A Sustainable City Planning Algorithm Based on TLBO and Local Search

Ke Zhang\textsuperscript{1,b), Li Lin\textsuperscript{1,c), Xuanxuan Huang \textsuperscript{1}, Yiming Liu \textsuperscript{1}, and Yonggang Zhang \textsuperscript{1,2,a)}

\textsuperscript{1}College of Computer Science and Technology, Jilin University, Changchun 130012, China
\textsuperscript{2}Key Laboratory of Symbol Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China

E-mail: \textsuperscript{a)} zhangyg@jlu.edu.cn, \textsuperscript{b)}monicazhk@163.com, \textsuperscript{c)}xiaoliang102@vip.qq.com

Abstract. Nowadays, how to design a city with more sustainable features has become a center problem in the field of social development, meanwhile it has provided a broad stage for the application of artificial intelligence theories and methods. Because the design of sustainable city is essentially a constraint optimization problem, the swarm intelligence algorithm of extensive research has become a natural candidate for solving the problem. TLBO (Teaching-Learning-Based Optimization) algorithm is a new swarm intelligence algorithm. Its inspiration comes from the "teaching" and "learning" behavior of teaching class in the life. The evolution of the population is realized by simulating the "teaching" of the teacher and the student "learning" from each other, with features of less parameters, efficient, simple thinking, easy to achieve and so on. It has been successfully applied to scheduling, planning, configuration and other fields, which achieved a good effect and has been paid more and more attention by artificial intelligence researchers. Based on the classical TLBO algorithm, we propose a TLBO_LS algorithm combined with local search. We design and implement the random generation algorithm and evaluation model of urban planning problem. The experiments on the small and medium-sized random generation problem showed that our proposed algorithm has obvious advantages over DE algorithm and classical TLBO algorithm in terms of convergence speed and solution quality.

1 Introduction

City has been a significant space for human activity since it was formed. For gathering substances, wealth and technology among a certain range of terrain, city gradually becomes the center of economic trade. Prosperous urban also gains opportunity to develop and promote. However when human went into the 21st century, series of severe problems of city development came along\cite{1}. Firstly, urban population exploited from 10\% of world total population in 1800 to 15\% in 1900, and it will be 50\% or more in 2000 as expected. With the expanding of city scale, earth is bearing grand pressure. Then, interaction and accumulation of urban environment, economy and social issues of different historic period tensing the city problems that are already in serious situation. Therefore, in this sense, only when city performs sustainable, will country or even the whole world develop sustainable. In the long run, studying how to design more sustainable city is the core issue in society development realm\cite{2,3}.

Inspired by the achievements artificial intelligence acquired in solving complex engineering problems, multiple artificial intelligence theories and methods start applying to systematically study on city development and sustainable city planning design. According to data published by United Nations'
projections, global urbanization rate will raise around 70% by 2050 [4]. As a result, crucial questions arise on how to develop conditions for a balance. Take China as an example, The Chinese Dream: A Society Under Construction written by Mars. N and Hornsby. A. says that China plans to annually create 20 whole new cities from now to 2020, around one million inhabitants each, to accommodate farmers in urban environments[5]. One famous instance of adopting artificial intelligence in city planning is a national research plan—SUSTAINS sponsored by French government. This project is aimed at getting urban planners and scholars in artificial intelligence together to plan newly built cities. The project focuses both on the provision of an interactive design tool which informs decision makers of the impacts of their choices, and an interactive communication tool on large tactile surfaces for public engagement.

The process of designing whole new cities is by nature a collaborative endeavor gathering urban planners and decision makers around a coarse-grain map of a territory on which to place urban shapes such as centers, industries, housings, commercial units, public equipment etc. The number of elements as well as their spatial layout needs to be strongly guided by a collection of rules related to social, economic, energy, mobility and sustainability issues. Belin from University of Nantes and his partners brings out the visualized intelligent city planning tools cover four stages [6].

### Figure 1. the visualized intelligent city planning tools cover four stages

This paper’s work focuses on the third stage, automatically positioning urban shapes while enforcing constraints and favoring preferences. In order to handle this issue, we propose a sustainable city planning algorithm based on TLBO and local search. We design and implement algorithm for creating random city planning task and the solution evaluating model, on basis of which, we carry out simulation experiment. The experiment result shows for middle and small scale of random data, our algorithm has notable advantage among differential evolution algorithm and original TLBO algorithm. Loosing completeness is not an issue for our application: it is pointless to pay a high computational cost to build optimal solutions, since these solutions will be modified by hand afterwards. We thus do not need optimality proofs.

2. **Fundamental introduction of basic algorithms**

In this section, every algorithm will use FLUSH function to refresh the original swarm. FLUSH is to compare the new swarm calculated from the original one with the old swarm, the original one, and if a member in new swarm performs better, the individual in new swarm substitute the old one in original swarm.

| FLUSH |
|-------|
| $f(X_i)$ is the performance of $X_i$ |

2
1: function Flush(class, oldclass)
2: for (i = 0 ; i < N ; i++) {
3:   if \( f(X_i) \) in oldclass is better then
4:       Xi in class \( \leftarrow \) Xi in oldclass
5:   end if
6: }
7: return class
8: end function

2.1 Teaching-Learning-Based-Optimization (TLBO) Algorithms
TLBO algorithms (Teaching-Learning-Based-Optimization algorithms) is a novel swarm intelligence optimization algorithm proposed by Rao and his team in 2010[7]. It simulates the teaching processes both between teachers and students and between every two students in order to perform evolution by means of the ‘teaching’ process by teachers/students and ‘learning’ process by students. TLBO algorithm has few parameters and is easy to understand, also it possesses high efficiency in calculating precise solutions with strong convergence ability. Though it’s not long since TLBO being proposed, it draws messes of attention by scholars and it is being wildly applied[8].

TLBO:
: \( X \) is an array of students, which equals class , \( N \) is the number of students in class
: \( f(X_i) \) is the performance of \( X_i \), \( teacher \) is the student with best \( f(X_i) \) in class
: \( mean \) is the average \( f(X_i) \) in class
: \( depth \) is the iterate times
: input : class
: output : class
1: oldclass \( \leftarrow \) class, using oldclass to record the original class
2: mean \( \leftarrow \sum_{i=0}^{N-1} f(X_i)/N \)
3: teacher \( \leftarrow \) MAX(f(Xi))
4: while depth not reached do
5:   for(i = 0 ; i < N ; i++)
6:      Xi learn from teacher
7:   FLUSH(class, oldclass)
8:   for(i = 0 ; i < N ; i++)
9:      choose random \( j \) (\( j \neq i \)) , \( X_i \) learn from \( X_j \)
10:  FLUSH(class, oldclass)
11:  choose new teacher and calculate new mean in class
12: end while
13: return class

2.2 Local Search
Local Search [9](LS) algorithm is a typical greedy algorithm, it peeks a best solution from the current solution domain as its temporary answer every time until it reaches a local optimal result. This meta-heuristic takes advantage of the structure of the problem in terms of constraints and variables to guide the search. Local Search has proven its efficiency on large and various instances. The implement of Local Search can traced back to late 50s and early 60s in 20 century[9]. In former Soviet Union, research in this area started from Zhuravlev, who putted forward the algebraic theory of Local Search, while in the meantime, Rasitirgin focused on Local Search in probability. Local Search succeeded in calculating TSP (Traveling Salesman Problem) problems in early time, and later it is selected to solve location problems, network designations, scheduling problems and configuration problems. However, only with Local Search don’t guarantee the optimal result in global. The practical applications of Local Search mostly are searching compound schemes or decomposition scheme originated from multiple strategies and finding approximate solutions in complex discrete problems[10].
Local Search

: depth is the iterate times
: f is the global cost function, fi is the cost function for variable Vi
: s is the current configuration
: T is the adaptive tabu list
: j is the index of the variable with the worst cost

1: s←Vk  // k is a random value
2: T←∅
3: while depth not reached do
4:   while T is not full do
5:     For all i such that Vi∉T  do compute fi(s) end for
6:     Select Vi a variable for which f(s) is maximum ;
7:     Compute the cost f of the configuration obtained from s by swapping Vi with another variable
8:     Select s0 the configuration for which f(s0) is minimum;
9:     Update T by removing its oldest variable ;
10:    if s0 can improve current solution s then
11:       s←s0
12:  else
13:     T←T∪Vj
14:    end if
15: end while
16: end while
17: return s

The basic idea is to compute the error function for each constraint, then combine for each variable the errors of all the constraints in which it appears, thereby projecting constraint errors onto the relevant variables. This combination of errors is problem-dependent. In our case, it is a weighted sum, so that the constraints can be given different priorities.

3. Algorithm based on tlbo and local search

3.1 TLBO_LS Algorithm

In order to solve the problem of urban planning, this paper proposes a sustainable city planning algorithm based on TLBO and Local Search, hereinafter referred to as TLBO_LS. TLBO_LS is based on the demand of simulating a more realistic teaching process. According to the above, the learning steps of TLBO only includes "learning from teacher" and "learning from students", whereas the practical situation, self-study is also an indispensable step in the teaching process. It can be seen that LS is an algorithm that changes the elements in a collection and does not change relationships between elements. Thus, the improved algorithm will use LS to simulate "self-learning" to enhance the efficiency of class performance improvement, in the meantime avoid the local optimal generation, so as to get a more appropriate solution. The specific procedure of TLBO_LS is explained as follows.

**TLBO_LS**

: X is an array of students, which equals class,  N is the number of students in class
: f(Xi) is the performance of Xi,  teacher is the student with best f(Xi) in class
: mean is the average f(Xi) in class
: depth is the iterate times
: input : class
: output : class

1: oldclass←class, using oldclass to record the original class
2: mean ↦ \frac{\sum_{i=0}^{N-1} f(X_i)}{N}
3: teacher ↦ \text{MAX}(f(X_i))
4: while depth not reached do
5:   for(i = 0 ; i < N ; i++)
6:       \text{Xi learn from teacher}
7:     FLUSH(class, oldclass)
8:   for(i = 0 ; i < N ; i++)
9:      \text{choose random } j (j \neq i) , \text{Xi learn from } X_j
10:   FLUSH(class, oldclass)
11:   for(i = 0 ; i < N ; i++)
12:      \text{Xi learn by itself using Local Search}
13:   FLUSH(class, oldclass)
14:   choose new teacher and calculate new mean in class
15: end while
16: return class

Comparing to the LS, TLBO_LS not only needs to search around the individual, but also calculate with other individuals in the population inducing the possibility to jump out of local optimality, then find a better solution. Compared with TLBO, TLBO_LS can broaden the scope of the calculation among the swarm individuals, so that the content able to be searched distributes in a larger interval. Therefore, the probability of finding a better solution will be increased. With appropriate size of swarm and iteration times will also lead TLBO_LS to find a more optimistic solution for large-scale data. Similarly, differential evolution (DE) is an algorithm for calculating the interaction between individuals within a population. However, the computational cost of DE is prohibitive. Since the computational efficiency of DE is influenced by various parameters, TLBO_LS possesses more advantages.

3.2 Constraint Model for City Planning Problem

| Parameters | Descriptions |
|------------|--------------|
| City       | Planning object |
| N*M        | Planning object size. |
| S          | Planning results |
| N_A        | Total number of architecture types |
| A_i        | Architectures of i type |
| n_i        | Number of A_i planned to be built. |

In the Table 1 listed five parameters used in our model, here are their specific explanations. In order to quantify the results clearly and straightforward, this paper makes the following assumptions about the model: the planned city can be approximated as a rectangle. In our model, in consideration of actual living conditions, common buildings are divided into 12 types: open space, residential, administrative office buildings, educational buildings, medical buildings, commercial buildings, theatres, gymnasium, hotels, traffic hubs, parks, factories. While calculating, an A_i values i. The model completes the planning task for newly built city using the parameters mentioned above in the Table 1. First, the size of the city is settled when user enters N and M. Next user enters n_i as the expected number of Architecture A_i according to user’s demand. After the execution of the whole algorithm, the planning result S is obtained.

Assuming that the constraints are divided into two categories:

a) It is acknowledged that an architecture A_j (i \neq j) is not expected to be set within a certain range of architecture A_i. For example, a factory is built near the house or a business area is located near the school.

b) It is inappropriate to see that an architecture A_i does not appear in a certain area of the architecture A_j (i \neq j). For example, no hotel is set near the hospital or no school is established near the house.
3.3 City Planning Problems Solving

In order to model the practical problems, the specific variables will be given corresponded practical significance. Class will be a set of various planning results \( X_i \) \((i \in [1, N*M])\); \( f(X_i) \) is the cost of each \( X_i \); the most appropriate with lowest cost planning result \( X_i \) will become teacher; and the mean is a plan with around average cost of the Class. The total number of architectures’ kind is \( N_A \)

This algorithm is based on users’ needs, that is, corresponding number of various types of building \( A_i \) is already determined by customers before calculating urban planning solutions. Therefore, to compute among Class and meet the demand at the same time, this paper puts forward a method in calculating every building’s location. The specific process states as follows:

I. \( X_i \) uses one-dimensional array with capacity of \( N*M \) to demonstrate. The value \( j \) on a position \( k \) in this one-dimensional array means a building \( A_j \) is placed at the position \( k \) in this plan \( X_i \).

II. \( X_i \) learns to \( X_i \) following these regulations. First, initialize temporary result \( X_i' \) as NULL. While traversing \( X_i \), at the position \( k \) finds the \( m \)-th \( A_i \) of \( X_i \). Meanwhile, in \( X_i \) search for the \( m \)-th \( A_i \) of \( X_i' \) and learn that its position is \( k_i \). Then do learning calculations between \( k_i \) and \( k_j \). When \( X_i \) is teacher, use the computational method of learning from teacher. Or if \( X_i \) is a student in Class, choose the learning from students way to compute. The new position \( k_i' \) is obtained by using the rounding and modulo calculation on the output of learning computing. In the temporary planning \( X_i' \), if there is already existed building at the position \( k_i' \) in \( X_i' \), then \( k_i' = k_i + 1 \) (mod \( N*M \)) until there is no building on \( k_i' \) in \( X_i' \). Then, \( A_i \) is settled down on the \( k_i' \) in \( X_i' \).

a. learning from teacher

When \( X_i \) is trying to learn from teacher:

1. First, according to original TLBO, we need to define the \( TF \) (teaching factor), a teaching factor that decides the value of mean to be changed. \( TF \) is computed from \( r \), a random number in the range of \((0,1)\), and \( TF \) is chosen either 1 or 2, which is again a heuristic step and decided randomly:

\[
r = \text{rand}(0,1) \quad (1)
\]

\[
TF = \text{round}[1 + r] \quad (2)
\]

2. Next, the difference \( [k] \) is required as describing the difference of the value at position \( k \) between teacher and mean.

\[
difference[k] = r \ast (\text{teacher}[k] - TF \ast \text{mean}[k]) \quad (3)
\]

3. This difference \( [k] \) modifies the existing solution according to the following expression, the total number of architectures’ kind is \( N_A \)

\[
X_{new}[k] = \text{round} \left( X_{old}[k] + \text{difference}[k] + N_A \right) \mod N_A \quad (4)
\]

b. learning from students

When \( X_i \) is trying to learn from \( X_i' \):

1. First, \( r_i \) is needed to simulate the learning ability of \( X_i \):

\[
r_i = \text{rand}(0,1) \quad (5)
\]

2. Then, \( X_i \) learn from \( X_i' \) according to the follow expression:

\[
X_{new}[k] = (X_{old}[k] + r_i \ast |X_{old}[k] - X_{old}[k]| + N_A) \mod N_A \quad (6)
\]

III. Self-study of \( X_i \) will reference to the idea of Local Search. Firstly, initialize a temporary empty plan \( X_i' \). Traversing \( X_i \) to place the current \( A_i \) building \( x \) on the position \( k_i' \), which is obtained by
calculating original position $k$ of current $A_i$ building $x$ using \textit{learning by itself}, $k$ will compute with a random number and then output the value $k'$ within domain $\delta$, the new position $k'$ is obtained by rounded and modulo operation on original result $k'$. In the temporary planning $X'$, if there is an already existed building at the position $k'$ in $X'$, then $k' = k' + 1 \pmod{N \times M}$ until there is no building on $k' \in X'$. Then, $A_i$ is settled down on the $k'$ in $X'$.

\textit{learning by itself}:

When $X'$ is trying to learn by itself:

(1) First, $r_{i,k}$ is needed to ensure the new position of the architecture on $X_{old}[k]$:

$$r_{i,k} = \text{rand}(\delta) \mod N \times M \quad (7)$$

(2) Then, $X'$ learn by itself according to the follow expression:

$$X_{new}[r_{i,k}] = X_{old}[k] \quad (8)$$

4. Experimental results and analysis

4.1 Experimental Design

In order to evaluate planning results, we designed an evaluation model applying scoring methods. Score will be accumulated according to the number of violated constraints. Specific calculating regulars based on constraints states as follows:

(1) If there is a kind of building $A_{i}(i \neq j)$ in a certain range of building $A_i$, it is scored from 5 to 1 according to the degree of intolerance in real life. For example, if the factory stands near the house, that is unbearable, we assume it values 5 points; if the school is established near business district, though it is unreasonable, we can tolerate it, then it weigh 3 points.

(2) If there is no building $A_{i}(i \neq j)$ in a certain range of building $A_i$, it is scored from 5 to 1 according to the degree of intolerance in real life. For example, if there is no hotel in the vicinity of the hospital, that is unbearable, we give it 5 points; if there is no school near the residential area, though it is unreasonable, we can tolerate it, then it weigh 3 points.

Judgment of result depends on its score, the lower the score, the better the plan. In this paper, a variety of comparative tests were performed on different sizes (10x10, 50x50) grids to assess the quality of the results and the actual computational costs. 10x10 grid represents 100 square kilometers, which is roughly the size of a county in China. 50x50 grid represents 2500 square kilometers, that is about a small and medium sized city in China[12]. Due to the planning task focuses on the initially opened up city, we do not need to consider the architectural factors already exist in city. Based on the planning data of Marne-la-Vallée from Bruno[6], we allocate all kinds of buildings and areas referring to the needs of various types of architectures. We will use the data in Table 2 to conduct experiment. In order to observe the efficiency of TLBO_LS algorithm, we will use TLBO, DE [11] to calculate the same data, and evaluate the planning effect of these three algorithms respectively. Experimental environment is i5-4210U 1.70GHz CPU, Windows 7 (32-bit) operating system and Java programming language. Experiments will test swarms of 100 population in 100 times of iteration by TLBO_LS, DE, TLBO respectively.

| Urban Size | Number of Architecture                                      |
|------------|------------------------------------------------------------|
| 10*10      | open space*7, residence*20, administrative office building*10, school*5, hospital*8, shopping plaza*5, theatre*5, sports center*5, hotel*5, traffic hub*5, park*20, industrial factory*5 |
| Small-scale| open space*200, residence*500, administrative office building*200, school*100, hospital*200, shopping plaza*150, theatre*100, sports center*100, hotel*150, traffic hub*100, park*500, industrial factory*200 |
| 50*50      | large-scale                                               |

TABLE 2 Experimental Data
4.2 Experimental Results and Analysis

The experimental results of DE/TLBO/TLBO_LS are evaluated by five parameters. Initial Cost is the cost of the initial urban planning; Final Cost represents the planning result’s cost after using algorithms; Time is the total time consumed during calculation; Reduction is how the initial planning being improved, and the formula is:

\[
\text{Reduction} = \text{Initial Cost} - \text{Final Cost}
\]  \hfill (9)

The last parameter is Convergence Rate used for evaluating convergence rate of the algorithm with the expression below:

\[
\text{Convergence Rate} = \frac{\text{Time}}{\text{Reduction}}
\]  \hfill (10)

4.2.1 Experimental Result and Analysis of Small-Scale City

| TABLE 3 | Results of Testing Small-scale Data Using DE |
|---------|-----------------------------------------------|
| Initial Cost | Final Cost | Time (ms) | Reduction | Convergence Rate (Time/Reduction) |
| 377       | 93         | 23121     | 284       | 81.41                                |
| 385       | 105        | 21535     | 280       | 76.91                                |
| 377       | 76         | 22377     | 301       | 74.34                                |
| 369       | 110        | 20758     | 259       | 80.15                                |
| 364       | 157        | 20409     | 207       | 98.59                                |
| 396       | 157        | 21098     | 239       | 88.28                                |
| 396       | 210        | 22215     | 186       | 119.44                               |
| 391       | 161        | 21371     | 230       | 92.92                                |
| 384       | 174        | 21556     | 210       | 102.65                               |
| 372       | 99         | 23560     | 273       | 86.3                                  |

| TABLE 4 | Results of Testing Small-scale Data Using TLBO |
|---------|-----------------------------------------------|
| Initial Cost | Final Cost | Time (ms) | Reduction | Convergence Rate (Time/Reduction) |
| 398       | 196        | 1036      | 202       | 5.13                                 |
| 375       | 188        | 1017      | 187       | 5.44                                 |
| 380       | 194        | 1048      | 186       | 5.63                                 |
| 391       | 192        | 1069      | 199       | 5.37                                 |
| 377       | 205        | 1055      | 172       | 6.13                                 |
| 370       | 193        | 1043      | 177       | 5.89                                 |
| 378       | 203        | 1090      | 175       | 6.23                                 |
| 371       | 189        | 1040      | 182       | 5.71                                 |
| 363       | 188        | 1025      | 175       | 5.86                                 |
| 388       | 197        | 1054      | 191       | 5.52                                 |

| TABLE 5 | Results of Testing Small-scale Data Using TLBO_LS |
|---------|-----------------------------------------------|
| Initial Cost | Final Cost | Time (ms) | Reduction | Convergence Rate (Time/Reduction) |
| 374       | 170        | 1843      | 204       | 9.03                                 |
| 374       | 162        | 1855      | 212       | 8.75                                 |
| 373       | 156        | 1846      | 217       | 8.51                                 |
| 390       | 156        | 1812      | 234       | 7.74                                 |
| 370       | 167        | 1810      | 203       | 8.92                                 |
| 387       | 169        | 1874      | 218       | 8.60                                 |
Referring to the Fig. 2, DE algorithm is the best one in the small-scale problem, for its promotion of the original population is significantly larger than the other two algorithms. However, it is so time-consuming that the required time reaches 10 to 20 times of the other two algorithms. As for TLBO algorithm, the quick speed and high promotion in the unit time is definitely outstanding. While the planning result of TLBO is the worst among three. TLBO_LS algorithm has the same advantages as the TLBO algorithm as similar time-consuming, and TLBO_LS can achieve as appropriate result as DE algorithms do in shorter time. In short, TLBO_LS algorithm is more suitable in solving small-scale task than the other two algorithms.

4.2.2 Experimental Result and Analysis of Large-Scale City

Table 6 Results of Testing big-scale data using DE

| Initial Cost | Final Cost | Time (ms) | Reduction | Convergence Rate (Time/Reduction) |
|--------------|------------|-----------|-----------|----------------------------------|
| 8243         | 6038       | 543534    | 2205      | 246.50                           |
| 8290         | 6011       | 530389    | 2279      | 232.73                           |
| 8300         | 6035       | 536070    | 2265      | 236.68                           |
| 8279         | 5969       | 529212    | 2310      | 229.10                           |
| 8282         | 5906       | 546572    | 2376      | 230.04                           |
| 8330         | 6403       | 532223    | 1927      | 276.19                           |
| 8357         | 5718       | 534037    | 2639      | 202.36                           |
| 8333         | 5701       | 532607    | 2632      | 202.36                           |
| 8175         | 6241       | 529058    | 1934      | 273.56                           |
| 8366         | 5909       | 529067    | 2457      | 215.33                           |
TABLE 7 Results of Testing big-scale data using TLBO

| Initial Cost | Final Cost | Time (ms) | Reduction | Convergence Rate (Time/Reduction) |
|--------------|------------|-----------|-----------|-----------------------------------|
| 8285         | 5662       | 86890     | 2623      | 33.13                             |
| 8304         | 5526       | 87194     | 2778      | 31.39                             |
| 8223         | 5506       | 87375     | 2717      | 32.16                             |
| 8258         | 5363       | 86565     | 2895      | 29.90                             |
| 8218         | 5533       | 87344     | 2685      | 32.53                             |
| 8291         | 5454       | 86845     | 2837      | 30.61                             |
| 8299         | 5441       | 88817     | 2858      | 31.08                             |
| 8234         | 5436       | 86598     | 2798      | 30.95                             |
| 8355         | 5540       | 88777     | 2815      | 31.54                             |
| 8226         | 5534       | 90193     | 2692      | 33.50                             |

TABLE 8 Results of Testing big-scale data using TLBO_LS

| Initial Cost | Final Cost | Time (ms) | Reduction | Convergence Rate (Time/Reduction) |
|--------------|------------|-----------|-----------|-----------------------------------|
| 8293         | 5194       | 350776    | 3099      | 113.19                            |
| 8249         | 5168       | 342061    | 3081      | 111.02                            |
| 8323         | 5205       | 341192    | 3118      | 109.43                            |
| 8352         | 5184       | 322725    | 3168      | 101.87                            |
| 8235         | 5108       | 336676    | 3127      | 107.67                            |
| 8231         | 5010       | 341577    | 3221      | 106.05                            |
| 8229         | 4963       | 360743    | 3266      | 110.45                            |
| 8221         | 4853       | 364556    | 3368      | 108.24                            |
| 8269         | 5761       | 326394    | 2508      | 130.14                            |
| 8397         | 4466       | 340168    | 3931      | 86.53                             |

**Figure 3** Comprehensive Analysis of Planning Results of Small-scale City

According to Fig 3, in the large-scale urban planning, the DE’s results, which are the worst of the three planning results, are in the opposite of its performance in small-scale city. It’s extremely time-consuming, and thus it is not suitable for large-scale urban planning. The advantage of TLBO’s speed is obvious and it costs far less time than DE, comparing to the smaller scale situation. As a result, TLBO is the ideal planning algorithm in smaller scale. As the optimization of TLBO, TLBO_LS provides the best planning results in little time, and the planning procedure is highly efficient. To sum up, TLBO_LS is an excellent large-scale urban planning algorithm.
5. Conclusions
This paper proposes a constraint model based on city planning, and meanwhile, on the basis of the classical TLBO algorithm, this paper brings about a TLBO_LS algorithm combined with local search, designs and implements both the random generation algorithm and evaluation model of urban planning problem. The experimental results show that the proposed algorithm outperforms the best-known discrete differential evolution algorithms or TLBO and some evolutionary algorithms due to its outstanding convergence rate and quality of solutions. TLBO_LS shows promise for a reliable and efficient constraint optimization city planning problem.

Acknowledgments
This work is supported by the National Natural Science Foundation of China (61170314,61373052), the Project of Jilin Provincial Science and Technology Development(20170414004GH).

Zhang Ke, born in 1997. BSc (expected). Her major is network and information security.
Lin li, born in 1997. BSc (expected). His major is network and information security.
Huang Xuanxuan, born in 1996. BSc (expected). Her major is network and information security.
Liu Yiming, born in 1996. BSc (expected). His major is network and information security.
Zhang Yonggang, born in 1975. PhD, associate professor. Member of China Computer Federation. His main research interests include Constraint solving and Optimization.

References
[1] Ye Zhongnan. The Use Of Computer Simulation Technology In Urban Design[J]. Planners, 2014, 30(4): 40-46(in Chinese)
[2] LONG Ying, HAN Haoying, GU Yizhen, et al. Spatio-temporal Evaluation of Urban Planning Implementation[J]. PROGRESS IN GEOGRAPHY, 2011, 30(8): 967-977(in Chinese)
[3] Zhao Sidong, Ouyang Dong, Zhong Yuan. Intelligent City Planning’s Effect To Urban Planning[J]. Planners, 2103, 29(2): 5-10(in Chinese)
[4] United Nations. Department of Economic and Social Affairs. World urbanization prospects: the 2007
revision. Highlights.[J]. New York New York United Nations Feb, 2007, 22(8):656-659

[5] Hornsby A. The Chinese Dream: A Society Under Construction[J]. Architectural Record, 2008, 196(7):55-55

[6] Belin B, Christie M, Truchet C. Interactive Design of Sustainable Cities with a Distributed Local Search Solver[M]// Integration of AI and OR Techniques in Constraint Programming. Springer International Publishing, 2014:104-119

[7] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems[J]. Computer-Aided Design, 2011, 43: 303-315

[8] WANG Pei-chong, MA Yue, GENG Ming-yue, WANG Shen-wen. New Teaching-Learning-Based Optimization with Neighborhood Structure Based on Small World[J]. Computer Science and Exploration, 2016, 10(9): 1341-1350(in Chinese)

[9] Codognet P, Diaz D. Yet Another Local Search Method for Constraint Solving[M]// Random Algorithms: Foundations and Applications. Springer Berlin Heidelberg, 2001:73-90

[10] Caniou Y, Codognet P, Diaz D, et al. Experiments in parallel constraint-based local search[C]// European Conference on Evolutionary Computation in Combinatorial Optimization. Springer-Verlag, 2011:96-107.

[11] Storn R, Price K. DE-a simple and efficient heuristic for global optimization over continuous space[J]. Journal of Global Optimization, 1997, 114(4):341-359

[12] Wang Yang, Wang Sahojian, Qin Jin. Spatial evaluation of land urbanization level and process in Chinese cities[J]. Geographical Research, 2014, 33(12): 2228-2238(in Chinese)