

Chapter 2

A Survey of Metaheuristics Methods for Bioinformatics Applications

Ahmed Fouad Ali and Aboul-Ella Hassanien

Abstract Over the past few decades, metaheuristics methods have been applied to a large variety of bioinformatic applications. There is a growing interest in applying metaheuristics methods in the analysis of gene sequence and microarray data. Therefore, this review is intended to give a survey of some of the metaheuristics methods to analysis biological data such as gene sequence analysis, molecular 3D structure prediction, microarray analysis and multiple sequence alignment. The survey is accompanied by the presentation of the main algorithms belonging to three single solution based metaheuristics and three population based methods. These are followed by different applications along with their merits for addressing some of the mentioned tasks.

2.1 Introduction

In the 1970s, metaheuristics have been emerged to combine basic heuristic methods in higher level frameworks to explore a search space in an efficient and an effective way. Metaheuristics have two classes, population based methods and single solution based method as shown in Fig. 2.1. The population based method includes but not restricted to ant colony optimization (ACO) [11], genetic algorithms (GA) [30], particle swarm optimization (PSO) [31], scatter search (SS) [22], etc., while the single solution based methods includes but not restricted to tabu search (TS) [21], simulated annealing (SA) [32], variable neighborhood search (VNS) [36, 37], iterated local search (ILS) [43]. The main key feature of designing any metaheuristics algorithm is its capability of performing wide diversification and deep intensifica-

A.F. Ali (✉)

Faculty of Computers and Information, Department of Computer Science,

Member of Scientific Research Group in Egypt, Suez Canal University, Ismailia, Egypt

e-mail: ahmed_fouad@ci.suez.edu.eg

A.-E. Hassanien

Faculty of Computers and Information, Chair of Scientific Research Group in Egypt,

Cairo University, Cairo, Egypt

© Springer International Publishing Switzerland 2016

A.-E. Hassanien et al. (eds.), *Applications of Intelligent Optimization*

in Biology and Medicine, Intelligent Systems Reference Library 96,

DOI 10.1007/978-3-319-21212-8_2

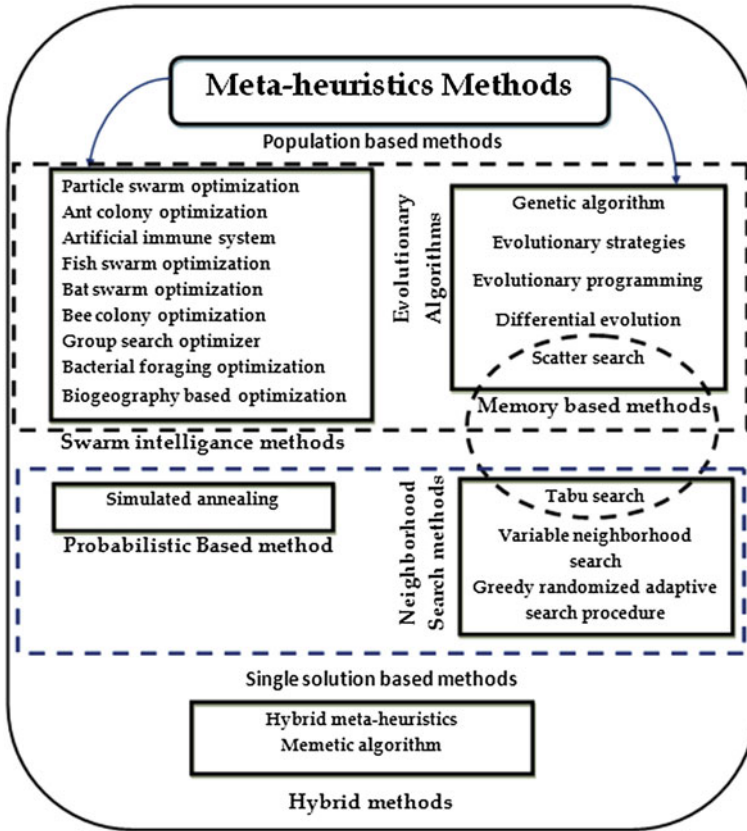


Fig. 2.1 Metaheuristics methods

tion. The term diversification generally refers to the exploration of the search space, whereas the term intensification refers to the exploitation of the accumulated search experience. There are several different single solution based methods, also called (trajectory methods), which can be seen as an extensions of local search algorithms. The goal of this kind of metaheuristic is to escape from local minima in order to proceed in the exploration of the search space and to move on to find better local minima such as TS, ILS, VNS, SA. We can find a different population based methods such as ACO and EC, they incorporate a learning component in the sense that they implicitly or explicitly try to learn correlations between decision variables to identify high quality areas in the search space. For instance, in evolutionary computation this is achieved by recombination of solutions and in ant colony optimization by sampling the search space at each iteration according to a probability distribution. Metaheuristics also classified into nature-inspired versus non nature-inspired metaheuristics, into memory-based versus memory-less methods, or into a dynamic or a static objective function methods. Metaheuristics have been applied to solve many

problems in different fields such as engineering, economics, management, biology, etc. In this work we describe some of the most important metaheuristics according to the single-point versus population-based search classification and how they have been applied in different bioinformatics applications. We survey the usage of metaheuristics methods with three different bioinformatics applications. The first application is a gene selection from gene expression data for cancer classification, we present the successful hybridization between PSO and TS methods. The second application is molecular 3D structure prediction by applying two proposed algorithms, the first algorithm is the group search optimizer (GSO) algorithm, the second proposed algorithm is a hybrid simulated annealing and variable neighborhood search algorithm. The two algorithms have been proposed to minimize the molecular potential energy function. Finally we present the role of genetic algorithm with the multiple sequence alignment application.

This paper is organized as follows. Section 2.2 presents the single solution based methods, and describes three algorithms of the main algorithms that belong to this methods, i.e. tabu search method (TS), simulated annealing method (SA) and variable neighborhood method (VNS). Section 2.3 presents the population based metaheuristics methods, and outlines three important population based methods, one of them belongs to the evolutionary algorithms such as genetic algorithm (GA), the other two methods belong to swarm intelligence methods such as particle swarm optimization (PSO) and group search optimizer (GSO) methods. In Sect. 2.4, we survey the role of the mentioned metaheuristics algorithms for solving the three different bioinformatics applications. Finally, Sect. 2.5 summarizes this paper by its main conclusions.

2.2 Single-Solution Based Metaheuristics Methods

In this section, we present single solution based metaheuristics (S-metaheuristics) methods, also called trajectory methods. They could be viewed as walks (moves) through neighborhoods in the search space of the problem [9]. S-metaheuristics methods are unlike P-metaheuristics methods, they iteratively apply the generation of the neighborhood solutions from the current single solution. This process iterates until a given stopping criteria e.g. (number of iterations). The most popular examples of such S-metaheuristics methods are tabu search (TS), simulated annealing (SA), iterated local search (ILS), guided local search (GLS) [46–48], variable neighborhood search (VNS), greedy randomized adaptive search procedure (GRASP) [14, 15]. The following subsections present a global overview of three methods of S-metaheuristics i.e. tabu search, simulated annealing and variable neighborhood and their principles.

2.2.1 Tabu Search

In 1986, Glover proposed a deterministic method called tabu search method (TS) in order to escape from local optima [21]. In 1990s, tabu search method becomes

very popular in solving optimization problems. The key feature of TS method is the use of memory, which records information related of the search process. TS generates a neighborhood solution from the current solution and accepts the best solution even if is not improving the current solution. This strategy may lead to cycles, i.e. the previous visited solutions could be selected again. In order to avoid cycles, TS discards the solution that have been previously visited by using memory which is called tabu list. The length of the memory (tabu list) control the search process. If the length of the tabu list is high the search will explore larger regions and forbids revisiting high number of solution. On the opposite, a low length of the tabu list concentrates the search on a small area of the search space. At each iteration the tabu list is updated (first in—first out queue). The tabu list contains a constant number of tabu moves called tabu tenure, which is the length of time for which a move is forbidden. If a move is good and can improve the search process but it is in tabu list, there is no need to be prohibited and the solution is accepted in a process called aspiration criteria. The main algorithm of tabu search method is reported in Algorithm 1. Good reviews of the TS method are provided in [18, 20]. TS have been applied to solve continuous optimization problems, see [5, 27].

Algorithm 1 Tabu search algorithm

Set $x = x_0$;	▷ Initial candidate solution
Set $length(L) = z$;	▷ Maximum tabu list length
Set $L = \{\}$;	▷ Initialize the tabu list
repeat	
Generate a random neighbor x' ;	
if $x' \notin L$ then	
if $length(L) > z$ then	
Remove oldest solution from L ;	▷ First in first out queue
Set $x' \in L$;	
end if	
end if	
if $x' < x$ then	
$x = x'$;	
end if	
until (Stopping criteria satisfied)	▷ e.g. Number of iterations
return x ;	▷ Best found solution

2.2.2 Simulated Annealing

Simulated annealing (SA) has been proposed by Kirkpatrick [32], SA is probably the most widely used meta-heuristic in combinatorial optimization problem. It was motivated by the analogy between the physical annealing of metals and the process of searching for the optimal solution in a combinatorial optimization problem. It is

inspired by the Metropolis algorithm [17]. The main objective of SA method is to escape from local optima and so to delay the convergence. The basic SA algorithm is described as shown in Algorithm 2. SA proceeds in several iterations from an initial solution x^0 . At each iteration, a random neighbor solution x' is generated. The neighbor solution that improves the cost function is always accepted. Otherwise, the neighbor solution is selected with a given probability that depends on the current temperature T and the amount of degradation ΔE of the objective function. ΔE represents the difference in the objective value between the current solution x and the generated neighboring solution x' . This probability follows, in general, the Boltzmann distribution as shown in Eq. 2.1.

$$P(\Delta E, T) = \exp\left(\frac{-f(x') - f(x)}{T}\right). \quad (2.1)$$

Many trial solutions are generated as an exploration process at a particular level of temperature. The temperature is updated until stopping criteria satisfied.

Algorithm 2 Simulated annealing algorithm

Set $x = x_0$;	▷ Generate the initial solution
Set $T = T_{max}$;	▷ Starting temperature
repeat	
repeat	▷ At a fixed temperature
Generate a random neighbor x' ;	
$\Delta E = f(x') - f(x)$;	
if $\Delta E \leq 0$ then	
$x = x'$;	▷ Accept the neighbor solution
else	
Accept x' with probability $e^{\frac{-\Delta E}{T}}$;	
$x = x'$;	
end if	
until (Equilibrium condition)	▷ e.g. number of iterations executed at each T
$T = g(T)$;	▷ Temperature update
until (Stopping criteria satisfied)	▷ e.g. $T < T_{min}$
return x ;	▷ Best found solution

In order to improve the performance of SA, we should carefully deal with the tuning of control parameters which included:

- **Choice of an initial temperature.** Choosing too high temperature will cost computation time expensively, while too low temperature will exclude the possibility of ascent steps, thus losing the global optimization feature of the method. We have to balance between these two extreme procedures.
- **Choice of the temperature reduction strategy.** If the temperature is decreased slowly, better solutions are obtained but with a more significant computation time. On the other side, a fast decrement rule causes increasing the probability of being trapped in a local minimum.

- **Equilibrium State.** In order to reach an equilibrium state at each temperature, a number of sufficient transitions (moves) must be applied. The number of iterations must be set according to the size of the problem instance and particularly proportional to the neighborhood size.
- **Stopping criterion.** Concerning the stopping condition, theory suggests a final temperature equal to 0. In practice, one can stop the search when the probability of accepting a move is negligible, or reaching a final temperature TF .

2.2.3 Variable Neighborhood Search

Variable neighborhood search (VNS) method has been proposed by Hansen and Mladenovic [37]. In VNS method, a set of predefined neighborhoods are explored to provide a better solution. VNS explores a set of neighborhoods to get different local optima and escape from local optima as shown in Fig. 2.2. The main steps of VNS algorithm are shown in Algorithm 3. In Algorithm 3, a set of neighborhood structure N_k are defined where $k = 1, 2, \dots, n$. At each iteration, an initial solution x is generated randomly. A random neighbor solution x' is generated in the current neighborhood N_k . The local search procedure is applied to the solution x' to generate the solution x'' . If the solution x'' is better than the x solution then the solution x'' becomes the new current solution and the search starts from the current solution. If the solution x'' is not better than x solution, the search moves to the next neighborhood N_{k+1} , generates a new solution in this neighborhood and try to improve it. These operations are repeated until a termination criteria satisfied.

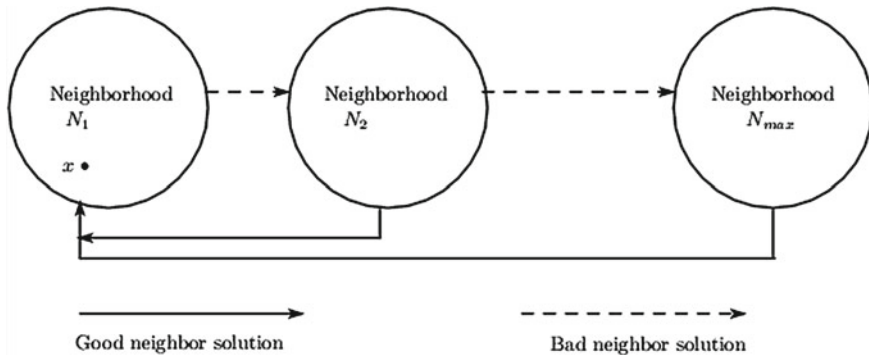


Fig. 2.2 Variable neighborhood search method

2.3 Population-Based Meta-heuristics Techniques

Population based metaheuristics methods (P-metaheuristics) start from an initial population of solutions, this is the main difference between them and the (S-metaheuristics) methods which start from a single solution. After the initial population is generated, the replacement phase is started by selecting a new population from the previous population. This operation iterates until a given stopping criteria. Most of the (P-metaheuristics) are nature-inspired methods. The most popular (P-metaheuristics) are evolutionary algorithms (EAs), deferential evolutionary (DE) [42], particle swarm optimization (PSO), ant colony (AC), group search optimizer (GSO), artificial immune system (AIS) [13]. In the following subsection we outline three of these methods, the three methods are genetic algorithm (GA), which is one of the most popular algorithm in EAs, particle swarm optimization (PSO) and group search optimizer (GSO). The three methods are different in the generation and the selection mechanisms and the search memory which they are using during the search.

Algorithm 3 Variable neighborhood search algorithm

```

Define a set of neighborhood structure  $N_k$  for  $k = 1, \dots, k_{max}$ ;
Set  $x = x_0$ ; ▷ Generate the initial solution
repeat
   $k = 1$ ;
  repeat
    Generate a random neighbor  $x'$  from the  $k^{th}$  neighborhood  $N_k(x)$  of  $x$ ;
     $x'' = \text{local search}(x')$ ;
    if  $f(x'') < f(x)$  then
       $x = x''$ ;
      Continue to search in  $N_1$ ;
       $k = 1$ ;
    else
       $k = k + 1$ ;
      Move to a new neighborhood area;
    end if
  until  $K = K_{max}$  ▷ e.g. Number of Neighborhood structures
until (Stopping criteria satisfied)
return  $x$ ; ▷ Best found solution

```

2.3.1 Evolutionary Algorithms

Evolutionary algorithms (EAs) are stochastic (P-metaheuristics) that have been successfully applied to many real and complex problems. EAs are based on the notion of competition. They are based on the evolution of a population of individuals, this population is usually generated randomly. Each individual in the population is evaluated

by using an objective function (fitness function). At each generation, individuals with better fitness are selected to form the parents. Then the selected parents are reproduced using crossover and mutation operators to generate new offsprings. In the final stage a survival selection is applied to determine which individuals of the population will survive from the offsprings and the parents. This process is iterated until a stopping criteria are satisfied. Algorithm 4 illustrates the main steps of an evolutionary algorithm.

Algorithm 4 Evolutionary algorithm

```

Set the generation counter  $t := 0$ ;
Generate an initial population  $P^0$  randomly;                                ▷ Initial population.
Evaluate the fitness function of all individuals in  $P^0$ ;
repeat
    Set  $t = t + 1$ ;                                                            ▷ Generation counter increasing.
    Select an intermediate population  $P^t$  from  $P^{t-1}$ ;                        ▷ Selection operator.
    reproduced  $P^t$ ;                                                         ▷ Crossover and mutation operators.
    Evaluate the fitness function of all individuals in  $P^t$ ;
until Termination criteria satisfied.
produce the best individual or best population found;

```

Genetic algorithm. Genetic algorithms (GAs) have been developed by Holland in the 1970s to understand the adaptive processes of natural systems [30]. Then, they have been applied to optimization and machine learning in the 1980s [10, 23]. Traditionally, GAs are associated with the use of a binary representation but nowadays one can find GAs that use other types of representations (continues). GA usually applies a crossover operator to two solutions that plays a major role, plus a mutation operator that randomly modifies the individual contents to promote diversity. GAs use a probabilistic selection that is originally the proportional selection. The replacement (survival selection) is generational, that is, the parents are replaced systematically by the offsprings. The crossover operator is based on the n-point or uniform crossover while the mutation is bit flipping. A fixed probability pm is applied to the mutation operator. The general structure of GA is shown in Algorithm 5. A problem with many standard search algorithm, i.e. hill-climbing, is that they often find solutions in the search space locally not globally when the space is not smooth. GAs due to their stochastic are able to avoid this behavior for the most part. The main steps of the GAs are illustrated as follows.

Initial population. The initial population consists of solutions, each solution is called chromosome. The chromosome is a genetic representation of a single solution to the problem and its performance at solving that problem is evaluated by a function which relates the chromosome representation of a problem. The most important skill in applying a GA to a problem is to be able to correctly map the problem to a set of integers or binary variables and compute a fitness so that it reflect the problem at hand.

Algorithm 5 Genetic algorithm

```

Set the generation counter  $t := 0$ ;
Generate an initial population  $P^0$  randomly;
Evaluate the fitness function of all individuals in  $P^0$ ;
repeat
    Set  $t = t + 1$ ;                                ▷ Generation counter increasing.
    Select an intermediate population  $P^t$  from  $P^{t-1}$ ;          ▷ Selection operator.
    Associate a random number  $r$  from (0, 1) with each row in  $P^t$ ;
    if  $r < p_c$  then
        Apply crossover operator to all selected pairs of  $P^t$ ;
        Update  $P^t$ ;
    end if                                          ▷ Crossover operator.
    Associate a random number  $r_1$  from (0, 1) with each gene;
    in each individual in  $P^t$ ;
    if  $r_1 < p_m$  then
        Mutate the gene by generating a new random value for the selected
        gene with its domain;
        Update  $P^t$ ;
    end if                                          ▷ Mutation operator.

    Evaluate the fitness function of all individuals in  $P^t$ ;
until Termination criteria satisfied
produce the best individual or best population found;

```

Selection operator. GA needs to remember good solutions and discard bad ones if it is to make progress towards the optimum solution. To make sure that the GA doesn't converge on a set of solution too quickly, a random element is usually introduced into the selection procedure. The parents are selected according to their fitness by one of the following strategies:

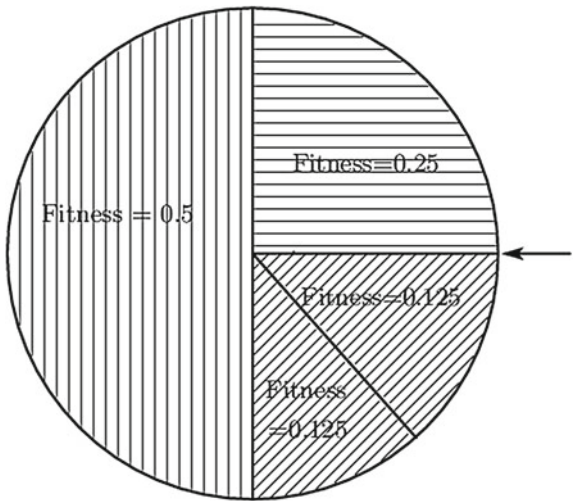
Roulette wheel selection. Roulette wheel selection is the most common selection strategy. It will assign to each individual a selection probability that is proportional to its relative fitness. In the roulette wheel selection the probability to be selected is $P_i = f_i / \sum_{j=1}^n f_j$. The total fitness of a population of individuals can be represented as a wheel, where the fitness of an individual chromosome is represented an appropriate slice of the wheel (Fig. 2.3).

Tournament selection. In tournament selection, a number of chromosomes are selected randomly (minimum 2) from the population and their fitness compared. The chromosome with greatest fitness is selected for entry to the next generation.

Rank selection. The rank of individuals is used instead of using the fitness value of an individual. The function is biased toward individuals with a high rank (the individual with good fitness). The rank may be scaled linearly using the following formula: $p(i) = \frac{2-s}{\mu} + \frac{r \cdot r(i)(s-1)}{\mu(\mu-1)}$.

Crossover operator. The role of crossover operator is to inherit some characteristics of the two parents to generate the offsprings. There are a number of methods of achieving this operator for instance, single point crossover [30], uniform crossover

Fig. 2.3 Roulette wheel selection



[34, 44], two point crossover [34], arithmetical crossover [34], geometrical crossover [35], simplex crossover [45], simulated binary crossover [7], parent-centric crossover [6]. Figures 2.4 and 2.5 show the 1-point crossover and the uniform crossover respectively.

Mutation operator. The mutation operator randomly alters one or more genes, of a selected individual (chromosome) so as to increase the structural variability of the population. The main role of mutation operator in GAs is that of restoring lost or unexplored genetic material into the population to prevent the premature convergence of GA to suboptimal solutions. Figure 2.6 shows the mutation operator. In Fig. 2.6 gene number 6 is randomly flipped from 0 to 1. Mutation can occur at each bit position in a string with some probability, usually very small (e.g., 0.01).

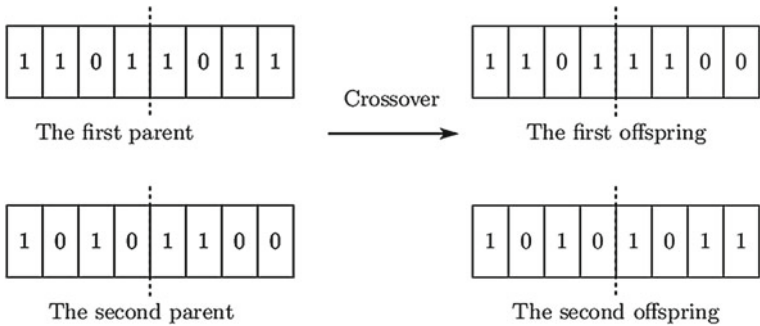


Fig. 2.4 One point crossover operation

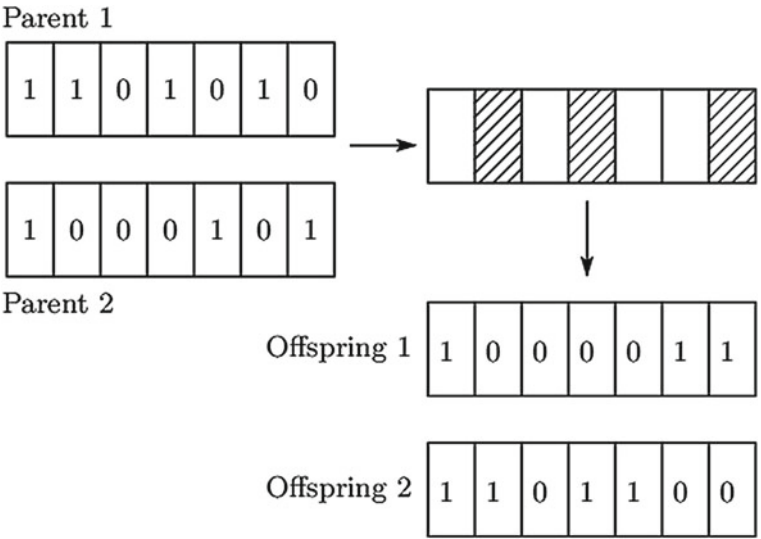


Fig. 2.5 Uniform crossover

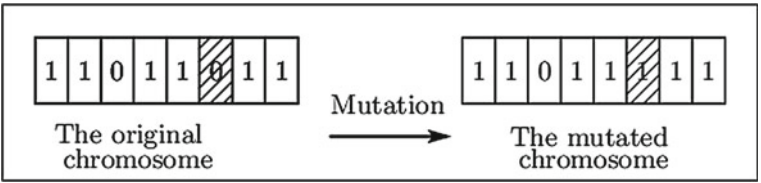


Fig. 2.6 Mutation operation

2.3.2 Swarm Intelligence

A group can be defined as a structured collection of interacting organisms (or members). The global behavior of a group (swarm) of social organisms therefore emerges in a nonlinear manner from the behavior of the individuals in that group. Thus, there exists a tight coupling between individual behavior and the behavior of the entire group. Swarm intelligence (SI) is an innovative distributed intelligence paradigm for solving optimization problems which takes inspiration from the behavior of a group of social organisms. There are many algorithms belong to SI such as ant colony optimization (ACO), particle swarm optimization (PSO), Bee colony optimization, artificial immune systems, etc. In the following subsections we outline two of these algorithms, PSO and GSO algorithms.

Particle swarm optimization. Particle swarm optimization (PSO) is one of the most popular swarm intelligence method. The initial concept of PSO was to simulate the graceful and unpredictable choreography of a bird flock [31]. In PSO, a swarm consists of a set particles, each particle represents a solution. The position of each particle is changed according to its own experience and its neighbors. Algorithm 6 shown the main structure of the particle swarm optimization method. As shown in Algorithm 6, the initial swarm is generated randomly, each particle has position x_i . At each iteration, the performance of each particle is evaluated by using the objective function. The performance of each particle is compared with its best value $pbest_i$ and global best particle $gbest$.

Algorithm 6 Particle swarm optimization algorithm

```

Set the iteration counter  $t := 0$ ;
Generate an initial swarm  $S^0$  randomly;
Evaluate the fitness function of each particle  $x_i$  in  $S^0$ ;
Set  $gbest$ ;                                ▷  $gbest$  is the best global solution in the swarm.
Set  $pbest_i$ ;                                ▷  $pbest_i$  is the best local solution in the swarm.
repeat
  Set  $t = t + 1$ ;                                ▷ Generation counter increasing.
  for  $i=1$  to  $m$  do                                ▷  $m$  is a swarm size.
     $v_i^{(t+1)} = v_i^{(t)} + c1 \times (pbest_i - x_i^{(t)}) + c2 \times (gbest - x_i^{(t)})$ ;    ▷ Update velocities.
     $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$ ;                                ▷ Update particle positions.
    Evaluate the fitness function  $f(x_i)$  of each particle  $x_i$  in  $S^t$ ;
    if  $f(x_i) < f(pbest_i)$  then                                ▷ Solving minimization problem.
       $pbest_i = x_i$ ;
    end if
    if  $f(x_i) < f(gbest)$  then
       $gbest = x_i$ ;
    end if
    Update  $x_i, v_i$ ;
  end for
until Termination criteria satisfied.
produce the best particle;

```

The position of each particle x_i is changed as shown in Eq. 2.2.

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t)} \quad (2.2)$$

where v_i is a particle velocity, the velocity of each particle is changed as shown in Eq. 2.3.

$$v_i^{(t+1)} = v_i^{(t)} + c1 \times (pbest_i - x_i^{(t)}) + c2 \times (gbest - x_i^{(t)}) \quad (2.3)$$

$c1, c2$ are positive acceleration constant. The operation is repeated until termination criteria satisfied. Usually a PSO algorithm is executed for a fixed number of iterations.

Group search optimizer. Group search optimizer (GSO) is a recent swarm intelligence algorithm (SI) proposed by [25]. GSO is based on Producer-Scrounger (PS) behavior of group living animals [24, 25], which assume group members producing (searching for foods) and scrounging (joining resources uncovered by others). Because of the efficiency and the promising performance in terms of accuracy and convergence speed of GSO, many researchers have been attracted to apply GSO algorithm in many applications. For example Fang et al. [12] proposed a hybrid group search optimizer algorithm to solve optimization problems. He et al. [26] proposed interactive dynamic neighbor deferential evolutionary GSO (IDGSO) to solve high dimensional problems. Akhand et al. [2] employ the concept of swap operator (SO) and swap sequence to modify GSO for travelling salesman problem (TSP). Liu et al. [33] presented a modified group search optimizer algorithm for high dimensional function optimization, which is based on Levy flight strategy, self-adaptive joining strategy, and chaotic mutation strategy. In the following section, we give an overview of GSO algorithm for function optimization.

Group search optimizer (GSO) is the novel population based nature inspired algorithm, especially animal searching behavior. The population of GSO algorithm is called a group and each individual in the population is called a member. The i th member at the k th iteration, has a current position $X_i^k \in R^n$, a head angle $D_i^k(\phi_i^k) = (d_{i1}^k, \dots, d_{in}^k) \in R^n$. Search direction (head direction) $\phi_i^k = (\psi_{i1}^k, \dots, \psi_{i1}^k) \in R^{n-1}$ that can be calculated from ϕ_i^k via a polar to Cartesian coordinates transformation as shown in Eq. 2.4

$$\begin{aligned} d_{i1}^k &= \prod_{p=1}^{n-1} \cos(\psi_{ip}^k) \\ d_{ij}^k &= \sin(\psi_{i(j-1)}^k) \cdot \prod_{p=i}^{n-1} \cos(\psi_{ip}^k) \\ d_{in}^k &= \sin(\psi_{i(n-1)}^k) \end{aligned} \quad (2.4)$$

In GSO, there are three kinds of members in a group: producer, scroungers and rangers (dispersed) members. There is only one producer at each searching iteration and remaining members are scroungers and rangers members. At each iteration the group member which has best fitness value is selected as the producer. All scroungers will join the resource found by the producer, rangers members are less efficient members who perform random walks. At k th iteration the producer X_p behaves as follows:

1. The producer will scan at zero degree and then scan laterally by randomly sampling three points in the scanning field:
one point at zero degree

$$X_z = X_p^k + r_1 l_{max} D_p^k(\phi^k) \quad (2.5)$$

one point in the right hand side hypercube

$$X_r = X_p^k + r_1 l_{max} D_p^k(\phi^k + r_2 \theta_{max}/2) \quad (2.6)$$

and one point in the left hand side hypercube

$$X_l = X_p^k + r_1 l_{max} D_p^k(\phi^k - r_2 \theta_{max}/2) \quad (2.7)$$

where $r_1 \in R^1$ is a normally distributed random number with mean 0 and standard deviation 1, $r_2 \in R^{n-1}$ is a random sequence in the range (0, 1), $\theta_{max} \in R^1$ is the maximum pursuit angle and l_{max} is the maximum pursuit distance.

2. The producer will then find the best point with the best resource (fitness value). If the best point has a better resource than its current position, then it will fly to this point. Otherwise it will stay in its current position and turn its head to a new randomly generated angle:

$$\phi^{k+1} = \phi^k + r_2 \alpha_{max} \quad (2.8)$$

where α_{max} is the maximum tuning angle.

3. If the producer cannot find a better area after a iterations, it will turn its head back to zero degree:

$$\phi^{k+a} = \phi^k \quad (2.9)$$

where $a \in R^1$ is a constant.

At k th iteration a number of group members are selected as scroungers, these members walks randomly toward the producer.

$$X_i^{k+1} = X_i^k + r_3(X_p^k - X_i^k) \quad (2.10)$$

where $r_3 \in R^n$ is a uniform random sequence in the range (0, 1). The group members, who are less efficient foragers than the dominant (rangers), will be dispersed from the group. If the i th group member is dispersed, they random walks searching for randomly distributed resources.

$$l_i = ar_1 l_{max} \quad (2.11)$$

and move to the new point

$$X_i^{k+1} = X_i^k + l_i D_i^k(\phi^{k+1}) \quad (2.12)$$

The main structure of GSO algorithm is shown in Algorithm 7.

2.4 Metaheuristics as a Tool for Bioinformatics Applications

The following sections show how metaheuristics methods described above can be applied with three different bioinformatics applications.

Algorithm 7 Group search optimizer algorithm

Set values of θ_{max} , l_{max} , α_{max} ; Generate an initial group randomly for all the members X_i ; Evaluate all the group members by calculating the fitness values. of each member $f(X_i)$; repeat repeat Find the producer X_p of the group; Create new points using the producer by randomly sampling three points in the scanning filed as shown in Equations 2.5, 2.6, 2.7; Evaluate the fitness function of each generated point; if the best point is better than the current position then Producer will fly to the best point; else Stay in the current position and turn its head to a new randomly generated angle as shown in Equation 2.8; end if Select a number of group members as scroungers; Each scrounger walks randomly to join the resources founded by the producer as shown in Equation 2.10; Select rest of the members as rangers (dispersed); Each dispersed member walks randomly searching for randomly distributed resources as shown in Equation 2.12; until (Visiting each member i in the group) until (Stopping criteria satisfied)	▷ Parameters initialization ▷ Group initialization ▷ Members evaluation ▷ Producing operation ▷ Accept the best point ▷ Scrounging operation ▷ Dispersion operation
--	--

2.4.1 Application 1: Selecting Genes from Gene Expression Data for Cancer Classification

Gene selection is an important component for gene expression-cancer (tumor) classification system. A large datasets of thousand of genes are produced by microarray experiments with expression values in order to be useful for cancer prediction. Most of genes in microarray may be irrelevant genes or noisy genes which make these genes difficult to analysis. Many efficient metaheuristics methods have been produced to solve this problem such as GAs, evolution algorithms (EAs) [16, 19, 38, 41, 49], simulated annealing (SA), tabu search (TS) and particle swarm optimization (PSO). Tabu search and particle swarm optimization are combined to solve this problem in many works, for example [40]. In [40] a hybrid PSO and tabu search method is proposed for gene selection for tumor classification, the method is called (HPSOTS).

Tabu search has the ability to avoid convergence to local minima, it increases the exploitation process of the algorithm. The main structure of TS algorithm is shown Algorithm 1. The PSO based methods are intractable to efficiently produce a small subset of informative genes for high classification accuracy. The main algorithm of PSO is shown in Algorithm 6. The main steps of HPSOTS method is described as follows.

1. Randomly initialize all the initial binary strings IND in HPSOTS with an appropriate size of population and evaluate the fitness function of individual in IND. IND is strings of binary bits corresponding to each gene.
2. Generate and evaluate the neighbors of 90 % of individual in IND according to information sharing mechanism of PSO.
3. Pick new individual from the explored neighborhood according to the aspiration criteria and tabu conditions and update the IND population.
4. To improve further the ability of HPSOTS to overleap local optima, other 10 % of particle in IND are forced to fly randomly not following the two best particles. Evaluate the fitness function of these ten percent of particles.
5. If the best object function of the generation fulfills the end condition, the training is stopped with the results output, otherwise, go to the step to renew population.

The different microarray data sets were used by HPSOTS method. HPSOTS are compared with the standard TS and PSO methods, the results show that HPSOTS method is a useful tool for gene selection and mining high dimensional data.

2.4.2 Application 2: Molecular 3D Structure Prediction

The potential energy of a molecule is derived from molecular mechanics, which describes molecular interactions based on the principles of Newtonian physics. An empirically derived set of potential energy contributions is used for approximating these molecular interactions. The molecular model considered here consists of a chain of m atoms centered at x_1, \dots, x_m , in a 3-dimensional space as shown in Fig. 2.7. For every pair of consecutive atoms x_i and x_{i+1} , let $r_{i,i+1}$ be the bond length which is the Euclidean distance between them. For every three consecutive atoms x_i, x_{i+1}, x_{i+2} , let $\theta_{i,i+2}$ be the bond angle corresponding to the relative position of the third atom with respect to the line containing the previous two. Likewise, for every four consecutive atoms $x_i, x_{i+1}, x_{i+2}, x_{i+3}$, let $\omega_{i,i+3}$ be the angle, called the torsion angle, between the normal through the planes determined by the atoms x_i, x_{i+1}, x_{i+2} and $x_{i+1}, x_{i+2}, x_{i+3}$.

The force field potentials corresponding to bond lengths, bond angles, and torsion angles are defined respectively [8] as

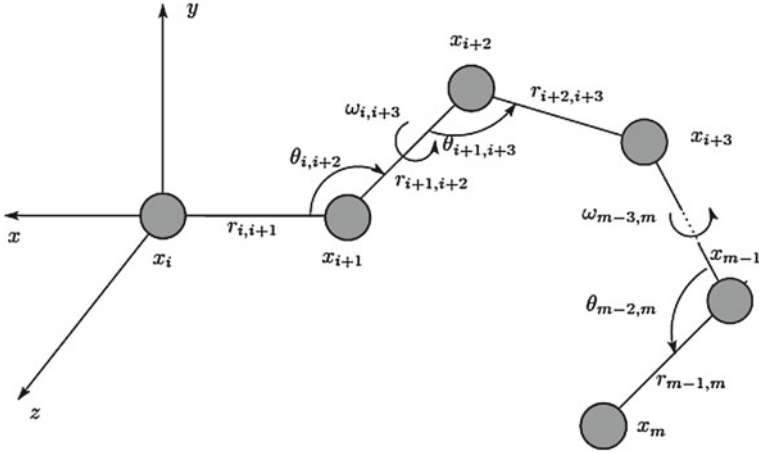


Fig. 2.7 Coordinate set of atomic chain

$$\begin{aligned}
 E_1 &= \sum_{(i,j) \in M_1} c_{ij}^1 (r_{ij} - r_{ij}^0)^2, \\
 E_2 &= \sum_{(i,j) \in M_2} c_{ij}^2 (\theta_{ij} - \theta_{ij}^0)^2, \\
 E_3 &= \sum_{(i,j) \in M_3} c_{ij}^3 (1 + \cos(3\omega_{ij} - \omega_{ij}^0)),
 \end{aligned} \tag{2.13}$$

where c_{ij}^1 is the bond stretching force constant, c_{ij}^2 is the angle bending force constant, and c_{ij}^3 is the torsion force constant. The constant r_{ij}^0 and θ_{ij}^0 represent the “preferred” bond length and bond angle, respectively, and the constant ω_{ij}^0 is the phase angle that defines the position of the minima. The set of pairs of atoms separated by k covalent bond is denoted by M_k for $k = 1, 2, 3$.

In addition to the above, there is a potential E_4 which characterizes the 2-body interaction between every pair of atoms separated by more than two covalent bonds along the chain. We use the following function to represent E_4 :

$$E_4 = \sum_{(i,j) \in M_3} \left(\frac{(-1)^i}{r_{ij}} \right), \tag{2.14}$$

where r_{ij} is the Euclidean distance between atoms x_i and x_j .

The general problem is the minimization of the total molecular potential energy function, $E_1 + E_2 + E_3 + E_4$, leading to the optimal spatial positions of the atoms. To reduce the number of parameters involved in the potentials above, we simplify the problem considering a chain of carbon atoms.

In most molecular conformational predictions, all covalent bond lengths and covalent bond angles are assumed to be fixed at their equilibrium values r_{ij}^0 and θ_{ij}^0 , respectively. Thus, the molecular potential energy function reduces to $E_3 + E_4$ and the first three atoms in the chain can be fixed. The first atom, x_1 , is fixed at the origin, $(0, 0, 0)$; the second atom, x_2 , is positioned at $(-r_{12}, 0, 0)$; and the third atom, x_3 , is fixed at $(r_{23} \cos(\theta_{13}) - r_{12}, r_{23} \sin(\theta_{13}), 0)$.

Using the parameters previously defined and Eqs. (2.13) and (2.14), we obtain

$$E = \sum_{(i,j) \in M_3} (1 + \cos(3\omega_{ij})) + \sum_{(i,j) \in M_3} \left(\frac{(-1)^i}{r_{ij}} \right). \quad (2.15)$$

Although the molecular potential energy function does not actually model the real system, it allows one to understand the qualitative origin of the large number of local minimizers the main computational difficulty of the problem, and is likely to be realistic in this respect.

Note that E_3 in Eq. (2.13), is expressed as a function of torsion angles, and E_4 in Eq. (2.14), is expressed as a function of Euclidean distance. To represent Eq. (2.15) as a function angles only, we can use the result established in [39] and obtain

$$\begin{aligned} r_{il}^2 = & r_{ij}^2 + r_{jl}^2 - r_{ij} \left(\frac{r_{jl}^2 + r_{jk}^2 - r_{kl}^2}{r_{jk}} \right) \cos(\theta_{ik}) \\ & - r_{ij} \left(\frac{\sqrt{4r_{jl}^2 r_{jk}^2 - (r_{jl}^2 + r_{jk}^2 - r_{kl}^2)^2}}{r_{jk}} \right) \sin(\theta_{ik}) \cos(\omega_{il}), \end{aligned}$$

for every four consecutive atoms x_i, x_j, x_k, x_l . Using the parameters previously defined, we have

$$r_{ij} = \sqrt{10.60099896 - 4.141720682(\cos(\omega_{ij}))} \quad \text{for all } (i, j) \in M_3. \quad (2.16)$$

From Eqs. (2.15) and (2.16), the expression for the potential energy as a function of the torsion angles takes the form

$$E = \sum_{(i,j) \in M_3} \left(1 + \cos(3\omega_{ij}) + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682(\cos(\omega_{ij}))}} \right), \quad (2.17)$$

where $i = 1, \dots, m-3$ and m is the number of atoms in the given system. The problem is then to find $\omega_{14}, \omega_{25}, \dots, \omega_{(m-3)m}$, considering $\omega_{ij} \in [0, 5]$, which corresponding to the global minimum of the function E , represented by Eq. (2.17). E is a nonconvex function involving numerous local minimizers even for small molecules. Finally, the function $f(x)$ can be defined as

$$f(x) = \sum_{i=1}^n \left(1 + \cos(3x_i) + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682(\cos(x_i))}} \right) \quad (2.18)$$

and $0 \leq x_i \leq 5$, $i = 1, \dots, n$.

Despite these simplification, the problem remains very difficult. A molecule with as few as 30 atoms has $2^{27} = 134,217,728$ local minimizers.

Many metaheuristics methods are applied to solve this problem see for example, [1, 3, 4, 28, 29]. To solve this problem, we proposed a group search optimizer method called group search optimizer with matrix coding partitioning (GSOMCP) to minimize the molecular potential energy function.

The algorithmic scenario of GSOMCP is described as follows. Each individual in the search space consists of n variables. GSOMCP starts with an initial population containing μ chromosomes. Therefore, the population can be coded as a matrix of size $\mu \times n$ called population matrix (PM). At generation t , the PM^t matrix is partitioned into several sub-matrices $PM_{(i,j)}^t$, $i = 1, \dots, \eta$, $j = 1, \dots, \nu$. The formal detailed description of GSOMCP is given in the following algorithm.

Algorithm 8 The proposed GSOMCP algorithm

Set values of m , μ , ν , and η . Set the generation counter $t := 0$.

Generate an initial population P_0 of size μ and code it to a matrix PM^0 .

$t = 0$

repeat

 Partition $\widetilde{PM^t}$ into $\nu \times \eta$ sub-matrices.

repeat

 Apply Algorithm 7 on each partition.

until visit all partitions

$t = t + 1$

until termination criteria satisfied

 produce the best solution;

Also in order to solve the molecular 3D structure prediction problem, we proposed a hybrid method by combining a variable neighborhood search and simulated annealing algorithm. The method is called simulated annealing with variable partitioning (SAVP). The description of SAVP are presented as follows.

SAVP starts with an initial solution x^0 generated randomly. At each iteration the solution is divided into η partitions. The variable neighborhood zone is generated in order to generate a trail neighborhood solutions in the random selected partitions. The generated neighbor solution that improve the objective function is always selected. Otherwise the solution is accepted with a given probability $e^{\frac{-\Delta E}{T}}$, where T is the current temperature, and ΔE the amount of the degradation of the objective function. The scenario is repeated until the equilibrium state is reached. In SAVP the equilibrium state is a given number of iterations executed at each temperature, this number is equals to μ , μ is a user predefined number. Once the equilibrium state is reached the temperature is decreased gradually according to a cooling schedule.

This process is repeated until the stopping criteria satisfied, which is in our algorithm $T \leq T_{min}$. The structure of the SAVP with the formal detailed description is given in Algorithm 9.

Algorithm 9 SAVP Algorithm

Choose an initial solution x^0 ; Set initial values for $T_{max}, T_{min}, \beta, \mu, \nu, z_0$; Set $z = z_0, T = T_{max}, x = x^0$; repeat $k := 0$; repeat Partition the solution x into η partitions; where $\eta = n/\nu$; Generate neighborhood trials y^1, \dots, y^μ around x in the generated neighborhood zones; Set x' equal to the best trial solution from y^1, \dots, y^μ ; $\Delta E = f(x') - f(x)$; if $\Delta E \leq 0$ then $x = x'$; else if $\text{rand}() < e^{\frac{-\Delta E}{T}}$ then $x = x'$; end if end if $k := k + 1$; until $k \leq \mu$ $T = T - \beta$; until $T \leq T_{min}$ best solution obtained in the previous stage;	▷ Generate the initial solution randomly ▷ Parameters Initialization ▷ Initial counter ▷ At a fixed temperature ▷ Variable partitioning ▷ Tail solution generation ▷ Accept the neighbor solution ▷ $\text{rand}() \in (0,1)$ ▷ Accept the solution with a probability $e^{\frac{-\Delta E}{T}}$ ▷ Increment counter ▷ Equilibrium condition ▷ Temperature update ▷ Stopping criteria satisfied
---	---

2.4.3 Application 3: Multiple Sequence Alignment

The third application is the multiple sequence alignment, which is the task of comparing sequences of nucleic or amino acids and find the similarity in the structure between genes and protein. Also it used to predict the 3D structure of the protein by comparing the primary structure of two proteins one of them with known 3D structure and the 3D structure of the other is unknown. The comparison of these sequences can help in the discovery of similar genes across species. For example, a very simple method would be to write to compare two sequences as shown in Fig. 2.8. In Fig. 2.8, the “√” character means a matched column between the two sequences, whereas the character “x” means there is no matched column between the two sequences. However if any editing operation is done by inserting or deleting any characters, the sequences alignment no longer exists. The problem now is to determine the optimal alignment of these sequences by correctly inserting gaps to realign the sequences.

Fig. 2.8 Multiple sequence alignment

T	A	C	G
T	C	C	G
✓	x	✓	✓

T	A	C	G	-
-	T	C	C	G
x	x	x	x	x

This problem becomes very difficult when the number of bases in a typical gene is very big. Genetic algorithms have been successful in this problem, the main structure of GA is presented in Algorithm 5. The implementation of GA with the multiple sequence alignment is described as follows.

1. The initial population of alignments are generated, where each alignment is evaluated according to its performance in terms of the number of columns which match and the number of gaps which are introduced into the sequences.
2. GA uses survival selection to select the elitist alignments, where 50 % of the best alignments are copied to the next generations.
3. The one point crossover is applied to generate the offspring as shown in Fig. 2.4, by taking two separate alignments, make a cut at a random point in the first alignment sequence and cut the second alignment at such a point that every sequence is cut adjacent to the same symbol.
4. The operation is repeated until termination criteria satisfied.

The representation of the problem is fixed, alignment can be made by inserting gaps in a sequence, and the genetic operators (crossover and mutation) had to be modified accordingly.

2.5 Conclusion

We provided a good examples of how metaheuristics methods could be combined together or work individually to produce good results when applied to bioinformatics applications. The metaheuristics methods are classified into two classes, single

solution based methods and population based methods. The single solution based methods or the trajectory methods start the search with a single solution, whereas the population based methods start the search with a group of solution called population. Six different metaheuristic methods are presented in this work, three of them are single solution based methods such as tabu search (TS), simulated annealing (SA) and variable neighborhood search (VNS), the other three methods are population based methods such as genetic algorithm (GA), particle swarm optimization (PSO) and group search optimizer (GSO). These methods have been applied to three different bioinformatics applications. The first application is the gene selection from gene expression data for cancer classification by applying a hybrid particle swarm optimization method with tabu search methods. The second application is minimizing the molecular potential energy function by proposing a new group search optimization algorithm. The last application is the alignment of multiple sequence using genetic algorithm (GA). As these examples showed the advantage of using the metaheuristic method for a different application in bioinformatics.

References

1. Ali, A.F., Hassanien, A.E.: Minimizing molecular potential energy function using genetic Nelder-Mead algorithm. In: 8th International Conference on Computer Engineering & Systems (ICCES), pp. 177–183 (2013)
2. Akhand, M.A.H., Junaed, A.B.M., Murase, K.: Group search optimization to solve traveling salesman problem. In: 15th ICCIT 2012, University of Chittagong, 22–24 Dec 2012
3. Bansal, J.C.: Shashi, Deep, K., Katiyar, V.K.: Minimization of molecular potential energy function using particle swarm optimization. *Int. J. Appl. Math. Mech.* **6**(9), 1–9 (2010)
4. Barbosa, H.J.C., Lavor, C., Raupp, F.M.: A GA-simplex hybrid algorithm for global minimization of molecular potential energy function. *Ann. Oper. Res.* **138**, 189–202 (2005)
5. Chelouah, R., Siarry, P.: Tabu search applied to global optimization. *Eur. J. Oper. Res.* **123**, 256–270 (2000)
6. Deb, K., Joshi, D.: A computationally efficient evolutionary algorithm for real parameter optimization, Technical Report 003, KanGal (2002)
7. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. *Complex Syst.* **9**, 115–148 (1995)
8. Dražić, M., Lavor, C., Maculan, N., Mladenović, N.: A continuous variable neighborhood search heuristic for finding the three-dimensional structure of a molecule. *Eur. J. Oper. Res.* **185**, 1265–1273 (2008)
9. Crainic, T.G., Toulouse, M.: Parallel strategies for metaheuristics. In: Glover, F.W., Kochenberger, G.A. (eds.) *Handbook of Metaheuristics*, pp. 475–513. Springer (2003)
10. De Jong, K.A.: Genetic algorithms: a 10 year perspective. In: *International Conference on Genetic Algorithms*, pp. 169–177 (1985)
11. Dorigo, M.: Optimization, learning and natural algorithms, Ph.D. thesis, Politecnico di Milano, Italy (1992)
12. Fang, J.Y., Cui, Z.H., Cai, X.J., Zeng, J.C.: A Hybrid group search optimizer with metropolis rule, In: *Proceedings of the 2010 International Conference on Modeling, Identification and Control (ICMIC)*, Okayama, Japan, pp. 556–561 (2010)
13. Farmer, J.D., Packard, N.H., Perelson, A.S.: The immune system, adaptation, and machine learning. *Physica D* **2**, 187–204 (1986)

14. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **8**, 67–71 (1989)
15. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. *J. Global Optim.* **6**, 109–133 (1995)
16. Furey, T., Cristianini, N., Duffy, N., Bednarski, D., Schummer, M., Haussler, D.: Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* **16**, 906–914 (2000)
17. Flynn, M.J.: Some computer organizations and their effectiveness. *IEEE Trans. Comput.* **C-21**, 948–960 (1972)
18. Gendreau, M., Potvin, J.Y.: Chapter 6: Tabu search. In: Burke, E.K., Kendall, G. (eds.) *Search Methodologies*, pp. 165–186. Springer (2006)
19. Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., et al.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286**, 531–537 (1999)
20. Glover, F.: Parametric combinations of local job shop rules. In: ONR Research Memorandum, No. 117, GSIA, Carnegie Mellon University, Pittsburgh (1963)
21. Glover, F.: Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13**, 533–549 (1986)
22. Glover, F.: A template for scatter search and path relinking. *Lect. Notes Comput. Sci.* **1363**, 13–54 (1997)
23. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading (1989)
24. He, S., Wu, Q.H., Saunders, J.R.: A novel group search optimizer inspired by animal behavioral ecology. In: *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, Vancouver, BC: Sheraton Vancouver Wall Center, pp. 1272–1278, July (2006)
25. He, S., Wu, Q.H., Saunders, J.R.: Group search optimizer—an optimization algorithm inspired by animal searching behavior. *IEEE Trans. Evol. Comput.* **13**(5), 973–990 (2009)
26. He, G.H., Cui, Z.H., Tan, Y.: Interactive dynamic neighborhood differential evolutionary group search optimizer. *J. Chin. Comput. Syst.* (accepted, 2011)
27. Hedar, A., Ali, A.F.: Tabu search with multi-level neighborhood structures for high dimensional problems. *Appl. Intell.* **37**, 189–206 (2012)
28. Hedar, A., Ali, A.F., Hassan, T.: Genetic algorithm and tabu search based methods for molecular 3D-structure prediction. *Int. J. Numer. Algebra, Control Optim. (NACO)* (2011)
29. Hedar, A., Ali, A.F., Hassan, T.: Finding the 3D-structure of a molecule using genetic algorithm and tabu search methods. In: *Proceeding of the 10th International Conference on Intelligent Systems Design and Applications (ISDA2010)*, Cairo, Egypt (2010)
30. Holland, J.H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor (1975)
31. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. *Proc. IEEE Int. Conf. Neural Netw.* **4**, 1942–1948 (1995)
32. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. *Science* **220**, 671–680 (1983)
33. Liu, C., Wang, L., Yang, A. (eds.): A Modified group search optimizer algorithm for high dimensional function optimization. In: *ICICA, Part II, CCIS*, vol. 308, pp. 219–226 (2012)
34. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, New York (1992)
35. Michalewicz, Z., Nazhiyath, G., Michalewicz, M.: A note on usefulness of geometrical crossover for numerical optimization problems. In: *5th Annual Conference on Evolutionary Programming*, San Diego, CA. MIT Press, pp. 305–312 (1996)
36. Mladenovic, N.: A variable neighborhood algorithm a new metaheuristic for combinatorial optimization. In: *Abstracts of Papers Presented at Optimization Days*, Montral, Canada, p. 112 (1995)
37. Mladenovic, M., Hansen, P.: Variable neighborhood search. *Comput. Oper. Res.* **24**, 1097–1100 (1997)

38. Peng, S.H., Xu, Q.H., Ling, X.B., Peng, X.N., Du, W., Chen, L.B.: Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *FEBS Lett.* **555**, 358–362 (2003)
39. Pogorelov, A.: *Geometry*. Mir Publishers, Moscow (1987)
40. Shen, Q., Wei-Min, S., Wei, K.: Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data. *Comput. Biol. Chem.* **32**, 53–60 (2008)
41. Sima, C., Dougherty, E.R.: What should be expected from feature selection in small-sample settings. *Bioinformatics* **22**(19), 2430–2436 (2006)
42. Storn, R.M., Price, K.V.: Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**, 341–359 (1997)
43. Stitzle, T.: Local search algorithms for combinatorial problems: analysis, improvements, and new applications, Ph.D. thesis, Darmstadt University of Technology (1998)
44. Syswerda, G.: Uniform crossover in genetic algorithms. In: Schaffer, J.D. (ed.) *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 2–9. Morgan Kaufmann Publishers, San Mateo (1989)
45. Tsutsui, S., Yamamura, M., Higuchi, T.: Multi-parent recombination with simplex crossover in real-coded genetic algorithms. In: *GECCO99 Genetic and Evolutionary Computation Conference*, pp. 657–664 (1999)
46. Voudouris, C.: Guided local search for combinatorial optimization problems, Ph.D thesis, University of Essex (1997)
47. Voudouris, C.: Guided local search: an illustrative example in function optimization. *BT Technol. J.* **16**, 46–50 (1998)
48. Voudouris, C., Tsang, E.: Guided local search. *Eur. J. Oper. Res.* **113**, 469–499 (1999)
49. Xiong, M., Li, W., Zhao, J., Jin, L., Boerwinkle, E.: Feature (gene) selection in gene expression-based tumor classification. *Mol. Genet. Metab.* **73**, 239–247 (2001)

Applications of Intelligent Optimization in Biology and
Medicine

Current Trends and Open Problems

Hassanien, A.-E.; Grosan, C.; Tolba, F.M. (Eds.)

2016, XIII, 307 p. 100 illus., 47 illus. in color., Hardcover

ISBN: 978-3-319-21211-1