Computer Science and Information Systems 21(4):1547–1565 https://doi.org/10.2298/CSIS230907047Y

A Hybrid GA-Powell Algorithm for Geometric Constraint Solving

Sun Yunlei[1] and Li Yucong[1]

Qingdao Institute of Software, College of Computer Science and Technology,
China University of Petroleum (East China)
Qingdao, 266580, China
sunyunlei@upc.edu.cn
s21070036@s.upc.edu.cn

Abstract. Geometric constraint solvers are crucial for computer-aided design (CAD), and their algorithms are the focus of research. Current geometric constraint solvers based on traditional numerical methods lack support for multi-solution problems, so we propose a hybrid algorithm that combines the genetic algorithm, which is good at global convergence, and Powell's method, which is good at local refinement, to address the limitations of traditional numerical methods in geometric constraint solving algorithms (sensitivity to initial values, susceptibility to falling into local optimums, and being only able to obtain a single solution) and the challenges of intelligent optimization algorithms (complex parameter tuning, slow convergence and low accuracy). Our method has a large accuracy improvement over the comparison method in basically all test cases, and its effciency can also meet the needs of real geometric constraint solving scenarios. This research provides new insights into the design of geometric constraint solving algorithms, offers a fresh perspective on improving the performance and generality of solvers, and contributes to technological advances in the CAD field.

Keywords: genetic algorithm, Powell's algorithm, geometric constraint solving, similarity calculation.

1. Introduction

Computer-aided design (CAD) is the most widely used modelling approach for engineering design. The typical starting point in these designs is 2D sketches which can later be extruded and combined to obtain complex three-dimensional assemblies [17]. The geometric constraint solver is a key component in this process, which is usually a key technology in parametric CAD design, and the geometric constraint solver algorithm is the core of the geometric constraint solver. The accuracy and efficiency of the geometric constraint solver algorithm is related to the quality of the final product and the experience of the designer, and the improvement of the geometric constraint solver algorithm has always been an important research topic.

Samy Ait-Aoudia et al [1] classified geometric constraint solving techniques into three categories: algebraic, rule-oriented and graph construction, where algebraic methods include numerical and symbolic methods. Numerical methods are widely used because they are fast and applicable to almost all systems, but numerical methods have their inherent defects, the most representative of which is the Newton-Raphson iterative method, which

is widely used in solving geometric constraints based on numerical methods [10,12,16]. In these studies, the numerical methods converge faster but rely heavily on proper initial value guessing, otherwise the algorithm will not converge or converge to a local optimal solution, and only one root can be found by such methods. Therefore, numerical algorithms can be used directly when dragging the geometry, when the solution is close to the desired result.

In recent years, many scholars have tried to introduce intelligent optimisation algorithms such as particle swarm algorithm, simulated annealing algorithm and genetic algorithm to solve the shortcomings of traditional numerical methods [5,23,6]. In these studies, the intelligent optimization algorithm is relatively more relaxed on the initial value requirements, with global convergence and can find multiple solutions to the problem, but the algorithm's parameter setting is complex, the convergence speed is slower, and the accuracy of the results obtained by the algorithm is poor, which can not meet the requirements of practical engineering use.

In this paper, a hybrid genetic-Powell algorithm (GA-Powell algorithm) combining the advantages of numerical method and intelligent optimization algorithms proposed based on the practical application scenario of chemical static equipment design. The algorithm first uses the global search ability of the genetic algorithm to give rough initial solutions, and then screens the initial solutions by similarity calculation to remove similar initial solutions that are easy to converge to the same exact solution, and then inputs the screened initial solutions into Powell algorithm for exact solution in order to obtain all the solutions that conform to the design accuracy. Compared with the traditional numerical algorithm, GA-Powell algorithm is slower, so it is not suitable to be used directly when the user drags the geometry. Generally, it is used when the geometry is generated according to commands or scripts. The proposed GA-Powell algorithm possesses the following contributions:

- Taking advantage of the genetic algorithm, it makes up for the inherent defects of Powell method that the initial value is sensitive, easy to fall into the local optimal solution, and can only obtain one solution.
- Using the advantages of the Powell method, it makes up for the inherent defects of the genetic algorithm that the parameter setting is complicated, the convergence speed is slow and the accuracy is poor.

The high efficiency and accuracy of the GA-Powell algorithm meet the needs of practical geometric constraint solving scenarios, and the design ideas of the algorithm can help guide the research of geometric constraint solving algorithms, which is conducive to improving the performance and applicability of geometric constraint solvers. In the following sections, we present our research results and analyses.

Section 2 provides an overview of existing geometric constraint solving methods, especially the research based on numerical methods with intelligent optimisation algorithms.

Section 3 presents the research methodology, including how geometric constraint solving problems can be transformed into the optimisation problems, an overview of genetic algorithms and Powell's method, and the design and implementation of a hybrid GA-Powell algorithm.

Section 4 describes our experiments, including the experimental setup, systems of equations for testing, and comparative analyses of number of solutions, solution accuracy and solving time.

In Section 5, we will give some principles of parameter setting based on the analysis of the experimental results in order to make the GA-Powell algorithm perform better in different problem scenarios.

Finally, in Section 6, we summarise our main findings, discuss the contributions and implications of the research, acknowledge the limitations and provide recommendations for future research.

2. Related Work

According to the classification proposed by Samy Ait-Aoudia et al [1], geometric constraint solving methods can be broadly classified into three categories: algebraic methods, rule-oriented methods and graph construction methods. In turn, algebraic methods include numerical and symbolic methods as shown in Fig. 1.

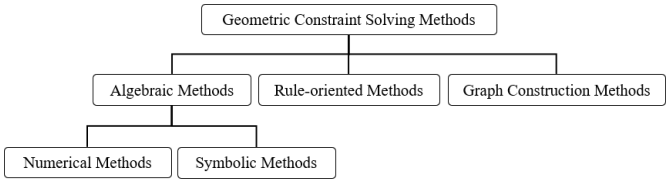

Fig. 1. Classification of geometric constraint solving methods

Numerical methods convert a geometrically constrained system into a set of nonlinear equations that are solved using numerical algorithms. This method, originally introduced by Hillyard [7] at Cambridge University and further developed and improved by Gossard [11] at MIT, is called variational geometry. Although these equations are usually multi-solution, numerical methods usually find only one solution.Borning [2], Hillyard and Braid [8] and Sutherland [21] use relaxation methods. This method interferes with the values assigned to the variables and minimises the measure of global error. Typically, convergence to a solution is slow.

The most widely used method is the Newton-Raphson iteration. It is used in the geometric constraint solving described in Refs [10,12,16]. Newton-Raphson is a local method that converges much faster than relaxation. However, this class of numerical methods requires computing the gradient of the problem to iteratively update the values, and the method is not applicable to systems of equations that are always overconstrained unless special steps are taken, such as solving a least squares problem. It works well if a good approximation of the expected solution is provided and the system is not pathological. Thus, if the starting point is taken from the user's sketch, then the sketch should approximate the expected solution. Numerical methods are generalized and are used as a last resort when other methods do not work well.

Powell's method is a direct search method, which makes full use of the objective function's sex state and constructs the conjugate direction by using the change of the objective function value, and the optimization search method is based on the conjugate direction, which makes it unnecessary to compute the gradient of the objective problem, which greatly saves the amount of computation, and therefore the method has a high convergence efficiency, and it is widely used in the field of optimization [4,9,13]. However, Powell's method still has the disadvantage that traditional numerical methods are sensitive to the initial value and easy to fall into local optimization.

The symbolic method, like the numerical method, transforms a system of geometric constraints into a set of nonlinear equations. However, it uses symbolic algebra to solve these constraints, using methods such as the Grobnacki method [3] or the Wu-Ritt characteristic column method [22]. If symbolic parameters are used in a system of nonlinear equations, the symbolic method can find generalised solutions for geometrically constrained systems, making it a very efficient method. However, the disadvantages of this method are its slowness and its high time and space complexity, and therefore the limitations on the types of geometric elements and constraints that can be used.

The rule-oriented geometric constraint solving method utilizes rules to define and carry out the construction process, hence the name "rule-construction solving method". This approach allows for clear representation of geometric knowledge, separating it from the processing stage, and makes it easy to expand the rule base. However, this method also has its drawbacks: the rules are often incomplete, the system is bulky, the solving speed is slow, and it is unable to solve cyclic constraints.

The graph construction method converts the geometric constraint system into graphs, deduces the construction process by analysing the geometric constraint graphs, and generates geometric shapes based on the construction steps. Currently, several researchers use machine learning training datasets (e.g., SketchGraphs [20]) and frameworks (e.g., SketchGen [17]) to train models to automatically generate sketches, thereby reducing design time and enabling new design workflows. The method is based on graph theory and is theoretically rigorous, fast and efficient. However, it can only solve closed-loop constraints by numerical methods. In addition, it is sensitive to the type of geometric elements and constraints used, thus requiring modification of the solution algorithm when new geometric elements or constraints are added, making this method less versatile.

The summary of geometric constraint solving methods is shown in Table 1. Numerical methods are widely used in practical geometric constraint solving due to their universal applicability, and improving their solving efficiency and stability is the focus of research in this field.

Intelligent optimisation algorithms have entered the scholarly scene in recent years. In this approach the problem is reinterpreted as an optimisation problem and solved using genetic, particle swarm or other evolutionary methods such as [5,23,6]. It is stable and less sensitive to initial value guessing, but its solution efficiency is low compared to traditional numerical methods, which cannot meet the real-time demands of design work, and its solution accuracy is relatively poor, problems that are illustrated in Ref. [24,25].

By comparing the characteristics of traditional numerical methods and intelligent optimisation algorithms, it can be found that the two can complement each other by combining their strengths - the global convergence of intelligent optimisation algorithms can make up for the shortcomings of traditional numerical methods that are sensitive to the

Table 1. Summary of geometric constraint solving methods

Methods		Advantages	Applicable scene
Algebraic	Numerical	Fast, general	Most systems
	Symbolic	Effective, generic solution	Small systems with restricted elements and constraints
Rule-oriented		Separate knowledge and processing, avoid numerical instability	Small systems
Graph Construction		Rigorous theory, fast solving	Systems without new types of elements and constraints

initial value, prone to falling into local optimal solutions and only able to obtain a single solution, whereas the accuracy and high efficiency of traditional numerical algorithms can make up for the slow convergence speed and accuracy of intelligent optimisation algorithms. optimisation algorithm's slow convergence speed and poor accuracy.

Therefore, in this paper, the classical algorithm genetic algorithm in intelligent optimization algorithm and the classical algorithm Powell algorithm in traditional numerical algorithm are fused to take advantage of the powerful global searching ability of the genetic algorithm and the fast local convergence ability of the Powell method which does not need to derive the objective function, in order to obtain a high-performance algorithm for geometric constraint solving.

3. Methodology

3.1. Nonlinear equation systems and optimization problems

Often, designers sketch graphics can be based on the principles of analytic geometry expressed in the form of a system of equations, the parameters and expressions of the three basic geometric elements of the point, the line and the circle, as shown in Table 2, and the problem of solving the system of equations can be transformed into the problem of finding the extreme points of the function.

Table 2. Parameters and expressions for the three basic geometric elements: point, line and circle

Geometric	Representation	Parameters	Equation
Point	P_0	$P_0(x_0, y_0)$	$x = x_0, y = y_0$
line	$L(P_1, P_2)$	$P_1(x_1, y_1), P_2(x_2, y_2)$	$\frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1}$
Circle	C_0	$C_0(x_0, y_0, r)$	$(x - x_0)^2 + (y - y_0)^2 = r^2$

A system of nonlinear equations consists of n equations involving n unknown quantities of the form

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0\,, \\ f_2(x_1, x_2, \cdots, x_n) = 0\,, \\ \cdots, \\ f_m(x_1, x_2, \cdots, x_n) = 0\,, \end{cases} \tag{1}$$

where $f_i(\boldsymbol{x}) = 0 (i = 1, 2, \cdots, m)$ is a system of nonlinear equations, $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is the unknown vector of the system of equations.

The use of numerical methods and intelligent optimisation algorithms in solving systems of non-linear equations requires the conversion of a system of non-linear equations problem into an optimisation problem. Converting a system of nonlinear equations into a single objective optimisation problem is expressed in the form of

$$\min F(\boldsymbol{x}) = \sum\nolimits_{i=1}^{m} |f_i(x_1, x_2, \cdots, x_n)|\,, \tag{2}$$

or

$$\min F(\boldsymbol{x}) = \sum\nolimits_{i=1}^{m} f_i^2(x_1, x_2, \cdots, x_n)\,. \tag{3}$$

Then the problem of solving the system of nonlinear equations is transformed into the problem of finding the minimum value of the fitness function $F(\boldsymbol{x})$. In order to test the ability of the GA-Powell hybrid algorithm to solve high power complex problems, in this paper, the method of Eq. 3 is chosen to transform the system of equations into an optimisation problem for solving the system of nonlinear equations.

3.2. Overview of genetic algorithm and Powell's method

Genetic Algorithm (GA) is a prominent algorithm among meta-heuristics that draws inspiration from biological evolutionary processes [15]. GA emulates the Darwinian theory of survival of the fittest in the natural world. J.H. Holland first proposed GA in 1992.

The fundamental elements of GA comprise chromosome representations, fitness selections, and biologically inspired operators. Additionally, Holland introduced a novel element called inversion, which is widely used in GA implementations [19]. Normally, chromosomes are formatted using binary strings. In a chromosome, each locus (a specific position on the chromosome) has two possible allele forms: 0 and 1. Chromosomes are regarded as points in the solution space. These chromosomes undergo processing using genetic operators, by iteratively cumulating their populations. The fitness function applies in assigning a value to every chromosome within the population [15].

Operators inspired by biology include selection, mutation, and crossover. During selection, a chromosome is selected for further processing based on its fitness value. In the crossover operator, a random locus is selected, and subsequences between chromosomes are altered to create offspring. During mutation, certain random bits of chromosomes are flipped according to the probability distribution [19,26,15]. Table 3 displays the relationship between biological evolution and genetic algorithm. And the general steps of the genetic algorithm are shown in Fig. 2.

Genetic algorithm searches for diversity in the solution space and is more capable of finding the global optimal solution compared to other intelligent optimization algorithms without easily falling into the local optimal solution, which makes it perform well

Table 3. Correspondence between biological evolution and genetic algorithm

Biological Evolution	Genetic Algorithm
Individual	Feasible solution to a problem
Population	A group of feasible solutions
Chromosome	Encoding corresponding to a feasible solution
Gene	Element of the encoding
Environment	Adaptive function
Survival of the Fittest	The more the adaptive value approximates the optimal value, the more likely a feasible solution is to be retained.
Selection	The process of passing genetic information from parent individuals to offspring individuals.
Crossover	The process of generating offspring individuals from parent individuals through crossover.
Mutation	Changing a portion of the chromosome encoding.

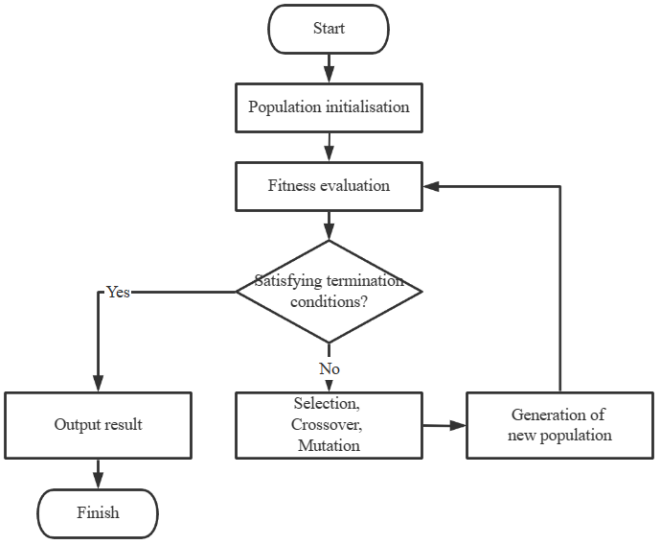

Fig. 2. Classification of geometric constraint solving methods

in complex multi-peaked functions and high-dimensional spaces. Moreover, the genetic algorithm is adaptive in the sense that it can automatically adjust to the complexity and characteristics of the problem. This makes it applicable to a wide range of problem types without the need to manually fine-tune the algorithm parameters. Meanwhile, the genetic algorithm is easy to parallelize, and multiple individuals can be evaluated and evolved at the same time, which improves the search efficiency, which facilitates the improvement of the algorithm's efficiency from the parallelization perspective in the future. Combining the above advantages, we chose the genetic algorithm as part of the hybrid algorithm.

Powell's method, also known as direction acceleration method, which was proposed by Powell [18] in 1964, is a search method formed by using the property that conjugate directions can accelerate the convergence speed. The steps of Powell's algorithm are shown in Algorithm 1. Powell's method can be used to solve general unconstrained optimization problems, and for the optimization problem of objective function with dimension $n < 20$, this method can obtain more satisfactory results. Unlike other direct methods, Powell's method has a complete theoretical system, so its computational efficiency is higher than other direct methods. The method does not require derivation of the objective function and can be applied when the derivative of the objective function is discontinuous, therefore, Powell's algorithm is a very effective direct search method, which we choose for fast convergence.

Algorithm 1: Powell's Method

Choose an initial point x_0 and set $p_i = e_i$, for $i = 1, 2, ..., n$;
Compute x_1 as the minimizer of f along the line $x_0 + \alpha p_n$;
Set $k \leftarrow 1$;
while *convergence test is not satisfied* **do**
    Set $z_1 \leftarrow x_k$;
    for $j = 1, 2, \ldots, n$ **do**
        Calculate α_j so that $f(z_j + \alpha_j p_j)$ is minimized;
        Set $z_{j+1} \leftarrow z_j + \alpha_j p_j$;
    end
    for $j = 1, 2, \ldots, n-1$ **do**
        Set $p_j \leftarrow p_{j+1}$;
    end
    Set $p_n \leftarrow z_{n+1} - z_1$;
    Calculate α_n so that $f(z_{n+1} + \alpha_n p_n)$ is minimized;
    Set $x_{k+1} \leftarrow z_{n+1} + \alpha_n p_n$;
    Set $k \leftarrow k + 1$;
end

3.3. Genetic-Powell hybrid algorithm

While the crossover and variational operators in genetic algorithms can look for solutions to equations in the complete space of variables, Powell's method does so rapidly and methodically in the area around the convergence point with a high level of local convergence.

To leverage the benefits of both approaches, this paper proposes the hybrid genetic-Powell algorithm. The algorithm's progression is explained as Algorithm 2.

Algorithm 2: GA-Powell Algorithm

Input: Optimized function $F(X)$ representing the equation system, number of genetic algorithm calls N, Euclidean distance threshold O_t
Output: Solution set of the equation system X^*

```
Initialize the initial solution set X_0;
for i from 1 to N do
    Obtain an initial solution x_i using the genetic algorithm;
    Add x_i to the set X_0;
end
for each initial solution x_i in set X_0 do
    for each other element x_j in set X_0 except x_i do
        if Euclidean distance between x_i and x_j is less than O_t then
            Remove x_j from set X_0;
        end
    end
end
Initialize the precise solution set X*;
for each initial solution x_i in set X_0 do
    Apply Powell's algorithm to x_i as the initial value and obtain the precise solution x*;
    Add x* to set X*;
end
Output Set X* containing precise solutions;
```

Genetic algorithm has strong global search ability, Powell locally and carefully searches for solutions near the convergence point and has high local convergence. The termination condition of the genetic algorithm is the set number of genetic generations. In order to play the global search ability of genetic algorithm, in this paper, when using genetic algorithm to obtain a set of initial solutions, it is not necessary to set the genetic generations to be very large in order to avoid missing some feasible solutions; at the same time, appropriately increase the number of individuals in the population and the mutation rate, in order to give full play to the global search ability of genetic algorithm.

After cyclically calling the genetic algorithm to obtain the initial solution, the algorithm calculates the similarity between the initial solutions to exclude the invalid solutions, i.e., to exclude multiple initial solutions that will converge to the same optimal solution, in order to greatly reduce the useless calculations and improve the efficiency of the algorithm. Euclidean distance is the simplest and easiest to understand method in all similarity calculations, and Euclidean distance is more suitable for data with obvious geometric structure in geometrically constrained solution scenarios, so we use Euclidean distance to determine the similarity of the solutions, and if the distance is greater than the set threshold then it is judged to be dissimilar, and the dissimilar solutions are retained.

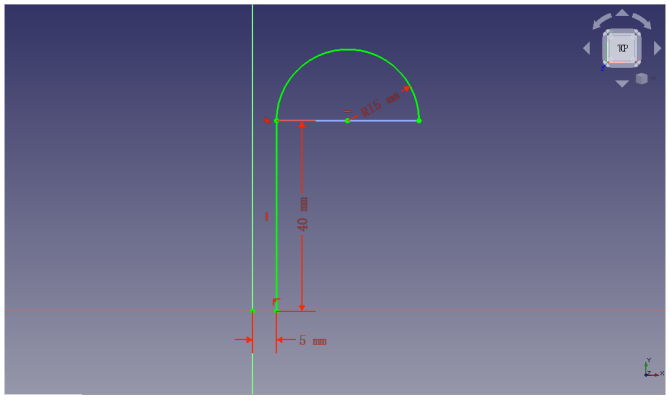

Fig. 3. The arc is tangent to the right of the line segment.

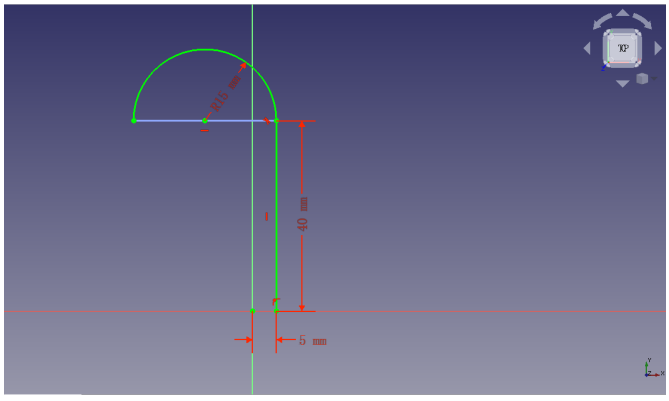

Fig. 4. The arc is tangent to the left of the line segment.

4. Experiment

In this paper, the application scenario of GA-Powell algorithm is static equipment design in chemical industry, which mainly involves geometric elements such as arcs and lines. The Fig. 3 and Fig. 4 show a simplified design example: if the head of the tower is tangent to the upper side of one barrel, there are two solutions as shown in the figure (regardless of the concave head into the barrel). Let the lower endpoint of the line segment be (x_1, y_1), the upper endpoint be (x_2, y_2), the center of the arc be (x_c, y_c), and the radius be R, then the constraint equation of this example is:

$$\begin{cases} \sqrt{(x_1-0)^2+(y_1-0)^2}=5, \\ y_1=0, \\ x_2=x_1, \\ \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}=40, \\ R=15, \\ y_c=y_2, \\ \sqrt{(x_c-x_2)^2+(y_c-y_2)^2}=R. \end{cases} \tag{4}$$

Regardless of the optimization of the actual solver, the equations are input into GA-Powell algorithm and two optimal solutions are obtained. Optimal solution 1 is (5.000000000000956, 0.00000000000074145893, 5.000000000001288, 39.999999999 9704, 14.999999999998659, -9.999999999995033, 39.99999999999881). Optimal solution 2 is (4.999999999997651, -0.00000000000083986589, 4.999999999998072, 39.9 99999999999254, 15.000000000000888, 19.999999999997126, 39.99999999999913). The errors of the solutions are all within 10^{-20}, and the solution time is 0.245536 seconds.

In order to better test the performance of the GA-Powell algorithm for solving multi-root nonlinear systems of equations, for the convenience of comparison and without loss of generality, we use 9 selected systems of equations from reference [14] as the test systems of equations and compare them with the PGWO method proposed in that literature. And in order to prove that the GA-Powell algorithm is insensitive to the initial value, the initial value intervals of all the test equation sets except Eq. 8 are set to [-10, 10]. The simulation experiments were implemented in python on a PC with a CPU of AMD Ryzen 7 6800H at 3.20 GHz and 16.0 GB of RAM. Some of the key quantitative results are selected below to show the effect of the algorithm.

The experiment 1 is to solve

$$\begin{cases} f_1(x)=x_1^2+x_2^2-2=0\,, \\ f_2(x)=x_1^2+x_2^2/4-1=0\,, \end{cases} \tag{5}$$

the equation has four solutions, which are $(-0.8165,-1.1547)^T, (0.8165,-1.1547)^T, (-0.8165,1.1547)^T, (0.8165,1.1547)^T$.

and

$$\begin{cases} f_1(x)=x_1^4+4x_2^4-6=0\,, \\ f_2(x)=x_1^2x_2-0.6787=0\,, \end{cases} \tag{6}$$

the equation has four solutions, which are $(-1.56353, 0.27763)^T, (1.56353, 0.27763)^T, (-0.78971, 1.08830)^T, (0.78971, 1.08830)^T$.

The parameters are set as follows: the number of cycles of the genetic algorithm N = 10, the population is 100, the maximum number of iterations of the genetic algorithm is 50, the mutation rate is 0.03, and Powell's method has a maximum number of iterations of 500 and a convergence accuracy of 10^{-20}. The results of the experiment are shown in Table 4.

Table 4. Experiment 1 results

Equation	GA-Powell	Error
Eq. 5	-0.816496580927726 1.1547005383792515	0.0
	-0.8164965809277257 -1.154700538379252	1.9721522630525295e-31
	0.8164965809277259 1.1547005383792575	1.8858706015439813e-28
	0.8164965809277255 -1.1547005383792457	2.197470659106281e-28
Eq. 6	-1.5635325915863936 0.27762845243830503	4.078467204661264e-23
	-0.7897063597753254 1.0882948602064995	4.2762300902543955e-24
	0.7897063597753038 1.0882948602066016	2.9496002284279395e-29
	1.5635325915863927 0.2776284524446274	8.496843844873951e-23

As can be seen in Table 4, the GA-Powell algorithm is able to find all solutions and has higher accuracy compared to the PGWO method solutions. However, the accuracy of the solution of eq. 6 with higher powers decreases compared to eq. 5. The solution took 0.321683 seconds and 0.317919 seconds, respectively.

Experiment 2 solves the Eq. 7, Eq. 8 and Eq. 9. The parameters are set as follows: the number of cycles of the genetic algorithm N = 50, the population is 100, the maximum number of iterations of the genetic algorithm is 50, the mutation rate is 0.03, and Powell's method has a maximum number of iterations of 500 and a convergence accuracy of 10^{-20}. The results are shown in Table 5, Table 6 and Table 7.

$$\begin{cases} \cos(2x_1) - \cos(2x_2) - 0.4 = 0\,, \\ 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 1.2 = 0\,, \end{cases} \tag{7}$$

the equation has 13 solutions, which are
$(-9.268258, -8.931402)^T, (-8.744542, -7.164787)^T, (-6.126665, -5.789809)^T,$
$(-5.602950, -4.023195)^T, (-2.985073, -2.648216)^T, (-2.461357, -0.881602)^T,$
$(0.156520, 0.493376)^T, (0.680236, 2.259991)^T, (3.298113, 3.634969)^T,$
$(3.821828, 5.401583)^T, (6.439705, 6.776562)^T, (6.963421, 8.543176)^T,$
$(9.581298, 9.918154)^T$.

Table 5. Experiment 2 results of Eq. 7

Equation	No.	GA-Powell	Error
Eq. 7	1	9.581298030452515 9.918154334992423	5.068086189398967e-25
	2	-6.12666523749644 -5.7898089329563325	2.3517915736901414e-28
	3	6.439705376862722 6.776561681402831	2.7240353133413064e-30
	4	-2.985072583906657 -2.648216279366548	5.053640174072107e-31
	5	-5.602949507250662 -4.023194565762204	2.6262773526874095e-24
	6	3.8218284535187137 5.401583395006905	8.69749223328177e-25
	7	-2.4613568536608685 -0.88160191217265	1.025081519222289e-24
	8	-9.268257891086272 -8.931401586543789	7.002171694601335e-23
	9	0.6802357999289202 2.259990741416713	6.570829235990062e-27
	10	-8.74454216084046 -7.164787219352157	1.474125009516472e-24
	11	6.963421107108508 8.543176048596612	5.089100457837923e-25
	12	3.2981127232729306 3.634969027817465	2.430470957633393e-22
	13	0.15652006968312956 0.4933763742281069	2.9416965098544255e-22

As can be seen from Table 5, the GA-Powell algorithm is still able to find all exact solutions for problems with a large number of roots. The solution took 1.758389 seconds.

$$\begin{cases} \sin({x_1}^3) - 3x_1x_2^2 - 1 = 0\,, \\ \cos(3{x_1}^2x_2) - \left|x_2^3\right| + 1 = 0\,, \end{cases} \tag{8}$$

the equation has 10 solutions, which are
$(-1.810885, -0.3490924)^T, (-1.810885, 0.3490924)^T, (-1.502221, -0.409077)^T,$
$(-1.502221, 0.409077)^T, (-1.791302, 0.301926)^T, (-1.791302, -0.301926)^T,$
$(-0.947268, 0.785020)^T, (-0.947268, -0.785020)^T, (-0.213057, 1.256845)^T,$
$(-0.213057, -1.256845)^T$.

Since Eq. 8 had too many solutions in the interval $[-10, 10]$, the interval was narrowed down to $[-2, 2]$. The GA-Powell algorithm did a better job of solving the problem, obtaining all the exact solutions. The solution took 1.923495 seconds.

Table 6. Experiment 3 results of Eq. 8

Equation	No.	GA-Powell	Error
Eq. 8	1	-0.9472681469862627 0.7850200155682888	4.930380657631324e-32
	2	-1.5022159860694961 0.40907656829415534	4.601031229701551e-28
	3	-0.2130566192382045 1.256845317434239	3.059160711793778e-22
	4	-1.5022159860694986 -0.4090765682941479	3.5745259767827097e-31
	5	-0.947268146986263 -0.7850200155682433	7.643742929444023e-26
	6	-1.810885199439986 -0.3490909919763665	3.5745259767827097e-31
	7	-1.791302084615447 -0.3019263417131378	4.930380657631324e-32
	8	-1.7913020846154477 0.3019263417147986	4.6459929979441084e-23
	9	-0.2130566192381985 -1.2568453174310898	2.7295323426479193e-24
	10	-1.810885199439986 0.34909099197636656	3.5745259767827097e-31

$$\begin{cases} -3.84{x_1}^2 + 3.84x_1 - x_2 = 0\,, \\ -3.84{x_2}^2 + 3.84x_2 - x_3 = 0\,, \\ -3.84{x_3}^2 + 3.84x_3 - x_1 = 0\,, \end{cases} \tag{9}$$

the equation has 8 solutions, which are
$(0.0000, 0.0000, 0.0000)^T$, $(0.4881, 0.9594, 0.1495)^T$, $(0.5403, 0.9538, 0.1694)^T$, $(0.9594, 0.1494, 0.4879)^T$, $(0.1494, 0.4881, 0.9594)^T$, $(0.9538, 0.1693, 0.5402)^T$, $(0.1693, 0.5399, 0.9538)^T$, $(0.7396, 0.7396, 0.7396)^T$.

Table 7. Experiment 3 results of Eq. 9

Equation	No.	GA-Powell	Error
Eq. 9	1	0.5403878416288977 0.9537362774344668 0.16943381967326443	2.465190328815662e-32
	2	0.7395833333333289 0.7395833333333411 0.739583333333685	5.465303046638133e-25
	3	0.16943381967327076 0.5403878416289855 0.9537362774344623	5.724788241092171e-27
	4	0.9537362774344795 0.16943381967316742 0.5403878416283857	9.46563760490936e-26
	5	0.14940689655344594 0.48800438713233574 0.9594474442442135	6.407722999371885e-29
	6	0.48800438713234184 0.959447444244211 0.14940689655344785	9.013968437314468e-29
	7	0.9594474442438022 0.14940689655528064 0.4880043871388201	3.51920906640077e-24
	8	-0.00000000000130681231 0.00000000000393663907 -0.00000000000165209884	3.8675468719474286e-22

As can be seen from Table 7, the GA-Powell algorithm still solves well when the number of variables increases. The solution took 1.951163 seconds.

From the above experiments, it can be seen that the GA-Powell algorithm proposed in this paper is able to find all the solutions of the above nine systems of equations with high solution accuracy, fast solution speed and insensitivity to the setting of the initial value.

5. Discussion

From the experiments in Section 4, it can be seen that the GA-Powell algorithm proposed in this paper indeed balances the advantages of the genetic algorithm and the Powell algorithm, and is able to accurately solve all the solutions of the problem with a large range of initial values, which is more effective compared to the PGWO method in reference [14]. The proposed algorithm theoretically gives an effective idea for solving multi-solution problems in the geometric constraint solving domain. This section mainly discusses the principle of parameterization of GA-Powell algorithm when targeting different types of systems of equations based on experimental results.

5.1. Systems of equations with a greater number of roots

The problems in Experiment 2 have a large number of solutions, and this kind of equations mainly test the global search ability of the genetic algorithm. When setting the parameters of the genetic algorithm, we should focus on enhancing its global search ability, so we need to increase the population size to increase the breadth of the search space, increase the variation rate to jump out of the local optimum, and also increase the number of times to call the genetic algorithm to obtain the initial value. The above operations can be used to make the genetic algorithm give initial values for all solutions for Powell's algorithm to converge to the exact solution.

5.2. Systems of equations with densely distributed roots

The roots of Eq. 9 are distributed in a small range of intervals, and the Euclidean distance between the solutions of such systems of equations is small, i.e., the similarity between the solutions is high, so when the genetic algorithm outputs the initial solutions and then screens them, it is necessary to reduce the Euclidean distance threshold for determining the similarity of the two values so as not to exclude the valid initial solutions.

5.3. Systems of equations with higher powers

Eq. 6 has higher powers of the variables, and these systems of equations are more complex and computationally intensive, requiring an increase in the number of iterations and termination accuracy of Powell's algorithm in order to obtain a highly accurate solution.

It can also be seen from the above discussion that the GA-Powell algorithm has limitations in terms of the parameter tuning problem, and the parameter adaptivity needs to be enhanced in the future; at the same time, the number of test cases used in this paper is not large enough, and there is a need to test the effectiveness of the algorithm by using more test cases and applying the algorithm to a real geometric constraint solver.

6. Conclusion

The conclusion of this paper highlights the research used to address the lack of support for solving multi-solution problems by geometric constraint solvers based on traditional numerical algorithms, which focuses on genetic algorithms and Powell algorithms, and the proposed hybrid GA-Powell algorithm combines the advantages and compensates for the shortcomings of the two. The results are compared and analysed in terms of the number of solutions, solution accuracy and solution time. The research successfully enhances the support of geometric constraint solving algorithm for solving multi-solution problems and obtains the following main research results:

- To address the shortcomings of the most prevalent geometric constraint solving algorithms based on traditional numerical algorithms investigate the selection of suitable algorithms in order to hybridize them in order to create complementary effects.
- The proposed GA-Powell hybrid algorithm has the advantages of high accuracy, high efficiency and high robustness in solving multi-solution problems.
- The experience of the GA-Powell hybrid algorithm is given to guide the parameter selection in different scenarios.

In this study, the method of mixing genetic algorithm with Powell algorithm to enhance the performance of geometric constraint solving algorithm is proposed, which provides a new idea to improve the solving of geometric constraints. The effectiveness of the GA-Powell algorithm is verified by conducting scientific analog simulation tests, which provides an important method for CAD systems to solve geometric constraint problems with multiple solutions. The results of the research are of great significance in guiding the further development and improvement of geometrically constrained solvers in the field of CAD, providing practical experience and suggestions for related work.

Future work in this research includes enhancing the parameter adaptivity of the algorithm, using more test cases, and applying the algorithm to a real geometric constraint solver through user experience in order to test the effectiveness of the algorithm.

References

1. Ait-Aoudia, S., Bahriz, M., Salhi, L.: 2d geometric constraint solving: An overview. In: 2009 Second International Conference in Visualisation. pp. 201–206. IEEE (2009)
2. Borning, A.: The programming language aspects of thinglab, a constraint-oriented simulation laboratory. ACM Transactions on Programming Languages and Systems (TOPLAS) 3(4), 353–387 (1981)
3. Bose, N., Bose, N.: Gröbner bases: An algorithmic method in polynomial ideal theory. Springer (1995)
4. Chen, R., Chu, J., Zhang, B.: A coincidental calculation method of some terminal-guided projectile inertial navigation segment ballistic coeffcients based on powell method. Fire Control & Command Control 45(8), 176–180 (2020)
5. Chunhong, C., Bin, Z., Limin, W., Wenhui, L.: The parametric design based on organizational evolutionary algorithm. In: PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence Guilin, China, August 7-11, 2006 Proceedings 9. pp. 940–944. Springer (2006)

6. Gao, X., Sun, L., Sun, D., et al.: Artificial immune-chaos hybrid algorithm for geometric constraint solving. Inf. Technology J pp. 360–365 (2009)
7. Hillyard, R., Braid, I.: Analysis of dimensions and tolerances in computer-aided mechanical design. Computer-Aided Design 10(3), 161–166 (1978)
8. Hillyard, R., Braid, I.: Characterizing non-ideal shapes in terms of dimensions and tolerances. In: Proceedings of the 5th annual conference on Computer graphics and interactive techniques. pp. 234–238 (1978)
9. Li, C., Li, G., Tan, Y., Xu, X.: Medical image registration algorithm based on powell algorithm and improved genetic algorithm. Journal of Computer Applications 33(3), 640–644 (2013)
10. Light, R., Gossard, D.: Modification of geometric models through variational geometry. Computer-Aided Design 14(4), 209–214 (1982)
11. Light, R.A., Gossard, D.C.: Variational geometry: a new method for modifying part geometry for finite element analysis. Computers & Structures 17(5-6), 903–909 (1983)
12. Lin, V.C., Gossard, D.C., Light, R.A.: Variational geometry in computer-aided design. ACM SIGGRAPH Computer Graphics 15(3), 171–177 (1981)
13. Liu, X., Yin, X., Li, C.: Improvements of powell mechanical optimization. Journal of Machine Design 36(6), 80–86 (2019)
14. Lu, M., Qu, L., He, D.: Pathfinder grey wolf algorithm for solving multiple-roots nonlinear equations. Chinese Journal of Engineering Mathematics 39(6), 957–968 (2022)
15. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary computation 4(1), 1–32 (1996)
16. Nelson, G.: Juno, a constraint-based graphics system. In: Proceedings of the 12th annual conference on Computer Graphics and Interactive Techniques. pp. 235–243 (1985)
17. Para, W., Bhat, S., Guerrero, P., Kelly, T., Mitra, N., Guibas, L.J., Wonka, P.: Sketchgen: Generating constrained cad sketches. Advances in Neural Information Processing Systems 34, 5077–5088 (2021)
18. Powell, M.J.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The computer journal 7(2), 155–162 (1964)
19. Sampson, J.R.: Adaptation in natural and artificial systems (john h. holland) (1976)
20. Seff, A., Ovadia, Y., Zhou, W., Adams, R.P.: Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. arXiv preprint arXiv:2007.08506 (2020)
21. Sutherland, I.E.: Sketchpad: A man-machine graphical communication system. In: Proceedings of the May 21-23, 1963, spring joint computer conference. pp. 329–346 (1963)
22. Wenjun, W.: Basic principles of mechanical theorem proving in elementary geometries. Selected Works Of Wen-Tsun Wu p. 195 (2008)
23. Yuan, H., Li, W., Yi, R., Zhao, K.: The tpso algorithm to solve geometric constraint problems. Journal of Computational Information Systems 2(4), 1311–1316 (2006)
24. Yuan, H., Chang, X.: Combining genetic algorithm with simlex method for geometric constraint solving. In: 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering. vol. 1, pp. 453–455. IEEE (2010)
25. Yuan, H., Yu, C.: Optimization genetic algorithm for geometric constraint solving. In: 2010 International Conference on Artificial Intelligence and Education (ICAIE). pp. 590–593. IEEE (2010)
26. Zbigniew, M.: Genetic algorithms+ data structures= evolution programs. Comput Stat pp. 372–373 (1996)

Sun Yunlei (Corresponding author: sunyunlei@upc.edu.cn) is a PhD, associate professor and master director, who graduated from Beijing University of Posts and Telecommunications with a doctorate degree in engineering in 2014. In recent years, he has been

engaged in the research of intelligent design, machine learning and other fields, presided over a number of special projects of basic scientific research operations of central universities, lateral projects of enterprises and institutions, and participated in the projects of National 863 and 973 projects, National Natural Science Foundation of China, and Shandong Provincial Natural Fund as an academic backbone.

Li Yucong received his B.S. degree in Computer Science and Technology from China University of Petroleum (East China) in 2021. Currently, he is a master's student at China University of Petroleum (East China). His main research interests are geometric constraint solving problems in computer-aided design.

Received: September 07, 2023; Accepted: January 05, 2024.