PCE16136

WATER RESOURCES SYSTEMS PLANNING & MANAGEMENT

LECTURE NOTES

Module-II

Prepared By

Dr. Prakash Chandra Swain

Professor in Civil Engineering

Veer Surendra Sai University of Technology, Burla

Branch - Civil Engineering
Specialization-Water Resources Engineering
Semester – 2th Sem

Department Of Civil Engineering
VSSUT, Burla

Disclaimer

Course Content

Module II

Methods of Systems Analysis: Linear Programming Models, Simplex Method, Sensitivity Analysis, Dual Programming, Dynamic Programming Models, Classical Optimisation Techniques, Non-linear Programming, Gradient Techniques, Genetic Algorithm, Stochastic Programming, Simulation, Search Techniques, Multi Objective Optimisation.

Lecture note 1

Methods of system analysis

1.1 Introduction

1. Identification of objectives

- very important- If the correct objectives are not identified, the correct problem will not be solved
- consult others
- use multi-disciplinary team
- may have multiple objectives
- determine your client - usually person paying the bill
- establish the needs of the client - sometimes difficult to establish
- identify the clients single most important objective
- choose a measure of effectiveness
- discuss the project objective with the client
- insure that the client clearly understands and agrees with the project objective

2. Quantification of objectives

- Identify and write objective function - this is a quantitative expression of the goals or objectives of the project
- objective function might take on the form F=G(X1, X2, X3, ..., Xn) where Xi's are independent variables and represent values of parameters under the control of the systems analyst
- constraint set should be identified; The constraint set consists of equations that define the domain of feasible solutions. For example, in determining the optimum mix of corn and soybeans to plant on a 450 hectare farm, a constraint on the amount of land that can be used might be written as: Corn Hectares + Soybean Hectares <= 450.

3. Development of a system model

- most often this is the responsibility of the systems analyst or engineer
- keep in mind that the model is an abstraction of the system
- a two stage process is sometimes used:
  - model decoupling - simplifying step where system components are modeled and analyzed as subsystems. This can be helpful in better understanding the system.
  - model integration - entire system is modeled (e.g., the subsystem components are integrated)
- delicate balance exists between model detail and the ability to effectively and efficiently analyze the mode. Modeling detail may offer better reality at increased computational expense. Under certain circumstances, a simple model may prove more valuable than a more complex model. The project objectives should dictate the level of detail required.
- many types of models are available for use
- the type of model chosen depends on system, the objectives, perspective (time scale) of models
- one should select the most "appropriate model" - by the end of the semester you should have a better feel for this
- why model?

4. Evaluation of alternatives

- goal is to find an optimum solution
- identify alternative solutions
- gather as much information about alternative solutions as possible - may require searching the literature, obtaining technical and cost data on equipment, operation, maintenance, and other pertinent information
- perform sensitivity analysis to determine response to change in model parameters
- verification - computer code reproduces model chosen
- validation - model of system faithfully reproduces the actual system

5. Detailed design and development

- complete the design and necessary actions

Optimum solution - the combination of resources that best meets the stated objective(s) and satisfies all constraints.

1.2 Types of models

- iconic - physical models that are images of the real world; dimensions are usually scaled up or down; for example, models of cars might be constructed and tested in a wind tunnel
- analog - model that substitutes one set of properties for another; may be iconic or mathematical; electric resistance often used as an analog of the friction of a fluid flowing in a pipe; this approach is not as widely used as at one time - digital computers have allowed the development of other modeling techniques that have replaced analog models
- stochastic - probabilistic model that uses randomness to account for unmeasurable factors (e.g., weather)
- deterministic - model that does not use randomness but uses explicit expressions for relationships that may or may not involve time rates of change
- discrete - model where state variables change in steps as opposed to continuously with time (e.g., number of cattle in a barn); may be deterministic or stochastic
- continuous - model whose state variables change continuously with time (e.g., biomass in a field); usually sets of differential equations used; initial conditions required (can be difficult to obtain for some systems!)
- combined - model where some state variables change continuously and others change in steps at event times; for example, a field of hay might be modeled using a combined approach with the biomass modeled continuously during growth and then as a discrete event when harvested
- mathematical - abstract model usually written in equation form
- object-oriented - use objects that are abstractions of real world objects and develop relationships and actions between objects; comes from field of artificial intelligence
- heuristic - heuristics (rules) are used to model the system; comes from field of artificial intelligence

1.3 Linear programming model

Linear programming (LP, also called linear optimization) is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships. Linear programming is a special case of mathematical programming (mathematical optimization).

More formally, linear programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. Its feasible region is a convex polytope, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function is a real-valued affine (linear) function defined on this polyhedron. A linear programming algorithm finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.

Linear Programming (LP) is the most useful optimization technique used for the solution of engineering problems. The term "linear" implies that the objective function and constraints are "linear" functions of "nonnegative" decision variables. Thus, the conditions of LP problems (LPP) are

1. Objective function must be a linear function of decision variables

2. Constraints should be linear function of decision variables

3. All the decision variables must be nonnegative

For example,

$$\begin{array}{lll}
\text{Maximize} & Z = 6x + 5y & \prec \text{Objective Function} \\
\text{subject to} & 2x - 3y \le 5 & \prec \text{1st Constraint} \\
 & x + 3y \le 11 & \prec \text{2nd Constraint} \\
 & 4x + y \le 15 & \prec \text{3rd Constraint} \\
 & x,\ y \ge 0 & \prec \text{Nonnegativ ity Condition}
\end{array}$$

is an example of LP problem. However, example shown above is in "general" form.

<u>Canonical Form of Lpp</u>

Canonical form of standard LPP is a set of equations consisting of the "objective function" and all the "equality constraints" (standard form of LPP) expressed in canonical form. Understanding the canonical form of LPP is necessary for studying simplex method, the most popular method of solving LPP.

1.4 Simplex method

The Simplex method is a method that proceeds from one BFS or extreme point of the feasible region of an LP problem expressed in tableau form to another BFS, in such a way as to continually increase (or decrease) the value of the objective function until optimality is reached. The simplex method moves from one extreme point to one of its neighbouring extreme point.

The Simplex method is an approach to solving linear programming models by hand using slack variables, tableaus, and pivot variables as a means to finding the optimal solution of an optimization problem. A linear program is a method of achieving the best outcome given a maximum or minimum equation with linear constraints. Most linear programs can be solved using an online solver such as Matlab, but the Simplex method is a technique for solving linear programs by hand. To solve a linear programming model using the Simplex method the following steps are necessary:

- Standard form
- Introducing slack variables
- Creating the tableau
- Pivot variables
- Creating a new tableau
- Checking for optimality
- Identify optimal values

Here it breaks down the Simplex method into the above steps and follows the example linear programing model shown below throughout the entire document to find the optimal solution.

$$\begin{aligned} Minimize: -z &= -8x_1 - 10x_2 - 7x_3 \\ s.t.: x_1 + 3x_2 + 2x_3 &\leq 10 \\ -x_1 - 5x_2 - x_3 &\geq -8 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Step 1: Standard Form

Standard form is the baseline format for all linear programs before solving for the optimal solution and has three requirements: (1) must be a maximization problem, (2) all linear constraints must be in a less-than-or-equal-to inequality, (3) all variables are non-negative. These requirements can always be satisfied by transforming any given linear program using basic algebra and substitution. Standard form is necessary because it creates an ideal starting point for solving the Simplex method as efficiently as possible as well as other methods of solving optimization problems.

To transform a minimization linear program model into a maximization linear program model, simply multiply both the left and the right sides of the objective function by -1.

$$-1 \times (-z = -8x_1 - 10x_2 - 7x_3)$$
$$z = 8x_1 + 10x_2 + 7x_3$$
$$Maximize : z = 8x_1 + 10x_2 + 7x_3$$

Transforming linear constraints from a greater-than-or-equal-to inequality to a less-than-or-equal-to inequality can be done similarly as what was done to the objective function. By multiplying by -1 on both sides, the inequality can be changed to less-than-or-equal-to.

$$-1 \times (-x_1 - 5x_2 - x_3 \geq -8)$$
$$x_1 + 5x_2 + x_3 \leq 8$$

Once the model is in standard form, the slack variables can be added as shown in Step 2 of the Simplex method.

Step 2: Determine Slack Variables

Slack variables are additional variables that are introduced into the linear constraints of a linear program to transform them from inequality constraints to equality constraints. If the model is in standard form, the slack variables will always have a +1 coefficient. Slack variables are needed in the constraints to transform them into solvable equalities with one definite answer.

$$x_1 + 3x_2 + 2x_3 + \mathbf{s_1} = 10$$
$$x_1 + 5x_2 + x_3 + \mathbf{s_2} = 8$$
$$x_1, x_2, x_3, \mathbf{s_1}, \mathbf{s_2} \geq 0$$

After the slack variables are introduced, the tableau can be set up to check for optimality as described in Step 3.

$$Maximize : z = 8x_1 + 10x_2 + 7x_3$$
$$s.t. : x_1 + 3x_2 + 2x_3 + s_1 = 10$$
$$x_1 + 5x_2 + x_3 + s_2 = 8$$

Step 3: Setting up the Tableau

A Simplex tableau is used to perform row operations on the linear programming model as well as to check a solution for optimality. The tableau consists of the coefficient corresponding to the linear constraint variables and the coefficients of the objective function.

In the tableau below, the bolded top row of the tableau states what each column represents. The following two rows represent the linear constraint variable coefficients from the linear programming model, and the last row represents the objective function variable coefficients.

x1	**x2**	**x3**	**s1**	**s2**	**z**	**b**
1	3	2	1	0	0	10
1	5	1	0	1	0	8
-8	-10	-7	0	0	1	0

Once the tableau has been completed, the model can be checked for an optimal solution as shown in Step 4.

Step 4: Check Optimality

The optimal solution of a maximization of linear programming model is the values assigned to the variables in the objective function to give the largest zeta value. The optimal solution would exist on the corner points of the graph of the entire model. To check optimality using the tableau, all values in the last row must contain values greater than or equal to zero. If a value is less than zero, it means that variable has not reached its optimal value. As seen in the previous tableau, three negative values exist in the bottom row indicating that this solution is not optimal. If a tableau is not optimal, the next step is to identify the pivot variable to base a new tableau on, as described in Step 5.

Step 5: Identify Pivot Variable

The pivot variable is used in row operations to identify which variable will become the unit value and is a key factor in the conversion of the unit value. The pivot variable can be identified by looking at the bottom row of the tableau and the indicator. Assuming that the solution is not optimal, pick the smallest negative value in the bottom row. One of the values lying in the column of this value will be the pivot variable. To find the indicator, divide the beta values of the linear constraints by their corresponding values from the column containing the possible pivot variable. The intersection of the row with the smallest non-negative indicator and the smallest negative value in the bottom row will become the pivot variable.

In the example shown below, -10 is the smallest negative in the last row. This will designate the x_2 column to contain the pivot variable. Solving for the indicator gives us a value of $\frac{10}{3}$for the first constraint, and a value of $\frac{8}{5}$for the second constraint. Due to $\frac{8}{5}$ being the smallest non-negative indicator, the pivot value will be in the second row and have a value of 5.

x1	x2	x3	s1	s2	z	b	Indicator
1	3	2	1	0	0	10	10/3
1	5	1	0	1	0	8	8/5
-8	-10	-7	0	0	1	0	

Smallest Value

Now that the new pivot variable has been identified, the new tableau can be created in Step 6 to optimize the variable and find the new possible optimal solution.

Step 6: Create the New Tableau

The new tableau will be used to identify a new possible optimal solution. Now that the pivot variable has been identified in Step 5, row operations can be performed to optimize the pivot variable while keeping the rest of the tableau equivalent.

I. To optimize the pivot variable, it will need to be transformed into a unit value (value of 1). To transform the value, multiply the row containing the pivot variable by the reciprocal of the pivot value. In the example below, the pivot variable is originally 5, so multiply the entire row by $\frac{1}{5}$.

x1	x2	x3	s1	s2	z	b	
1/5	1	1/5	0	1/5	0	8/5	Pivot row

II. After the unit value has been determined, the other values in the column containing the unit value will become zero. This is because the x_2 in the second constraint is being optimized, which requires x_2 in the other equations to be zero.

x1	x2	x3	s1	s2	z	b
	0					
1/5	1	1/5	0	1/5	0	8/5
	0					

Pivot row

Pivot Column

III. In order to keep the tableau equivalent, the other variables not contained in the pivot column or pivot row must be calculated by using the new pivot values. For each new value, multiply the negative of the value in the old pivot column by the value in the new pivot row that corresponds to the value being calculated. Then add this to the old value from the old tableau to produce the new value for the new tableau. This step can be condensed into the equation on the next page:

New tableau value = (Negative value in old tableau pivot column) x (value in new tableau pivot row) + (Old tableau value)

Old Tableau:

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	5	1	0	1	0	8
-8	-10	-7	0	0	1	0

Old pivot column

New

x1	x2	x3	s1	s2	z	b
2/5	0	7/5	1	-3/5	0	26/5
1/5	1	1/5	0	1/5	0	8/5
-6	0	-5	0	2	1	16

New pivot row

Tableau:

Numerical examples are provided below to help explain this concept a little better.

Numerical examples:

I. *To find the s_2 value in row 1:*
New tableau value = (Negative value in old tableau pivot column) * (value in new tableau pivot row) + (Old tableau value)

New tableau value = (-3) * ($\frac{1}{5}$) + 0 = -$\frac{3}{5}$

II. *To find the x_1 variable in row 3:*
New tableau value = (Negative value in old tableau pivot column) * (value in new tableau pivot row) + (Old tableau value)

New value = (10) * ($\frac{1}{5}$) + -8 = -6

Once the new tableau has been completed, the model can be checked for an optimal solution.

Step 7: Check Optimality

As explained in Step 4, the optimal solution of maximization linear programming model are the values assigned to the variables in the objective function to give the largest zeta value. Optimality will need to be checked after each new tableau to see if a new pivot variable needs to be identified. A solution is considered optimal if all values in the bottom row are greater than or equal to zero. If all values are greater than or equal to zero, the solution is considered optimal and Steps 8 through 11 can be ignored. If negative values exist, the solution is still not optimal and a new pivot point will need to be determined which is demonstrated in Step 8.

Step 8: Identify New Pivot Variable

If the solution has been identified as not optimal, a new pivot variable will need to be determined. The pivot variable was introduced in Step 5 and is used in row operations to identify which variable will become the unit value and is a key factor in the conversion of the unit value. The pivot variable can be identified by the intersection of the row with the smallest non-negative indicator and the smallest negative value in the bottom row.

x1	x2	x3	s1	s2	z	b	Indicator
2/5	0	7/5	1	−3/5	0	26/5	(26/5) / (2/5) = 13
1/5	1	1	0	1/5	0	8/5	(8/5) / (1/5) = 8
-6	0	-5	0	2	1	0	

↑
Smallest Value

With the new pivot variable identified, the new tableau can be created in Step 9.

Step 9: Create New Tableau

After the new pivot variable has been identified, a new tableau will need to be created. Introduced in Step 6, the tableau is used to optimize the pivot variable while keeping the rest of the tableau equivalent.

I. Make the pivot variable 1 by multiplying the row containing the pivot variable by the reciprocal of the pivot value. In the tableau below, the pivot value was $\frac{1}{5}$, so everything is multiplied by 5.

x1	x2	x3	s1	s2	z	b
1	5	1	0	1	0	8

II. Next, make the other values in the column of the pivot variable zero. This is done by taking the negative of the old value in the pivot column and multiplying it by the new value in the pivot row. That value is then added to the old value that is being replaced.

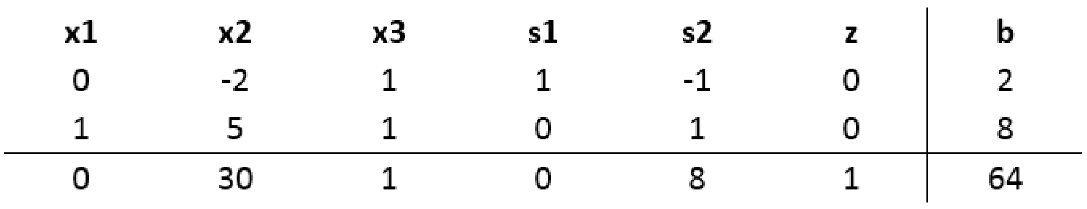
x1	x2	x3	s1	s2	z	b
0	-2	1	1	-1	0	2
1	5	1	0	1	0	8
0	30	1	0	8	1	64

Step 10: Check Optimality

Using the new tableau, check for optimality. Explained in Step 4, an optimal solution appears when all values in the bottom row are greater than or equal to zero. If all values are greater than or equal to zero, skip to Step 12 because optimality has been reached. If negative values still exist, repeat steps 8 and 9 until an optimal solution is obtained.

Step 11: Identify Optimal Values

Once the tableau is proven optimal the optimal values can be identified. These can be found by distinguishing the basic and non-basic variables. A basic variable can be classified to have a single 1 value in its column and the rest be all zeros. If a variable does not meet these criteria, it is considered non-basic. If a variable is non-basic it means the optimal solution of that variable is zero. If a variable is basic, the row that contains the 1 value will correspond to the beta value. The beta value will represent the optimal solution for the given variable.



x1	x2	x3	s1	s2	z	b
0	-2	1	1	-1	0	2
1	5	1	0	1	0	8
0	30	1	0	8	1	64

Basic variables: x_1, s_1, z
Non-basic variables: x_2, x_3, s_2

For the variable x_1, the 1 is found in the second row. This shows that the optimal x_1 value is found in the second row of the beta values, which is 8.

Variable s_1 has a 1 value in the first row, showing the optimal value to be 2 from the beta column. Due to s_1 being a slack variable, it is not actually included in the optimal solution since the variable is not contained in the objective function.

The zeta variable has a 1 in the last row. This shows that the maximum objective value will be 64 from the beta column.

The final solution shows each of the variables having values of:

x_1	= 8	s_1	= 2
x_2	= 0	s_2	= 0
x_3	= 0	z	= 64

The maximum optimal value is 64 and found at (8, 0, 0) of the objective function.

The Simplex method is an approach for determining the optimal value of a linear program by hand. The method produces an optimal solution to satisfy the given constraints and produce a maximum zeta value. To use the Simplex method, a given linear programming model needs to be in standard form, where slack variables can then be introduced. Using the tableau and pivot variables, an optimal solution can be reached. From the example worked throughout this document, it can be determined that the optimal objective value is 64 and can be found when $x_1=8$, $x_2=0$, and $x_3=0$.

1.5 Sensitivity analysis

Sensitivity analysis is the study of how the uncertainty in the output of a mathematical model or system (numerical or otherwise) can be apportioned to different sources of uncertainty in its inputs. A related practice is uncertainty analysis, which has a greater focus on uncertainty quantification and propagation of uncertainty; ideally, uncertainty and sensitivity analysis should be run in tandem.

The process of recalculating outcomes under alternative assumptions to determine the impact of a variable under sensitivity analysis can be useful for a range of purposes, including:

- Testing the robustness of the result of a model or system in the presence of uncertainty.
- Increased understanding of the relationships between input and output variables in a system or model.
- Uncertainty reduction, through the identification of model inputs that cause significant uncertainty in the output and should therefore be the focus of attention in order to increase robustness (perhaps by further research).
- Searching for errors in the model (by encountering unexpected relationships between inputs and outputs).
- Model simplification – fixing model inputs that have no effect on the output, or identifying and removing redundant parts of the model structure.
- Enhancing communication from modelers to decision makers (e.g. by making recommendations more credible, understandable, compelling or persuasive).
- Finding regions in the space of input factors for which the model output is either maximum or minimum or meets some optimum criterion (see optimization and Monte Carlo filtering).

- In case of calibrating models with large number of parameters, a primary sensitivity test can ease the calibration stage by focusing on the sensitive parameters. Not knowing the sensitivity of parameters can result in time being uselessly spent on non-sensitive ones.
- To seek to identify important connections between observations, model inputs, and predictions or forecasts, leading to the development of better models

Lecture note 2

2.1 Genetic algorithm

In computer science and operations research, a genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection.

A genetic algorithm makes uses of techniques inspired from evolutionary biology such as selection, mutation, inheritance and recombination to solve a problem. The most commonly employed method in genetic algorithms is to create a group of individuals randomly from a given population. The individuals thus formed are evaluated with the help of the evaluation function provided by the programmer. Individuals are then provided with a score which indirectly highlights the fitness to the given situation. The best two individuals are then used to create one or more offspring, after which random mutations are done on the offspring. Depending on the needs of the application, the procedure continues until an acceptable solution is derived or until a certain number of generations have passed.

A genetic algorithm differs from a classical, derivative-based, optimization algorithm in two ways:

A genetic algorithm generates a population of points in each iteration, whereas a classical algorithm generates a single point at each iteration.

A genetic algorithm selects the next population by computation using random number generators, whereas a classical algorithm selects the next point by deterministic computation.

Compared to traditional artificial intelligence, a genetic algorithm provides many advantages. It is more robust and is susceptible to breakdowns due to slight changes in inputs or due to the presence of noise. With respect to other optimization methods like praxis, linear programming, heuristic, first or breadth-first, a genetic algorithm can provide better and more significant results while searching large multi-modal state spaces, large state spaces or n-dimensional surfaces.

Genetic algorithms are widely used in many fields such as robotics, automotive design, optimized telecommunications routing, engineering design and computer-aided molecular design.

2.2 Methodology in Optimization problems

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a *generation*. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

Initialization

The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the *search space*). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

Selection

During each successive generation, a portion of the existing population is selected to breed a new generation. Individual solutions are selected through a *fitness-based* process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The fitness function is defined over the genetic representation and measures the *quality* of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The *fitness* of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise.

In some problems, it is hard or even impossible to define the fitness expression; in these cases, a simulation may be used to determine the fitness function value of a phenotype (e.g. computational fluid dynamics is used to determine the air resistance of a vehicle whose shape is encoded as the phenotype), or even interactive genetic algorithms are used.

Genetic operators

The next step is to generate a second generation population of solutions from those selected through a combination of genetic operators: crossover (also called recombination), and mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests that more than two "parents" generate higher quality chromosomes.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions. These less fit solutions ensure genetic diversity within the genetic pool of the parents and therefore ensure the genetic diversity of the subsequent generation of children.

Opinion is divided over the importance of crossover versus mutation. There are many references in Fogel (2006) that support the importance of mutation-based search.

Although crossover and mutation are known as the main genetic operators, it is possible to use other operators such as regrouping, colonization-extinction, or migration in genetic algorithms.

It is worth tuning parameters such as the mutation probability, crossover probability and population size to find reasonable settings for the problem class being worked on. A very small mutation rate may lead to genetic drift (which is non-ergodic in nature). A recombination rate that is too high may lead to premature convergence of the genetic algorithm. A mutation rate that is too high may lead to loss of good solutions, unless elitist selection is employed.

Heuristics

In addition to the main operators above, other heuristics may be employed to make the calculation faster or more robust. The *speciation* heuristic penalizes crossover between candidate solutions that are too similar; this encourages population diversity and helps prevent premature convergence to a less optimal solution.

Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

2.2.1 The building block hypothesis

Genetic algorithms are simple to implement, but their behaviour is difficult to understand. In particular it is difficult to understand why these algorithms frequently succeed at generating solutions of high fitness when applied to practical problems. The building block hypothesis (BBH) consists of:

A description of a heuristic that performs adaptation by identifying and recombining "building blocks", i.e. low order, low defining-length schemata with above average fitness.

A hypothesis that a genetic algorithm performs adaptation by implicitly and efficiently implementing this heuristics. Goldberg describes the heuristic as follows:

"Short, low order, and highly fit schemata are sampled, recombined [crossed over], and resampled to form strings of potentially higher fitness. In a way, by working with these particular schemata [the building blocks], we have reduced the complexity of our problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings.

"Because highly fit schemata of low defining length and low order play such an important role in the action of genetic algorithms, we have already given them a special name: building blocks. Just as a child creates magnificent fortresses through the arrangement of simple blocks of wood, so does a genetic algorithm seek near optimal performance through the juxtaposition of short, low-order, high-performance schemata, or building blocks."

Despite the lack of consensus regarding the validity of the building-block hypothesis, it has been consistently evaluated and used as reference throughout the years. Many estimation of distribution algorithms, for example, have been proposed in an attempt to provide an environment in which the hypothesis would hold. Although good results have been reported for some classes of problems, skepticism concerning the generality and/or practicality of the building-block hypothesis as an explanation for GAs efficiency still remains. Indeed, there is a reasonable amount of work that attempts to understand its limitations from the perspective of estimation of distribution algorithms.

2.3 Limitations of genetic algorithm

There are limitations of the use of a genetic algorithm compared to alternative optimization algorithms:

Repeated fitness function evaluation for complex problem is often the most prohibitive and limiting segment of artificial evolutionary algorithms. Finding the optimal solution to complex high-dimensional, multimodal problems often requires very expensive fitness function evaluations. In real world problems such as structural optimization problems, a single function evaluation may require several hours to several days of complete simulation. Typical optimization methods can not deal with such types of problem. In this case, it may be necessary to forgo an exact evaluation and use an approximated fitness that is computationally efficient. It is apparent that amalgamation of approximate models may be one of the most promising approaches to convincingly use GA to solve complex real life problems.

Genetic algorithms do not scale well with complexity. That is, where the number of elements which are exposed to mutation is large there is often an exponential increase in search space size. This makes it extremely difficult to use the technique on problems such as designing an engine, a house or plane. In order to make such problems tractable to evolutionary search, they must be broken down into the simplest representation possible. Hence we typically see evolutionary algorithms encoding designs for fan blades instead of engines, building shapes instead of detailed construction plans, and airfoils instead of whole aircraft designs. The second problem of complexity is the issue of how to protect parts that have evolved to represent good solutions from further destructive mutation, particularly when their fitness assessment requires them to combine well with other parts.

The "better" solution is only in comparison to other solutions. As a result, the stop criterion is not clear in every problem.

In many problems, GAs have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. This means that it does not "know how" to sacrifice short-term fitness to gain longer-term fitness. The likelihood of this occurring depends on the shape of the fitness landscape: certain problems may provide an easy ascent towards a global optimum, others may make it easier for the function to find the local optima. This problem may be alleviated by using a different fitness function, increasing the rate of mutation, or by using selection techniques that maintain a diverse population of solutions, although the No Free Lunch theorem proves that there is no general solution to this problem. A common technique to maintain diversity is to impose a "niche penalty", wherein, any group of individuals of sufficient similarity (niche radius) have a penalty added, which will reduce the representation of that group in subsequent generations, permitting other (less similar) individuals to be maintained in the population. This trick, however, may not be effective, depending on the landscape of the problem. Another possible technique would be to simply replace part of the population with randomly generated individuals, when most of the population is too similar to each other. Diversity is important in genetic algorithms (and genetic programming) because crossing over a homogeneous population does not yield new solutions. In evolution strategies and evolutionary programming, diversity is not essential because of a greater reliance on mutation.

Operating on dynamic data sets is difficult, as genomes begin to converge early on towards solutions which may no longer be valid for later data. Several methods have been proposed to remedy this by increasing genetic diversity somehow and preventing early convergence, either by increasing the probability of mutation when the solution quality drops (called triggered hypermutation), or by occasionally introducing entirely new, randomly generated elements into the gene pool (called random immigrants). Again, evolution strategies and evolutionary programming can be implemented with a so-called "comma strategy" in which parents are not maintained and new parents are selected only from offspring. This can be more effective on dynamic problems.

GAs cannot effectively solve problems in which the only fitness measure is a single right/wrong measure (like decision problems), as there is no way to converge on the solution (no hill to climb). In these cases, a random search may find a solution as quickly as a GA. However, if the situation allows the success/failure trial to be repeated giving (possibly) different results, then the ratio of successes to failures provides a suitable fitness measure.

For specific optimization problems and problem instances, other optimization algorithms may be more efficient than genetic algorithms in terms of speed of convergence. Alternative and complementary algorithms include evolution strategies, evolutionary programming, simulated annealing, Gaussian adaptation, hill climbing, and swarm intelligence (e.g.: ant colony optimization, particle swarm optimization) and methods based on integer linear programming. The suitability of genetic algorithms is dependent on the amount of knowledge of the problem; well known problems often have better, more specialized approaches.

2.4 Multi-objective optimization

Multi-objective optimization (also known as multi-objective programming, vector optimization, multicriteria optimization, multiattribute optimization or Pareto optimization) is an area of multiple criteria decision making, that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Multi-objective optimization has been applied in many fields of science, including engineering, economics and logistics where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Minimizing cost while maximizing comfort while buying a car, and maximizing performance whilst minimizing fuel consumption and emission of pollutants of a vehicle are examples of multi-objective optimization problems involving two and three objectives, respectively. In practical problems, there can be more than three objectives.

For a nontrivial multi-objective optimization problem, no single solution exists that simultaneously optimizes each objective. In that case, the objective functions are said to be conflicting, and there exists a (possibly infinite) number of Pareto optimal solutions. A solution is called nondominated, Pareto optimal, Pareto efficient or noninferior, if none of the objective functions can be improved in value without degrading some of the other objective values. Without additional subjective preference information, all Pareto optimal solutions are considered equally good (as vectors cannot be ordered completely). Researchers study multi-objective optimization problems from different viewpoints and, thus, there exist different solution philosophies and goals when setting and solving them. The goal may be to find a representative set of Pareto optimal solutions, and/or quantify the trade-offs in satisfying the different objectives, and/or finding a single solution that satisfies the subjective preferences of a human decision maker (DM)

A multi-objective optimization problem is an optimization problem that involves multiple objective functions. In mathematical terms, a multi-objective optimization problem can be formulated as

Min ($f_1(x_1), f_2(x_2), f_3(x_3), \dots . f_k(x)$)

Subject to x ∈ X,

Where the integer k ≥ 2 is the number of objectives and the set is the feasible set of decision vectors. The feasible set is typically defined by some constraint functions.

2.5 Applications of multi-objective optimization

2.5.1 Economics

In economics, many problems involve multiple objectives along with constraints on what combinations of those objectives are attainable. For example, consumer's demand for various goods is determined by the process of maximization of the utilities derived from those goods, subject to a constraint based on how much income is available to spend on those goods and on the prices of those goods. This constraint allows more of one good to be purchased only at the sacrifice of consuming less of another good; therefore, the various objectives (more consumption of each good is preferred) are in conflict with each other. A common method for analyzing such a problem is to use a graph of indifference curves, representing preferences, and a budget constraint, representing the trade-offs that the consumer is faced with.

Another example involves the production possibilities frontier, which specifies what combinations of various types of goods can be produced by a society with certain amounts of various resources. The frontier specifies the trade-offs that the society is faced with — if the society is fully utilizing its resources, more of one good can be produced only at the expense of producing less of another good. A society must then use some process to choose among the possibilities on the frontier.

Macroeconomic policy-making is a context requiring multi-objective optimization. Typically a central bank must choose a stance for monetary policy that balances competing objectives — low inflation, low unemployment, low balance of trade deficit, etc. To do this, the central bank uses a model of the economy that quantitatively describes the various causal linkages in the economy; it simulates the model repeatedly under various possible stances of monetary policy, in order to obtain a menu of possible predicted outcomes for the various variables of interest. Then in principle it can use an aggregate objective function to rate the alternative sets of predicted outcomes, although in practice central banks use a non-quantitative, judgement-based, process for ranking the alternatives and making the policy choice.

2.5.1 Finance

In finance, a common problem is to choose a portfolio when there are two conflicting objectives — the desire to have the expected value of portfolio returns be as high as possible, and the desire to have risk, often measured by the standard deviation of portfolio returns, be as low as possible. This problem is often represented by a graph in which the efficient frontier shows the best combinations of risk and expected return that are available, and in which indifference curves show the investor's preferences for various risk-expected return combinations. The problem of optimizing a function of the expected value (first moment) and the standard deviation (square root of the second central moment) of portfolio return is called a two-moment decision model.

2.5.2 Optimal control

In engineering and economics, many problems involve multiple objectives which are not describable as the-more-the-better or the-less-the-better; instead, there is an ideal target value for each objective, and the desire is to get as close as possible to the desired value of each objective. For example, energy systems typically have a trade-off between performance and cost or one might want to adjust a rocket's fuel usage and orientation so that it arrives both at a specified place and at a specified time; or one might want to conduct open market operations so that both the inflation rate and the unemployment rate are as close as possible to their desired values.

Often such problems are subject to linear equality constraints that prevent all objectives from being simultaneously perfectly met, especially when the number of controllable variables is less than the number of objectives and when the presence of random shocks generates uncertainty. Commonly a multi-objective quadratic objective function is used, with the cost associated with an objective rising quadratically with the distance of the objective from its ideal value. Since these problems typically involve adjusting the controlled variables at various points in time and/or evaluating the objectives at various points in time, intertemporal optimization techniques are employed.

2.5.3 Optimal design

Product and process design can be largely improved using modern modeling, simulation and optimization techniques.[citation needed] The key question in optimal design is the measure of what is good or desirable about a design. Before looking for optimal designs it is important to identify characteristics which contribute the most to the overall value of the design. A good design typically involves multiple criteria/objectives such as capital cost/investment, operating cost, profit, quality and/or recovery of the product, efficiency, process safety, operation time etc. Therefore, in practical applications, the performance of process and product design is often measured with respect to multiple objectives. These objectives typically are conflicting, i.e. achieving the optimal value for one objective requires some compromise on one or more of other objectives.

For example, when designing a paper mill, one can seek to decrease the amount of capital invested in a paper mill and enhance the quality of paper simultaneously. If the design of a paper mill is defined by large storage volumes and paper quality is defined by quality parameters, then the problem of optimal design of a paper mill can include objectives such as: i) minimization of expected variation of those quality parameter from their nominal values, ii) minimization of expected time of breaks and iii) minimization of investment cost of storage volumes. Here, maximum volume of towers are design variables. This example of optimal design of a paper mill is a simplification of the model used in. Multi-objective design optimization have also been implemented in engineering systems - e.g. design of nano-CMOS semiconductors, design of solar-powered irrigation systems, optimization of sand mould systems, engine design, optimal sensor deployment and optimal controller design.

2.5.4 Process optimization

Multi-objective optimization has been increasingly employed in chemical engineering. In 2009, Fiandaca and Fraga used the multi-objective genetic algorithm (MOGA) to optimize the pressure swing adsorption process (cyclic separation process). The design problem involved the dual maximization of nitrogen recovery and nitrogen purity. The results provided a good approximation of the Pareto frontier with acceptable trade-offs between the objectives.

In 2010, Sendín et al. solved a multi-objective problem for the thermal processing of food. They tackled two case studies (bi-objective and triple objective problems) with nonlinear dynamic models and used a hybrid approach consisting of the weighted Tchebycheff and the Normal Boundary Intersection approach. The novel hybrid approach was able to construct a Pareto optimal set for the thermal processing of foods.

In 2013, Ganesan et al. carried out the multi-objective optimization of the combined carbon dioxide reforming and partial-oxidation of methane. The objective functions were methane conversion, carbon monoxide selectivity and hydrogen to carbon monoxide ratio. Ganesan used the Normal Boundary Intersection (NBI) method in conjunction with two swarm-based techniques (Gravitational Search Algorithm (GSA) and Particle Swarm Optimization (PSO)) to tackle the problem. Applications involving chemical extraction and bioethanol production processes have posed similar multi-objective problems and were solved.

In 2013 Abakarov et al proposed an alternative technique to solve multi-objective optimization problems arising in food engineering. The Aggregating Functions Approach, the Adaptive Random Search Algorithm, and the Penalty Functions Approach were used to compute the initial set of the non-dominated or Pareto-optimal solutions. The Analytic Hierarchy Process and Tabular Method were used simultaneously for choosing the best alternative among the computed subset of non-dominated solutions for osmotic dehydration processes.

2.5.5 Radio resource management

The purpose of radio resource management is to satisfy the data rates that are requested by the users of a cellular network. The main resources are time intervals, frequency blocks, and transmit powers. Each user has its own objective function that, for example, can represent some combination of the data rate, latency, and energy efficiency. These objectives are conflicting since the frequency resources are very scarce, thus there is a need for tight spatial frequency reuse which causes immense inter-user interference if not properly controlled. Multi-user MIMO techniques are nowadays used to reduce the interference by adaptive precoding. The network operator would like to both bring great coverage and high data rates, thus the operator would like to find a Pareto optimal solution that balance the total network data throughput and the user fairness in an appropriate subjective manner.

Radio resource management is often solved by scalarization; that is, selection of a network utility function that tries to balance throughput and user fairness. The choice of utility function has a large impact on the computational complexity of the resulting single-objective optimization problem.[25For example, the common utility of weighted sum rate gives an NP-hard problem with a complexity that scales exponentially with the number of users, while the weighted max-min fairness utility results in a quasi-convex optimization problem with only a polynomial scaling with the number of users.

2.5.6 Electric power systems

Reconfiguration, by exchanging the functional links between the elements of the system, represents one of the most important measures which can improve the operational performance of a distribution system. The problem of optimization through the reconfiguration of a power distribution system, in terms of its definition, is a historical single objective problem with constraints. Since 1975, when Merlin and Back introduced the idea of distribution system reconfiguration for active power loss reduction, until nowadays, a lot of researchers have proposed diverse methods and algorithms to solve the reconfiguration problem as a single objective problem. Some authors have proposed Pareto optimality based approaches (including active power losses and reliability indices as objectives). For this purpose, different artificial intelligence based methods have been used: microgenetic, branch exchange, particle swarm optimization and non-dominated sorting genetic algorithm

References:

1. Loucks, D.P., Stedinger, J.R. and Haith, D.A. (1982) "Water Resources Systems Planning and Analysis", Prentice Hall Inc. N York
2. Chaturvedi, M.C. (1987), "Water Resources Systems Planning and Management", Tata McGraw Hill Pub. Co., N Delhi.
3. Hall. W.A. and Dracup, J.A. (1975), "Water Resources Systems", Tata McGraw Hill Pub. N Delhi
4. James, L.D. and Lee (1975), "Economics of Water Resources Planning", McGraw Hill Inc. n York
5. Kuiper, E. (1973) "Water Resources Development, Planning, Engineering and Economics", Buttersworth, London
6. Biswas, A.K. (1976) "Systems Approach to Water Management", McGraw Hill Inc. N York
7. Taha H A, (1996), "Operations Research", Prentice Hall of India, N Delhi.