

# OptimaLink: A MATLAB-Based Code for Teaching/Learning Precision-Point and Optimum Synthesis and Simulation of Mechanisms\*

AHMAD SMAILI, NAJI ATALLAH and FIRAS ZEINEDDINE

*Mechanical Engineering Department, American University of Beirut, Beirut, Lebanon.*

*E-mail: asmaili@aub.edu.lb*

*This paper presents OptimaLink, a MATLAB-based code to facilitate the teaching/learning of design of mechanisms and intelligent optimization methods. The current version of OptimaLink accommodates dimensional synthesis, analysis, and animation of four-bar (RRRR) mechanisms. Dimensional synthesis includes both precision-point and optimum synthesis methods. The paper begins by introducing the objective function for synthesis of the RRRR mechanism, followed by a summary of the intelligent search methods coded in OptimaLink, namely Simulated Annealing, Genetic Algorithm and Tabu Search. OptimaLink's structure and its use are then briefly outlined and an application example is provided to demonstrate the usefulness of the code. Finally, possible didactic uses of the code are proposed. Although dedicated to mechanism design, the user can tap into the optimization algorithms coded in OptimaLink to solve other design problems by coding the corresponding user-interface module.*

## INTRODUCTION

THE IMPORTANCE OF mechanisms derives from their extensive use in a myriad of applications such as home-tools, toys, automobiles, and machines. Every mechanical engineering program includes a course designed to introduce students to various types of mechanisms. The wide range of topics to be covered and the limited time available pose a challenge in terms of providing students with the necessary modern tools to analyze and synthesize mechanisms. As such, a simple code using a familiar programming environment would enhance course effectiveness and a better understanding of mechanism design. Additionally, it is becoming more important for students to learn how to apply optimization in design at an early stage.

Mechanism design can utilize general purpose computer-aided design programs such as ADAMS [1], Pro/E [2], AutoCAD [3], IDEAS [4], and Working Model [5] to design mechanisms. Although powerful, these programs require the user to have a good grasp of the principles of mechanism design, something students lack early on. Packages dedicated to mechanism design such as LINCAGES (Linkage Interactive Computer Analysis and Graphical Enhanced Synthesis Package) [6], SyMec [7], WATT by Heron Technologies [8], and SAM (Simulation and Analysis

of Mechanisms) by Artas [9] are also available. Specialized research packages have also emerged over the past 30 years. The latest is Synthetica [10]. This package is designed for the specification and analysis of serial and parallel chains and for synthesis of serial chain mechanisms. Recently, Cheng and Trang [11] introduced a web-based Ch (an embeddable C/C++ interpreter with extensions) mechanism toolkit for the analysis and design of mechanisms. Packages that are dedicated to mechanism design are limited to precision-position synthesis, mainly RRRR and slider-crank (RRRP) mechanisms, and cannot readily accommodate optimization.

OptimaLink has been developed to facilitate the teaching/learning of mechanism design and the application of intelligent optimization techniques using the simulation, graphics, control, and optimization capabilities of MATLAB. In addition to simulation and animation, OptimaLink provides for precision-position and optimum synthesis of RRRR mechanisms; it accommodates three- and four-precision position synthesis for motion, function, and path generation tasks using dyadic equations in complex number forms and the Burmester theory [12] and it provides for optimum synthesis if the task requires more than four positions. OptimaLink also allows the user to perform kinematic analysis of a synthesized linkage, to assess its performance in light of the design objective and to ensure that the mechanism is free from branching and order defects. This is essential, because the mathematical solution on which precision-point

\* Accepted 2 July 2005.

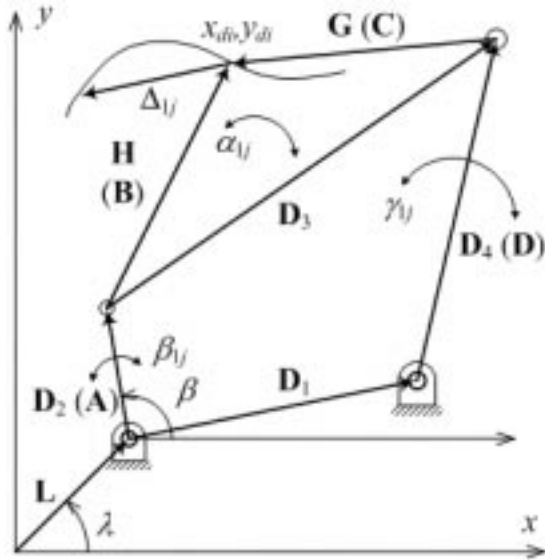


Fig. 1. A 4R linkage showing parameters referred to in the text.

synthesis is based does not always guarantee a workable mechanism. This is not a problem for an optimized mechanism, because the objective function includes constraints that reject a mechanism that does not conform to the motion requirements. OptimaLink implements several optimization methods, including the Weighted Least-Square technique, Simulated Annealing (SA), Genetic Algorithm (GA), and Tabu Search (TS) [9–12]. The user can choose to link the GA and TS methods with a gradient-based search to drive the solution further toward the global minimum. The optimization methods are coded in such a way that the user can tap into the modules to acquire an optimum solution to any design problem other than mechanism synthesis. All the user needs to do is construct a proper user interface for the design problem of interest.

MATLAB is the platform of choice for several reasons: (1) it is a code familiar to students and faculty, as it is already available and is used in many courses to solve a wide range of engineering problems, including control, communications, digital signal processing (DSP), optimization, etc.; (2) it has a vast library of functions dedicated to numerical computation; (3) it has an extensive graphics library; (4) it has Graphics User Interface capability; (5) it has the necessary hooks for communication with external devices; (6) it has an extensive set of tool boxes covering a wide spectrum of engineering problems such as optimization and control; (7) the code can be compiled to create an executable file that could run independently of MATLAB.

The following sections present the objective function and constraints for optimum synthesis of the RRRR mechanism followed by a brief summary of the optimization methods. The basic structure of OptimaLink is then introduced, its menus are explained, and didactic uses of the code are discussed. Finally, an example is provided to show

the capabilities of the code. The example synthesizes a RRRR mechanism for a ten-position path generation with prescribed timing task using Genetic Algorithm and Tabu Search with gradient methods.

## OPTIMUM SYNTHESIS OF THE 4R MECHANISM

A generic RRRR mechanism indicating the notations used hereafter is shown in Fig. 1. A boldface letter represents a vector quantity. A vector representing a link is indicated by an uppercase letter, and the lowercase counterpart denotes its length.  $\mathbf{D}_1$ – $\mathbf{D}_4$  are the position vectors representing the ground link, drive link, coupler link and follower link, respectively;  $\mathbf{H}$  and  $\mathbf{G}$  are the vectors representing the coupler link left and right sides, respectively; and  $\mathbf{L}$  is the position vector of the input pivot.

### The objective function

The objective function is developed to: (1) minimize the error between the desired trajectories and that generated by the synthesized mechanism; (2) satisfy Grashof's criteria to yield a crank-rocker mechanism if desired; (3) force the crank to rotate in one direction to eliminate order problems; (4) maintain an upper and lower bound on link lengths; and (5) reject solutions with branching problems. A solution that violates either condition (4) or (5) is rejected immediately. The first three conditions are satisfied by constructing the following objective function:

$$F(\mathbf{r}) = \sum_{i=1}^n [(x_{di} - x_{gi})^2 + (y_{di} - y_{gi})^2] + w_G \times noGrashof + w_O \times noOrder \quad (1)$$

$$\mathbf{r} = \{\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4, \mathbf{H}, \mathbf{G}\} \quad (2)$$

The parameters in equations (1) and (2) are as follows:  $\mathbf{r}$  is the vector of unknown mechanism parameters;  $x_{di}$ ,  $y_{di}$  and  $x_{gi}$ ,  $y_{gi}$  are the desired and the generated x- and y- coordinates of the desired path points; and  $w_C$  and  $w_G$  are normalization weights attached to violating order and Grashof's criteria, respectively. These weights reflect how important the conditions they represent are. In addition, they act as scaling factors to fix the order of magnitude of different variables in the objective function. The *noGrashof* and *noOrder* algorithms are expressed as follows:

$$\begin{aligned} &\text{if } (l_{\min} + l_{\max} > l_a + l_b) \\ &noGrashof = (l_{\min} + l_{\max} - l_a - l_b)^2 \\ &\text{if } (l_{\min} + l_b > l_a + l_{\max}) \\ &noGrashof = (l_{\min} + l_b - l_a - l_{\max})^2 \\ &\text{if } (l_{\min} + l_a > l_b + l_{\max}) \\ &noGrashof = (l_{\min} + l_a - l_b - l_{\max})^2 \end{aligned} \quad (3)$$

and

$$Direction = \text{sgn}(\beta_2 - \beta_1)$$

$$\Delta\beta_i = \beta_i - \beta_{i-1}; i = 2, \dots, n$$

$$noOrder = \frac{1}{4} \sum_2^n [(\text{sgn}(\Delta\beta_i) - Direction) \times \Delta\beta_i]^2 \quad (4)$$

$l_{min}$ ,  $l_{max}$ , and  $l_a$  and  $l_b$  are, respectively, the lengths of the shortest link, longest link, and the other two links of the mechanism.  $(\beta_i - \beta_{i-1})$  is the angular rotation of the drive link from position  $i-1$  to the position  $i$  and  $\text{sgn}(\beta_i - \beta_{i-1})$  represents the direction of rotation of the drive link. The objective function of equation (1) can be easily modified by the user to include mechanical advantage ( $MA = \text{Output force}/\text{Input force}$ ) specifications, affix a lower limit on the transmission angle, and other specification criteria as needed.

The user can define the limits of any variable (angles, link dimensions, etc.) or choose to hold any set of variables to specific values or to fall within a range of values, while requesting the code to optimize for the other variables. To accomplish this, the user is prompted to input the relevant information through two complementary steps: an input vector called *Modify* and an input window; the elements in both have a one-to-one correspondence. The inputs to the *Modify* vector are either zeros or nonzeros and the entries to the input window are values affixed to the associated parameters. An entry of 0 in the *Modify* vector commands the code to fix the value of the associated parameter to that provided in the input window. A nonzero entry to the *Modify* vector is used by the code as the upper limit of the associated parameter, while the lower limit is the value provided in the input window. Thus a solution that includes a link length beyond the desired range is automatically rejected. While this coding scheme increases computation time, it facilitates the use of the same objective function for many optimization possibilities. The code also provides for changing the number of iterations as the number of variables changes.

## OPTIMIZATION TECHNIQUES

The complex-number method and the Burmester theory are implemented in OptimaLink for precision-position synthesis of RRRR mechanisms. The theoretical treatment of these techniques is well documented in the literature and will not be addressed here [12]. However, the coded global optimization algorithms are briefly presented.

Although not guaranteed, global search algorithms have a much better chance of converging to the global minimum, depending on the problem in hand [13–16]. As stated earlier, SA, GA, and TS intelligent search methods have been coded in OptimaLink. The user can opt to link GA and

TS to a gradient algorithm toward the end of the search. This will enhance efficiency because, as the search gets closer to the global minimum, convergence becomes very slow. The gradient search takes the near global solution generated by GA or TS as an initial guess and further refines it, moving it ever closer to the global minimum. Due to space limitation, only a brief review of SA, TS, and GA will be presented. Interested readers can refer to [17–19] for details on the application of these search methods to mechanism design.

### Simulated Annealing

Simulated Annealing is a stochastic iterative optimization technique analogous to the annealing of metals. The flowchart in Fig. 2 shows the generic steps of the SA algorithm. If a probe is heated and then allowed to cool down slowly, it will reach a minimum energy state in which atoms rearrange themselves in a crystal lattice.

Temperature allows for random fluctuations of atoms and cooling will eventually lead to the lowest energy state. The cost function in an optimization problem represents the state of energy; it attains a minimum when the energy does also. Even if a system is close to a local minimum, any fluctuation may move the search to an even better minimum.

If a new configuration has lower energy (cost), it is accepted straight away. If its energy is higher, it will be accepted probabilistically according to:

$$p(\Delta E) = \exp(-\Delta E/T) \quad (5)$$

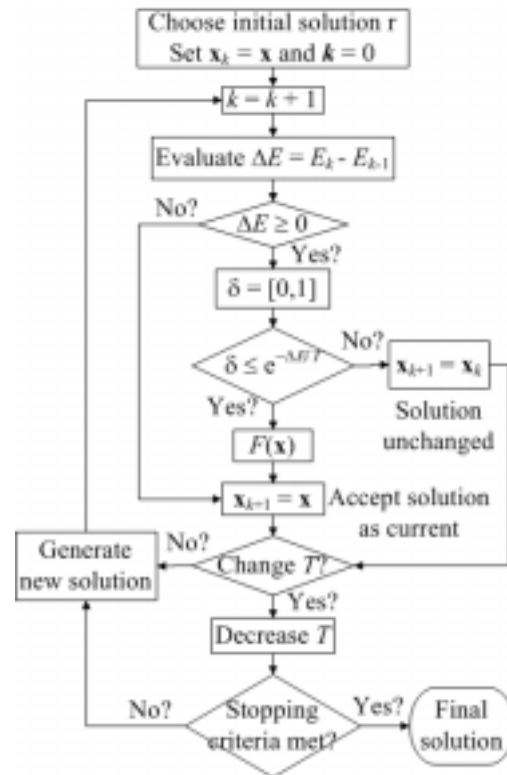


Fig. 2. Flowchart for implementing SA.

In equation (5),  $p$  is the Boltzmann probability (probability that the system will change its current energy state  $E$  by  $\Delta E$ ),  $\Delta E$  is the difference in energy (cost function) value between the current solution and the newly generated solution, and  $T$  is temperature. If  $\Delta E \geq 0$ , a random number  $\delta \in [0, 1]$  is generated from a uniform distribution. If  $\delta \leq p(\Delta E)$  (Metropolis criteria), then the newly generated solution is accepted as the current solution. If not, the current solution is unchanged. The statistical distribution of energies allows the system to escape local minima. The current code uses a geometric cooling schedule,  $T_{i+1} = c \times T_i$ . The default value for the temperature reduction factor  $c$  is 0.9, but it may be adjusted to suit the problem at hand. As the temperature cools down from its very hot initial value, the code continuously updates probabilistic ranges of variables by selecting them to be mainly in the high probability regions, although low probabilities are also selected to reduce the number of iterations to achieve a minimal acceptance rate. The range selection is made according to:

$$T_{new} = T_{current} + \Delta T(\delta - 0.5) \left[ \frac{T_{current}}{T_{melting}} \right]^{0.25} \quad (6)$$

$T_{new}$ ,  $T_{current}$ ,  $\Delta T$ ,  $\delta$ , and  $T_{melting}$ , are, respectively, the newly generated temperature, current temperature, allowed temperature range, random number in  $[0, 1]$ , and melting temperature.

#### Tabu Search

Tabu Search is an iterative dynamic neighborhood search technique characterized by intensive use of various adaptive memory strategies to search the solution space of a combinatorial optimization problem until a chosen termination criterion is satisfied. Depending on the type and complexity of the problem, TS employs short-term and/or long-term memories. The essential characteristics of a TS algorithm are summarized in the flowchart given in Fig. 3. Guided by a properly formed evaluation function, a crucial step in TS is to determine how to make a move from a current solution  $x$  to a solution  $x'$  in the neighborhood  $N(x)$ . Short-term memory is employed to prevent a move to a recently visited solution by maintaining a Tabu list of moves to most recently examined solutions. The placing of a move in the Tabu list is based on recency, frequency, quality, or influence [13–17].

Long-term memory can be explicit or attributive. Explicit memory is used to record elite solutions, promising unexplored neighbors of elite solutions and/or information about solution attributes that change during a move. Attributive memory is used to guide the search to unexplored, yet highly promising, regions. Explicit and attributive memories are also used, with intensification and diversification operators that help convergence. Intensification is applied to intensify the search in promising regions that have been

recorded in explicit memory. It is also applied to moves stored in attributive memory to further enhance the quality of the stored solution. Intensification is usually applied for a few iterations followed by applying a diversification strategy, which encourages the exploration of other regions during the search.

#### Genetic Algorithm

Genetic Algorithm can find global optimal solutions to non-linear multimodal functions. In an analogy with the principles of gene mechanics, individuals in a population, called phenotypes, are likened to chromosomes. GA does not deal directly with the parameters of the problem but rather with binary or real number codes representing those parameters. The fitness evaluation (or cost) function acts as the interface between the GA and the optimization problem.

To begin the search, an initial population of solutions is either randomly created or constructed of solutions derived from *a priori* knowledge about the given optimization problem. A population size of between 30 and 200 constitutes a good compromise [14]; a large population size increases computation time, but increases the probability of converging to a global solution. Genetic Algorithm subjects the phenotypes in a generation through many operators found in nature. The GA implemented in OptimaLink employs reproduction, single-point crossover, and mutation. Figure 4

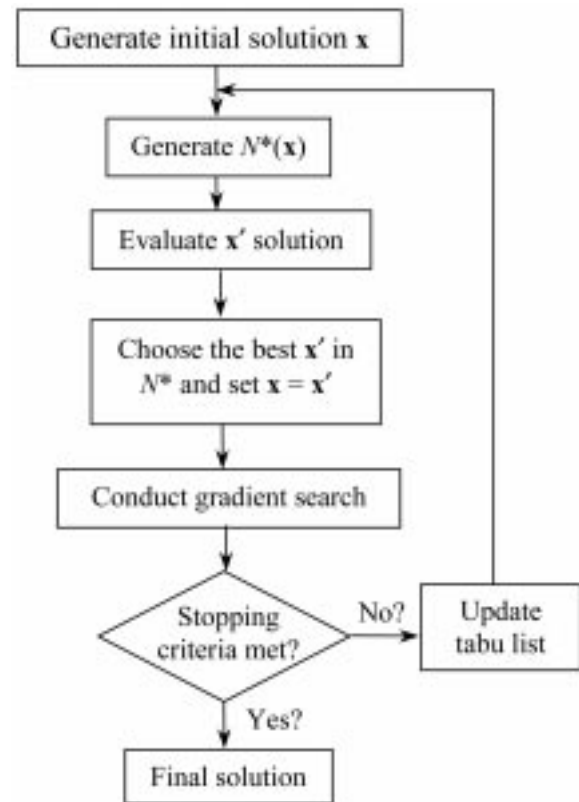


Fig. 3. Flowchart for the Tabu Search gradient algorithm.

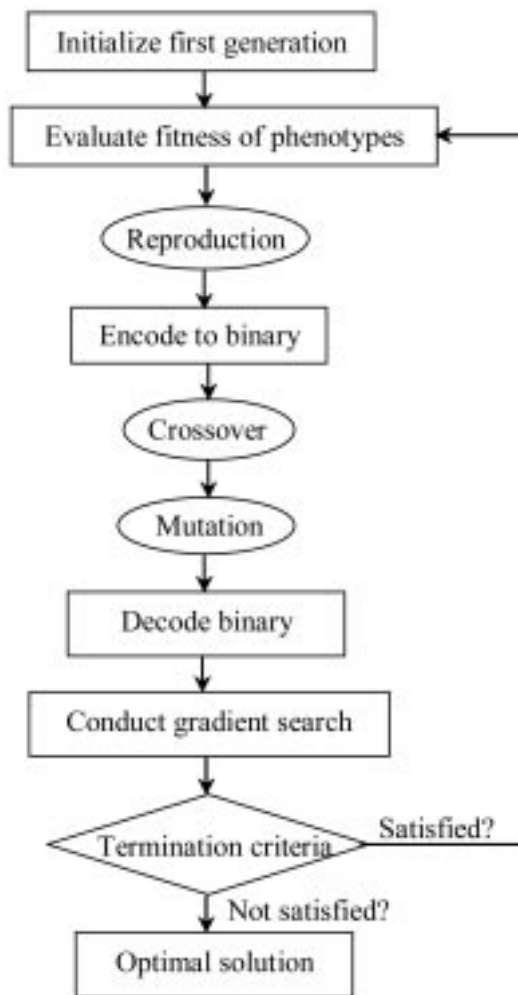


Fig. 4. Flowchart for the Genetic-Gradient Algorithm.

shows a flowchart of the GA linked to gradient search.

#### *Reproduction (selection)*

Reproduction is the process of passing a number of individuals in a generation unaltered to the next generation. The cost function determines the fitness of each phenotype to the required performance. A probability of reproduction, defined as the ratio of the phenotype's fitness to the sum of fitness values of all phenotypes in its generation, is assigned to each phenotype; a phenotype with higher probability has a higher chance of survival than a lower probability one. 'Roulette wheel' (or proportional) selection is herein employed to help steer the solution to a promising area while maintaining population diversity. The number of surviving phenotypes is equal to the number of phenotypes in a generation. Hence, several phenotypes whose fitness values are high will pass to the next generation more than once. Thus, more copies of these individuals than those whose fitness values are low are reproduced.

#### *Crossover*

The crossover operator is applied following reproduction to create two new children (individuals)

from two existing individuals (parents). Crossover mimics meiosis in biological systems; when two chromosomes undergo meiosis the two chromosomes are likely to split and swap parts at some point. The analogy in GA is that each phenotype is given a random probability of crossover. Phenotypes that will undergo this operator are paired together randomly. The location in the string where the two chromosomes split and swap parts is also determined randomly. An important control parameter is the frequency of crossover operation or crossover rate (CR); a low CR decreases the speed of convergence, whereas a high CR leads to saturation around one solution.

#### *Mutation*

Mutation is employed after crossover to force the algorithm to search new areas and ensure that the final solution is the global optimum. Mutation is a monadic operation where a child string is produced from a single parent string by flipping a random bit of a randomly chosen phenotype (being in binary beforehand), creating in the process a child of the original phenotype. Mutation provides a generation with unpredictable solutions that might lead the search to a new neglected area. Mutation rate (MR) is another important control parameter. A high MR introduces high diversity but may lead to instability, whereas a low MR finding a global optimal solution is difficult.

OptimaLink codes the phenotypes in binary after reproduction to ease crossover and mutation operations. The strings of characters are decoded back after mutation. If desired, the best phenotype in a generation is provided as an initial solution to a gradient-based search to drive the solution ever closer to the global minimum, improving the accuracy of the solution. The gradient search results are then used to assess their fulfillment of the termination criteria set by the code.

## OPTIMALINK STRUCTURE

OptimaLink is designed to allow any user to easily modify the code or add to it as desired. This explains the clear separation between three sequential components of the code: the user input/output (I/O) interface, the solver modules, and the simulator. The user interface is the set of menus, buttons, pop-up windows, shortcut keys and other I/O functions that interact with the user. The solver modules are either precision-point solvers or optimization solvers. The simulation option is made a separate entity to facilitate future improvements such as enhancing graphics refresh rates, choice of link shapes, choices of colors and surrounding plot, and other features.

From the operational standpoint, the software is divided into three levels: I/O, Data Transfer and Manipulation (DTM), and the Solver level (SOL). Each level is discussed briefly below. Figure 5

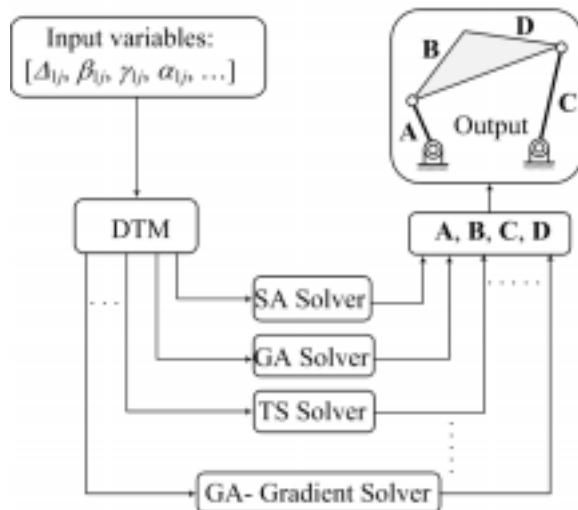


Fig. 5. OptimaLink structure.

shows a block diagram of the structure and operation of OptimaLink dedicated to RRRR mechanism design.

#### Input/Output (I/O)

The I/O is a critical component for OptimaLink to operate properly. Its role is twofold: (1) upon receiving a command, the I/O acquires the relevant data, checks its validity and alerts the user if the data provided is not adequate, and then transfers the data to the DTM level (discussed next), and (2) it displays the output results in a coherent and relevant manner.

To validate the input data, the solver option is first checked for the set of constraints and conditions it poses on the input. The code then assesses

the user input data and modifies obvious errors or accepts different inputs for the same set of options. In the case of ambiguity or erroneous inputs, the input module prompts the user and returns to the input window for re-entry of data.

The output is organized to present the results in an efficient and user-friendly format. A set of output variables that are common to all solvers are always provided. Outputs that are task specific, like Burmester curves, are dealt with separately, so as not to affect other operations that might follow. Further details on I/O are presented in the examples to follow, where I/O is represented in a broader context.

#### Data transfer and manipulation (DTM)

The DTM stage is not a separate module in the code, but rather bits and pieces of code that are distributed inside functions and subroutines. It represents the link between the user and the solver. Its role is to take the raw data provided by the user and organize it in a manner that suits the particular solver. In general, the input provided by the user is directly passed to the solver, but, in many cases, the solver requires data to be repartitioned into matrices or vectors differently. Some solvers require a different number of inputs. The DTM fills the missing inputs. In summary, the DTM performs two important tasks: (1) it re-formats input data according to the solver used, and (2) it adds missing data or options that are not transparent to the user.

#### Solvers (SOL)

Solvers are the MATLAB functions that implement the specific codes applied to synthesize a

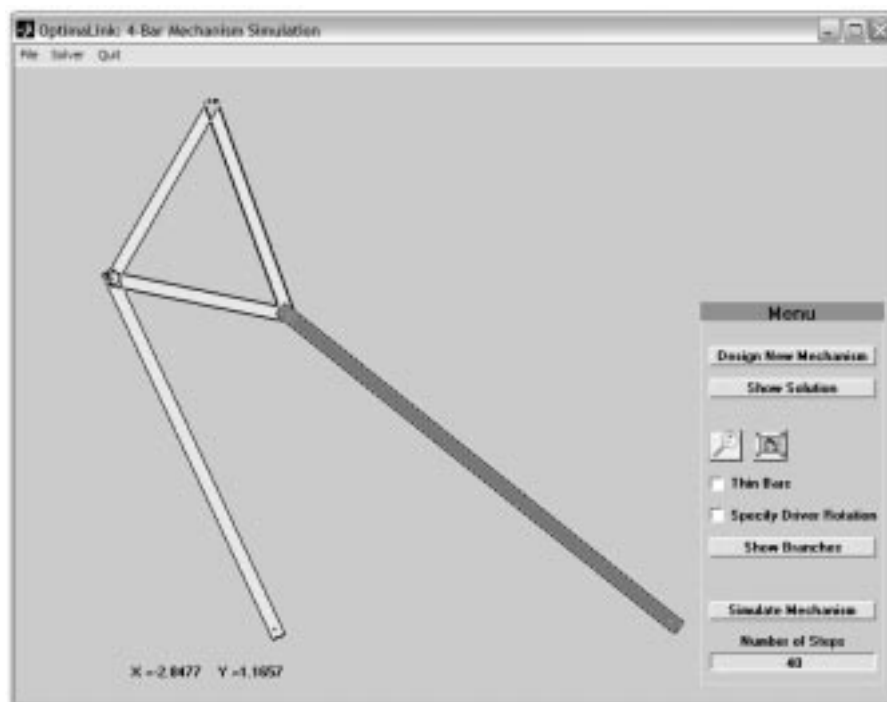


Fig. 6. Initial screen loaded upon launching OptimaLink.

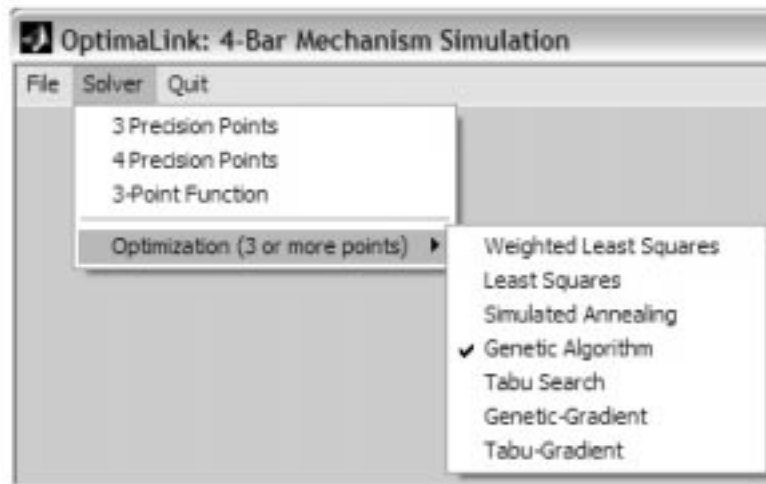


Fig. 7. Solver menu of OptimaLink (vectors shown correspond to four-point synthesis).

RRRR mechanism using the precision-position and optimization methods described earlier. Whenever a new feature is to be added to OptimaLink, the first step is to design the solver, test it, and use it as a function without the user interface shell to verify its integrity. Once verified, a DTM code is customized for the solver and a new input form associated with it is constructed. If the solver is tested again and the results do not match the desired results, the problem is more easily determined and limited to the pertinent I/O and DTM parts. This coding scheme results in robust and independent solvers. It facilitates the addition and building of a toolbox for 4R linkage design with minor additional effort.

#### Operating OptimaLink

Typing 'OptimaLink' at the MATLAB prompt launches OptimaLink and loads the default screen



Fig. 8. Typical OptimaLink input window (vectors shown correspond to four-point synthesis).

shown in Fig. 6. There are currently three user interface menus in OptimaLink: File, Solver and Quit. There are also shortcut keys: S to simulate mechanism, D to input a new design, and H for help. More will be added in future versions.

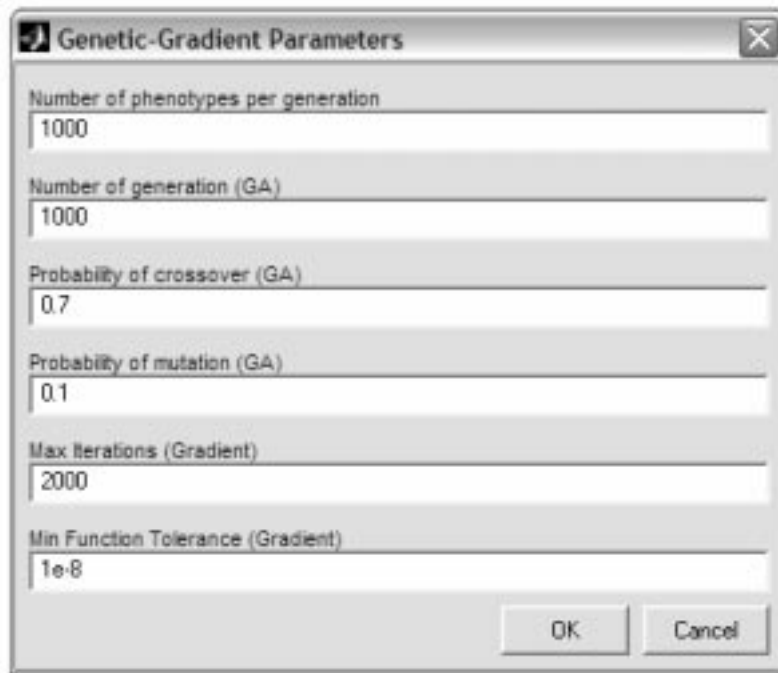
The File menu includes: the Save command to save the mechanism task parameters and the link dimensions for later use as a file in a '.mech' format; the Open command to load a saved file; and the Save As command to save a file with a different name.

The Solver menu shown in Fig. 7 allows the user to choose the desired synthesis method. Once a synthesis method is chosen, the user is prompted to enter the relevant parameters. For three-position, four-position, least-square, and SA synthesis, the code requests the user to provide the coupler point displacements  $\Delta_{ij}$ , the crank angular displacements  $\beta_{ij}$ , the follower angular displacements  $\gamma_{ij}$ , and finally the angular rotations of the coupler link  $\alpha_{ij}$ , as the mechanism moves from the first position to the  $j$ -th position ( $j = 2, 3, \dots, n$ ). These increments are entered as  $1 \times n$  MATLAB vectors. The input window used to enter the parameters for a four-precision-point synthesis is shown in Fig. 8. For Weighted Least Square optimization, an additional weights vector is required and is entered in a separate window. A synthesis task always results in the first position of the mechanism.

Once the task parameters are entered, the user is prompted to provide optimization-specific parameters. Figures 9 and 10 show the windows to enter the parameters required for the GA-Gradient and TS-Gradient search methods.

#### APPLICATION EXAMPLE

Tabu Search and Genetic Algorithm with and without gradient search are applied to the optimum synthesis of a RRRR mechanism. The task is to move a coupler point along ten positions in coordination with crank rotations. The task



**Genetic-Gradient Parameters**

Number of phenotypes per generation  
1000

Number of generation (GA)  
1000

Probability of crossover (GA)  
0.7

Probability of mutation (GA)  
0.1

Max iterations (Gradient)  
2000

Min Function Tolerance (Gradient)  
1e-8

OK Cancel

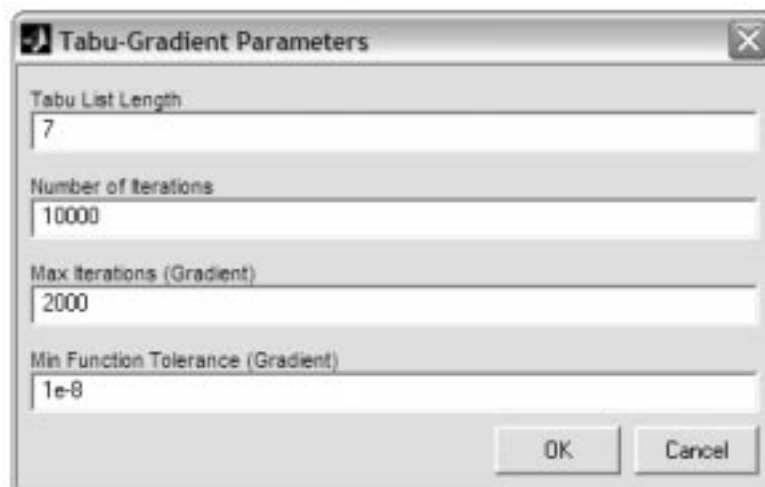
Fig. 9. OptimaLink input window for GA-Gradient parameters.

parameters are given in Table 1. To further the accuracy of the results, the solution obtained from Tabu Search or Genetic Algorithm is applied as the initial solution to a gradient-based search that drives the solution closer to the global optimum. The results of the searches are given in Table 2. Note that the link vectors are expressed in complex notations. The table compares the solutions obtained using four different algorithms: Tabu Search, Tabu-Gradient, Genetic Algorithm, and Genetic-Gradient.

Tabu and Tabu-Gradient searches were conducted using a recency based Tabu list of six entries while searching all the non-Tabu neighborhood of a solution in each iteration. The maximum number of iterations used for Tabu Search was

84,000, taking about 900 seconds of run time on a Pentium-III PC. Meanwhile, with a Tabu-Gradient search solution, the number of iterations was dramatically reduced to  $\sim 318$  iterations, requiring a run time of  $\sim 180$  seconds. The average root-square (RS) error between the desired and generated coupler point positions is 0.183 for TS and 0.167 for TS-Gradient search solutions.

Genetic Algorithm and Genetic-Gradient searches were conducted using 1000 phenotypes per generation and probabilities of crossover and mutation of 0.95 and 0.05, respectively. The GA solution was found in 765 seconds in the 964th generation, whereas the GA-Gradient solution was found in 63.9 seconds in the 38th generation. The average RS error between the desired and



**Tabu-Gradient Parameters**

Tabu List Length  
7

Number of iterations  
10000

Max iterations (Gradient)  
2000

Min Function Tolerance (Gradient)  
1e-8

OK Cancel

Fig. 10. OptimaLink input window for Tabu-Gradient parameters.



Table 1. Desired trajectory points

Point	$x_{di}$	$y_{di}$	$\Delta\gamma_i$	Point	$x_{di}$	$y_{di}$	$\Delta\gamma_i$
1	1	1.5	0°	6	3.5	2	175°
2	1.5	2	35°	7	4	1.5	192.5°
3	2	2	87.5°	8	3.5	1	262.5°
4	2.5	2	105°	9	2.5	1	280°
5	3	2	140°	10	1.5	1	315°

Table 2. Solutions obtained from four algorithms

		TS	TS-Gradient	GA	GA-Gradient
Parameter	$D_1$	$0.154 - 2.162i$	$-0.523 - 3.107i$	$-3.820 - 2.459i$	$-0.615 - 1.230i$
	$D_2$ (A)	$0.575 + 0i$	$0.577 + 0i$	$0.387 + 0i$	$0.584 + 0i$
	$D_3$	$2.092 - 2.142i$	$3.910 - 3.106i$	$2.995 - 1.524i$	$2.982 - 1.229i$
	$D_4$ (D)	$2.513 + 0.020i$	$5.010 + 0.001i$	$7.202 + 0.935i$	$4.181 + 0.001i$
	H (B)	$-1.496 - 7.155i$	$-2.014 - 8.167i$	$-1.196 - 6.982i$	$-2.441 - 7.831i$
	G (C)	$-3.588 - 5.013i$	$-5.924 - 5.061i$	$-4.191 - 5.458i$	$-5.423 - 6.602i$
	L	$1.951 + 8.745i$	$2.014 + 9.242i$	$2.7580 + 8.2320i$	$1.964 + 10.000i$
RS error		0.183	0.167	0.303	0.167

generated coupler point positions is .303 and 0.167 for GA and GA-Gradient search solutions, respectively.

The mechanism with the least RS error is the one produced by a Genetic-Gradient search (shown in Fig. 11). The closed coupler curve is the one generated by the synthesized mechanism, shown at the first desired position.

### DIDACTIC USE OF OPTIMALINK

OptimaLink provides a computer-aided instructional tool to enhance the teaching/learning of analysis and synthesis of mechanisms. In a kinematics course, the instructor can easily use OptimaLink to demonstrate synthesis techniques discussed in lectures. Answers to ‘What-If’ ques-

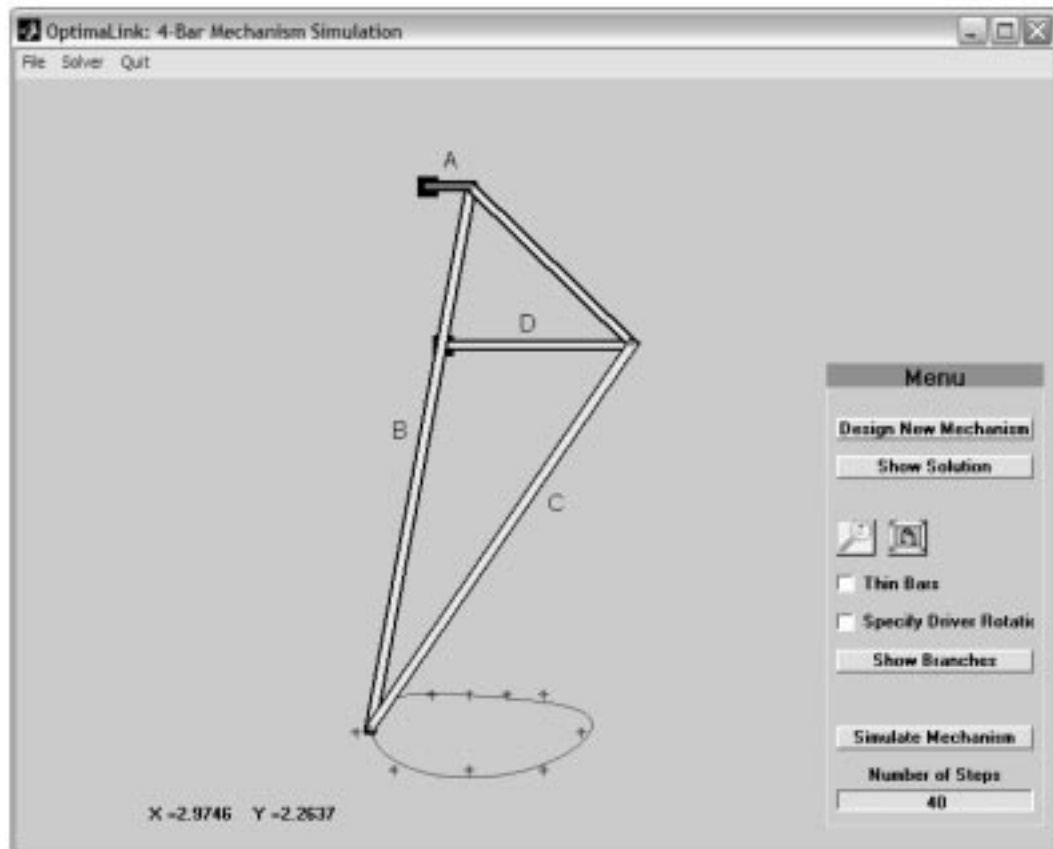


Fig. 11. The Genetic-Gradient synthesized mechanism showing the desired points and the generated coupler curve.

tions on mechanism design issues can be easily generated to enhance students' understanding of the subject and allow them to explore their creativity in many ways. For example, the instructor may pose questions like: How does slightly changing the dimension of a link affect the mechanism's performance? or How does a synthesized mechanism change if any of the task parameters is modified to accommodate a new design requirement? And so on. After students have reflected on these questions, the instructor can use OptimaLink to easily implement design changes and quickly show the results. The code can also be made available to students to solve homework problems and investigate various design possibilities.

The optimization algorithms in OptimaLink can be used to teach optimization through mechanism design. The same methods can also be used to generate optimum solutions to other design problems. To that end, the user can construct a MATLAB module that implements the objective function and relevant constraints of the design problem of interest (e.g. gear train, truss, etc.) and provide the interface with the algorithm of choice.

## CONCLUSIONS

This paper presented OptimaLink, a MATLAB-based code for the teaching/learning of analysis and dimensional synthesis of RRRR mechanisms. The code accommodates precision-point synthesis using the complex-number method and the Burmeister theory, and intelligent optimum synthesis methods using the Simulated Annealing, Genetic Algorithm, and Tabu Search optimization techniques, in addition to the Least Square, Weighted Least Square, and Gradient methods. The objective function and constraints for optimum synthesis of the RRRR mechanism were introduced and a brief summary of the optimization methods was presented. The basic structure and menus of OptimaLink were then explained and a ten-position synthesis example was provided. Finally, didactic uses of the code were discussed. OptimaLink will evolve ultimately to become an educational package for the analysis, synthesis and control of various mechanical systems including linkages, cam-followers, gear drives, screw drives, tendon drives, and Geneva wheels.

## REFERENCES

1. Mechanical Dynamics, <http://www.adams.com/>.
2. Parametric Technology Corporation, <http://www.ptc.com/>.
3. AutoDesk Inc., <http://www.autodesk.com/>.
4. Structural Dynamics Research Corporation, <http://www.sdrc.com/>.
5. Knowledge Revolution, <http://www.krev.com/>.
6. N. Yu, A. Erdman and B. Byers, LINCAGES 2000: Latest development and case study, *Proc. ASME Design Engineering Technical Conferences*, DETC2002/MECH-34375 (2002).
7. SyMech Inc, <http://www.symech.com/>.
8. Heron Technologies, <http://www.heron-technologies.com/>.
9. ARTAS, <http://www.artas.nl/>.
10. A. Perez, H. J. Su and M. McCarthy, Synthetica 2.0: Software for the synthesis of constrained serial chains, *Proc. ASME Design Engineering Technical Conferences*, DETC2004-57524 (2004).
11. H. H. Cheng and D. T. Trang, Web-based mechanism design and analysis, *Proc. ASME Design Engineering Technical Conferences*, DETC2004-57594 (2004).
12. G. Sandor and A. Erdman, *Design of Mechanism*, Vol. II, Prentice-Hall (1984).
13. P. M. Pardalos and M. G. C. Resende, *Handbook of Applied Optimization*, Oxford University Press (2002).
14. D. T. Pham and D. Karaboga, *Intelligent Optimization Techniques*, Springer (2000).
15. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipe, the Art of Scientific Computing*, Cambridge University Press (1989).
16. F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers (1997).
17. I. Ulah and S. Kota, Optimal synthesis of mechanisms for path generation using Fourier descriptors and global search methods, *ASME J. Mechanical Design*, **119** (1997), pp. 504–510.
18. Kunjur and S. Krishnamurty, Genetic algorithms in mechanism synthesis, *J. Applied Mechanisms and Robotics*, **4**(2) (1997), pp. 8–24.
19. A. Smaili and N. Attalah, A Tabu search-based optimization technique for synthesis of mechanisms, *Proc. ASME DETC 2004*, 96-DETC2004-57420 (2004).

**Ahmad Smaili** has served on the Mechanical Engineering Faculties at Mississippi State University (1987–1991) and Tennessee Technological University (1991–1999). Currently, he is an Associate Professor of Mechanical Engineering at the American University of Beirut. His teaching interests are in the area of design and mechatronics. His research interests are in the areas of mechanisms, vibration control, and robotics.

**Naji Atallah** graduated from the American University of Beirut with a Bachelor of Engineering degree in 2002 and a Master's degree in 2004, both in Mechanical Engineering.

Currently, he is a design engineer at Petrofac International Limited, UAE. His research interests are in the areas of mechanisms, mechatronics, and heuristic optimization techniques.

**Firas Zeineddine** graduated from the American University of Beirut with a Bachelor of Engineering degree in 2001 and Master's degree in 2003, both in Mechanical Engineering. Currently, he is a design engineer at Extreme Pops, Lebanon. His research interests are in the areas of mechanisms, mechatronics, and computer vision.