P., Hao, J.K.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 3(4), 379–397 (1999)
9. Galinier, P., Hertz, A., Zufferey, N.: An adaptive memory algorithm for the K-colouring problem. Discrete Applied Mathematics 156(2), 267–279 (2008)
10. Glover, F., Parker, M., Ryan, J.: Coloring by tabu branch and bound. In: Johnson, D.S., Trick, M.A. (eds.) DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 285–307. American Mathematical Society, Providence (1996)
11. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
12. Hansen, P., Mladenović, N.: Variable neighborhood search: principles and applications. European Journal of Operational Research 130, 449–467 (2001)
13. Hertz, A., De Werra, D.: Using tabu search techniques for graph coloring. Computing 39(4), 345–351 (1987)
14. Hoos, H.H., Stützle, T.: Stochastic local search: foundations and applications. Morgan Kaufmann, San Francisco (2004)
15. Lach, G., Lübbecke, M.E.: Curriculum based course timetabling: optimal solutions to the udine benchmark instances. In: Proceedings of the 7th PATAT Conference (2008), http://www.math.tu-berlin.de/~luebbeck/papers/udine.pdf
16. Lourenco, H.R., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Meta-heuristics, pp. 321–353. Springer, Heidelberg (2003)
17. Lü, Z., Hao, J.K.: Adaptive tabu search for course timetabling. European Journal of Operational Research (2009), doi:10.1016/j.ejor.2008.12.007
18. Müller, T.: ITC2007 solver description: A hybrid approach. In: Proceedings of the 7th PATAT Conference (2008), http://www.unitime.org/papers/itc2007.pdf
19. Tsang, E.: Foundations of constraint satisfaction. Academic Press, London (1993)
20. Zhang, Q., Sun, J.: Iterated local search with guided mutation. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 924–929 (2006)