Group Theory-based Optimization Algorithm (GTOA)
----based on Genetic Algorithm

Zhang Zichao

Genetic Algorithm

In some optimization problem, the feasible solutions are vectors with a known length, and the i-th components are in {0, 1, … ,m_i}, in which m_i is an integer. Our aim is to find a solution with quality as high as possible, and the quality (an integer) is easy to calculate.

In this case, we can think the vectors (with the known length) as chromosomes and think the components as genes.

Genetic Algorithm

First, generate some vectors (chromosomes) randomly (or in another way) with a number NP. These are the first generation.

Then, select a part of them to do the next operations.

Genetic Algorithm

For the selected vectors, do crossover to generate new vectors. Usually, there are several input vectors and one or more output vectors. The input vectors can be selected randomly or in another way.

For the generated vectors in the previous step, do mutation operation.

Genetic Algorithm

Up to now, the main steps of generating one generation has been completed. However, there may be infeasible vectors in the generation, so we need to repair them. Usually, we use some simple algorithms (such as greedy algorithm).

Repeat these steps until reaching the stopping condition, and choose the vector with the highest quality (fitness) to be the optimal solution.

Genetic Algorithm

Some ingredients in genetic algorithm:

- Encoding mode
- First generation
- Select operator
- Crossover operator
- Mutate operator
- (Greedy) Repaire operator
- (Greedy) Optimize operator
- Fitness
- Stopping condition

Group Theory-based Optimization Algorithm (GTOA)

- GTOA can be used in problems whose solutions (vectors) are with components in different ranges.
- We Implementated GTOA in three different knapsack problems: SUKP, D{0-1}KP, BKP. The main difference of the three algorithms (for the three problems) is the encoding mode.

Group Theory-based Optimization Algorithm (GTOA)

- SUKP: All components are 0 or 1.
- D{0-1}KP: All components are 0, 1, 2 or 3.
- BKP: The i-th components is in $\{0, 1, \ldots, m_i\}$, in which m_i can be all different integers.
- Besices, there are some other differences, such as the definitions of fitness, the repair operators, and even the max iteration times.

An Introduction of BKP

There are n types of objects. The i-th type is with profit p_i and weight w_i. The number (bound) of the i-th type objects is b_i.

We have a knapsack, and the total weight we can put in is limited. Our aim is to put in objects with profit as high as possible.

We can use a vector with length n, each components of which is the number of each object we put in, to represent the solutions.

GTOA for BKP

- First generation: All components are a random integer in their ranges.
- Select operator: /
- Crossover operator:

input: X_1, X_2, X_3 (selected randomly)
output: $Y=X_1+F(X_2-X_3)$, and F is a random vector in $\{-1, 0, 1\}^n$
(Regard the i-th components as elements in Z_{mi+1}, then $[x_i]=[x_i+m_i+1]$ for all $x_i \in Z$)

GTOA for BKP

- Mutate operator: (Parameter p_m is the probability for each component to change)
For each component x_i, it change to $-x_i$ (equivalent to m_i+1-x_i) for probability $p_m/2$, and change to a random number not equivalent to x_i for probability $p_m/2$, else don't change.

GTOA for BKP

- Repair operator: (Suppose X is the input vector)

Sort all elements in profit per weight, such that $p_0/w_0 \geq p_1/w_1 \geq \ldots \geq p_{n-1}/w_{n-1}$.

for i=0 to n-1

If the i-th element of number x_i can not be put in, then let x_i be the max amount that the i-th element can be put in, and $x_{i+1}=\ldots=x_{n-1}=0$.

end for

GTOA for BKP

- Optimize operator: (Suppose X is the input vector)

Sort all elements in profit per weight, such that $p_0/w_0 \geq p_1/w_1 \geq \ldots \geq p_{n-1}/w_{n-1}$.

for i=0 to n-1

Let x_i be as large as possible, such that $x_i \leq m_i$, and the weight doesn't exceed the limit.

end for

GTOA for BKP

- Exactly, the repair operator and the optimize operator is implemented together, in just one operator.
- Fitness: Equals the total profit.

(For the i-th vector in the new generation, compare the fitness of the new generated vector and the i-th vector in the previous generation, and put the larger one into the new generation.)

GTOA for BKP

- Stop condition: When the iteration times reach the value MIT (max iteration times) =”length of X” * 2, stop.

GTOA for D{0-1}KP

- First generation: All components are a random integer in {0, 1, 2, 3}.
- Select operator: /
- Crossover operator: Similar to BKP.
- Mutate operator: Similar to BKP.

GTOA for D{0-1}KP

- Repair operator, optimize operator: Similar to BKP, but the definition of “profit per weight” is a little different.
- Fitness: Also equals to the total profit.
- Stop condition: Also when the iteration times reach MIT, but now MIT=length * 12.

GTOA for SUKP

- First generation: All components are 0 or 1 randomly.
- Select operator: /
- Crossover operator: Similar to BKP: $Y=X_1+F(X_2-X_3)$, regarding all components as residual class. Since all components are 0 or 1, the equation is equivalence to $Y=X_1$ xor (F and (X_2 xor X_3)), which is simpler.

GTOA for SUKP

- Mutate operator: For probability p_m, $x_i=1-x_i$.
- Repair operator, optimize operator: Similar to BKP, but the definition of “profit per weight” is a little different.
- Fitness: Also equals to the total profit.
- Stop condition: Also when the iteration times reach MIT, but MIT=max{item amount, element amount}.

Group Theory-based Optimization Algorithm (GTOA)

The whole algorithm:

Generate NP vectors randomly
in gen[NP][length]
for ch_th=0 to NP-1
 repair gen[ch_th]
end for
for it=0 to MIT-1
 for ch_th=0 to NP-1
  $Y=X_1+F(X_2-X_3)$
  mutate(Y)

repair(Y)

if fit(Y)>fit(gen[ch_th])

new_gen[ch_th]=Y

else

new_gen[ch_th]=gen[ch_th]

end for

gen=new_gen

Group Theory-based Optimization Algorithm (GTOA)

I have implemented GTOA and EGA in C language, compiled with clang, running in Debian. Compare to EGA (modified according to specific problems, so may not be standard EGA), GTOA has better convergence speed and can give solutions with larger fitness.