OPTIMIZATION OF BACKHOE-LOADER MECHANISMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

LEVENT İPEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

AUGUST 2006

Approval of the Graduate School of Natural And Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

_____

Prof. Dr. Kemal İder
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

_____

Prof. Dr. Eres Söylemez
Supervisor

Examining Committee Members

Prof. Dr. Mehmet Çalışkan	(METU, ME)	_____
Prof. Dr. Eres Söylemez	(METU, ME)	_____
Prof. Dr. Reşit Soylu	(METU, ME)	_____
Prof. Dr. M. Kemal Özgören	(METU, ME)	_____
Prof. Dr. M. Polat Saka	(METU, ES)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Levent İPEK

Signature :

ABSTRACT

OPTIMIZATION OF BACKHOE-LOADER MECHANISMS

İpek, Levent
M.S., Department of Mechanical Engineering
Supervisor: Prof. Dr. Eres Söylemez

August 2006, 71 pages

This study aims to develop a computer program to optimize the performance of loader mechanisms in backhoe-loaders. The complexity and the constraints imposed on the loader mechanism does not permit the use of classical optimization techniques used in the synthesis of mechanisms. Genetic algorithm is used to determine the values of the design parameters of the mechanism while satisfying the constraints and trying to maximize breakout forces that the machine can generate.

Keywords: Genetic Algorithm, Mechanism Optimization, Backhoe-Loader Mechanism

ÖZ

KAZICI-YÜKLEYİCİ MEKANİZMALARININ OPTİMİZASYONU

İpek, Levent
Yüksek lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Eres Söylemez

Ağustos 2006, 71 sayfa

Bu çalışmanın amacı kazıcı-yükleyici mekanizmalarının yükleyici performansını optimize etmek üzere bir bilgisayar programı geliştirmektir. Yükleyici mekanizması karmaşık olduğundan ve mekanizmanın uzuv boyutları ile yerleştirilmesinde çeşitli sınırlamalar olduğundan mekanizmaların sentezinde kullanılan optimizasyon yöntemlerini uygulamak zordur. Mekanizma tasarım parametrelerinin değerlerini belirlemek için Genetik Algoritma kullanılmıştır. Mekanizma üzerinde bazı sınırlamalar sağlanmaya çalışılırken aynı zamanda makinanın kol ve kova koparma kuvvetleri artırılmaya çalışılmıştır.

Anahtar Kelimeler: Genetik Algoritma, Mekanizma Optimizasyonu, Kazıcı-Yükleyici Mekanizması

ACKNOWLEDGMENTS

The author expresses sincere appreciation to Prof. Dr. Eres Söylemez for his guidance and insight throughout the research.

The author would also like to thank to his office mates Alper Yalçınkaya, Ayhun Ünal, Cevdet Can Uzer, Ferhan Fıçıcı, Mehmet Yener, Onursal Önen and Taner Karagöz for their support at different stages of this thesis and for their willingness to answer to his endless questions.

To My Family

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Backhoe-Loader is an earth-moving machine equipped with mechanisms to guide its working attachments. It has a backhoe at the rear and a loader mechanism in the front, which guides the bucket for loading materials. It is desired to guide this bucket so that it can dig by a certain amount into the ground, lift above ground level to dump on a truck. It also has to reach forward by a certain amount so that it can dump clear of the truck. Besides these kinematical challenges, it also has to have as much lifting and breakout force as possible to work easily on heavy loads.

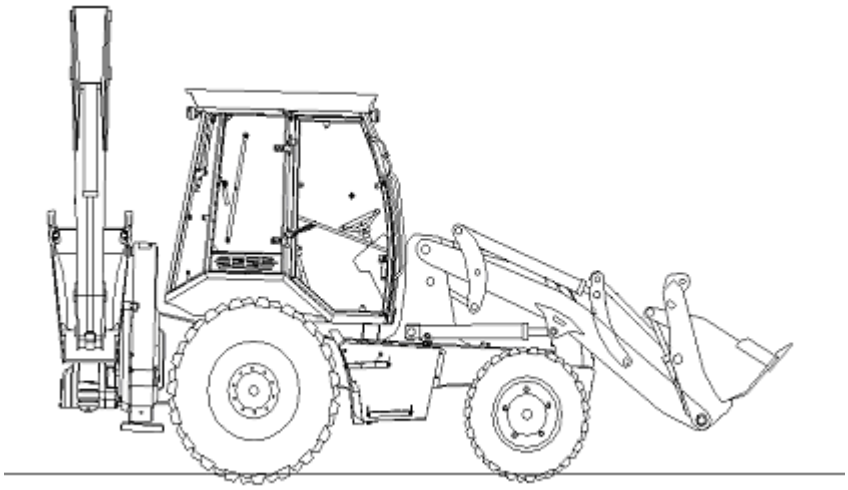

Figure 1.1 – Backhoe-Loader Machine

Loader mechanism is a complex mechanism with 11 linkages and two degrees of freedom. Moreover, there are constraints on the mechanism to be satisfied. They can be listed as:

- Dump height
- Digging depth
- Location and interference of joints

Dump height is the maximum height that the bucket can reach and digging depth is the maximum depth that the bucket can level below ground as shown in Figure 1.2. A minimum value of dump height and digging depth must be achieved. Also, joints must not be too close to each other and linkages should be able to move clear of each other. There are three joints that mount the mechanism to machine chassis and these joints have to be placed at the front pole of the chassis.

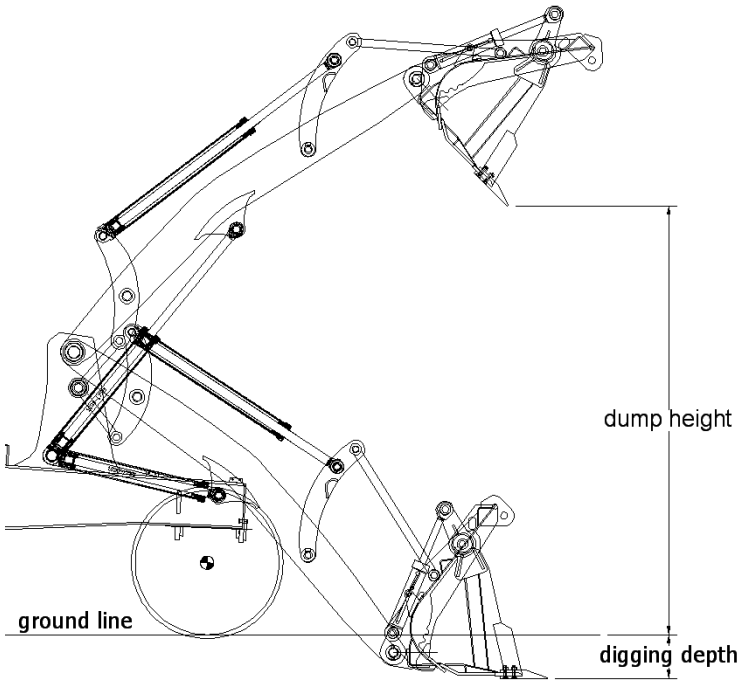

Figure 1.2 – Dump height and digging depth

While maintaining these constraints, bucket and arm breakout forces and lifting capacity of the mechanism are to be maximized.

Bucket and arm breakout forces and lift capacity are defined by SAE (Society of Automotive Engineers) in the SAE J732 standard along with other specification definitions for loaders [23].

Arm breakout force in newtons, as defined in SAE J732, is the maximum sustained vertical upward force exerted 100 mm behind the tip of the bucket and is achieved by applying maximum available pump pressure to arm hydraulic cylinder (lift cylinder) while not exceeding the allowable system pressure in any other circuits of the hydraulic system. Positioning of the loader shall be such that cutting edge of the bucket will be parallel to the ground line and its height with respect to ground level will be in the range ± 20 mm as shown in Figure 1.3.

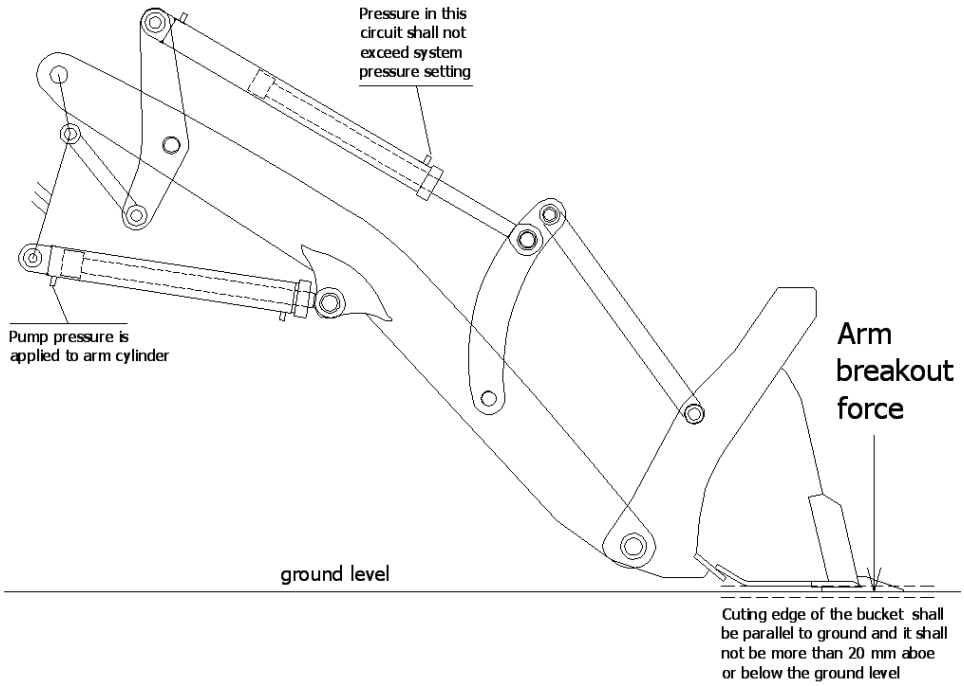

Figure 1.3 – Description of arm breakout force

Bucket breakout force in newtons is the maximum sustained vertical upward force exerted 100 mm behind the tip of the bucket and is achieved by applying maximum available pump pressure to bucket hydraulic cylinder. Positioning of the loader is same as the arm breakout case except arm shall be supported at the joint where bucket is attached. So there isn't any pressure developing in the arm cylinder.

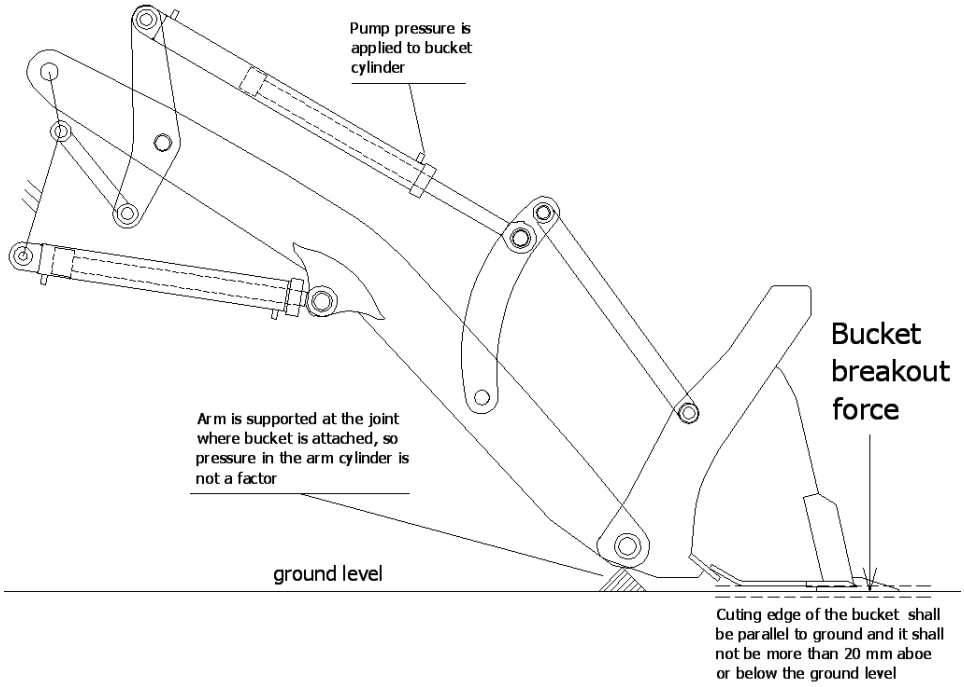

Figure 1.4 – Description of bucket breakout force

Lift capacity is defined as the maximum load in newtons that can be lifted to maximum height as shown in Figure 1.5. Lifting is achieved with arm hydraulic cylinder and pressure at any other circuit shall not exceed system pressure setting. During lifting, bucket hydraulic cylinder shall be at its minimum length.

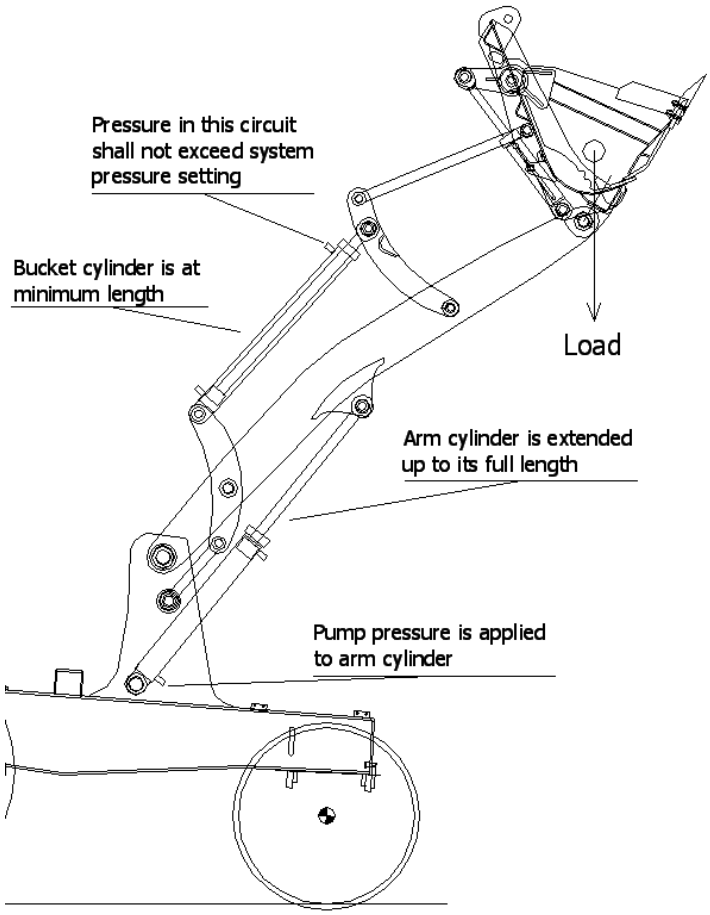

Figure 1.5 – Lift capacity

The aim of this study is to implement a computer program and then increase said breakout forces and lift capacity by altering mechanism design parameter values using the computer program implemented.

With too many parameters of the mechanism and constraints, analytical methods are not practical to implement on this problem. However, using heuristic methods such as Genetic Algorithm it is possible to improve a preliminary design. Advantage of Genetic Algorithm lies in that it does not need to have insight about the problem itself, it only has to know whether the solution found is better or worse than a

previously available solution. So only requirement is to build a fitness function, which will assign fitness values to the mechanisms generated by Genetic Algorithm. Fitness function will calculate all required outputs of the mechanism and combine them into a single fitness value with some weighting factors. Therefore, its necessary to conduct position and force analyses on the mechanism.

In this study a Genetic Algorithm is developed and is implemented in Microsoft Excel ® [1]. Analysis of the mechanism is conducted in Excel sheets and its results are processed by Genetic Algorithm, which is coded in VBA (Visual Basic for Applications) as Excel macros.

Next chapter of this thesis will investigate optimization methods for mechanisms in general and further it will lay examples of works on heuristics methods and Genetic Algorithms used in mechanism optimization. Third chapter describes the working mechanisms of the Genetic Algorithm used in this work and describes its implementation. Forth chapter explains the analysis of the mechanism. Position and force analyses are conducted and necessary outputs are calculated for fitness function. In the fifth chapter a case study is given. Starting off with an initial solution, a better mechanism is tried to be found. Results and findings of this work are discussed in the last chapter.

[1] Excel is a trademark of Microsoft Corporation

CHAPTER 2

LITERATURE SURVEY

Mechanism optimization can be achieved by various approaches. Each method has its own advantages and drawbacks. While analytical methods are more precise in some cases, they may be hard to implement on complex problems. Heuristic methods are more preferable on complex problems because of their easy implementation and robustness.

A general formulation for optimization problem can be stated as finding a vector of parameters such as [22]:

$$x = (x_1, x_2, ..., x_n)$$

to minimize or maximize an objective function

$$f(x) = f(x_1, x_2, ..., x_n)$$

subject to equality constraints

$$h_j(x) \equiv h_j(x_1, x_2, ..., x_n) = 0 \qquad \text{j=1 to p}$$

and inequality constraints

$$g_i(x) = g_i(x_1, x_2, ..., x_n) \leq 0 \qquad \text{i=1 to m}$$

Different mechanism optimization methods are presented with the following sections.

2.1 GRAPHICAL AND ANALYTICAL METHODS

Analytical methods are among the first methods used in mechanism synthesis. Solution procedure usually involves preparation of an analytical formulation of the problem and then it is solved to yield proportions of the linkages. Though they give accurate results for the selected precision points, one cannot control precision of other points. To overcome this problem, there are methods for selecting precision points to give better overall precision, but still one cannot solve complex mechanism synthesis problems with analytical methods. To achieve optimization, other constraints are implemented to the solution formulation. This will reduce the number of free parameters. Hence, it will reduce the number of alternatives. For complex problems adding constraints will complicate the problem even more.

Graphical methods are also one of the first techniques used in synthesis of mechanisms. They are similar to analytical methods considering precision. Only limited number of precision points may be selected. Also they have the same drawbacks with analytical methods considering complex problems. In complex problems there may be many design parameters and their effects on the outputs depend on the values of other parameters. Analytical methods get very complicated considering this kind of non-linear complex design problems [5].

2.2 CASE-BASED DESIGN

It is possible to store solutions for predefined problems and use them later to solve other problems that show similarity to the original stored problem. Later modifications and adaptations to the result may be applied if necessary. This method is known as Case-based Design.

Ashim Bose, Maria Gini and Donald Riley worked on a Case-based design method to design four-bar linkages [7]. They developed a method to store solutions to four-bar linkages and design four-bar linkages using these stored cases with incomplete specifications. Matched case is later adapted to give a better result for the problem. They have listed a few problems with this method such as inability to match to a case when the desired coupler curve is not smooth. Possible solution offered for this problem is smoothing the desired curve before matching process. Although they have reached satisfactory results with four-bar linkages, with increasing number of linkages, matching will be more problematic and will require a greater database of cases and may render this method impractical.

2.3 STATISTICAL APPROACH

Kunjur and Krishnamurty worked on an optimization method that employs ANOVA (Analysis Of Variance between groups) to investigate effect of design parameters relative to each other [8]. They developed a robust multi-criteria optimization method based on ANOVA results. They used Taguchi's method as a starting point. Taguchi's method is based on keeping solutions that improve at least one of the design criteria while not degrading others. Main problem of this method is expressed as difficulty in finding all these "non-inferior" sets. They developed Taguchi's method to overcome this problem and applied it to two case problems one of which is a four-bar mechanism's coupler path optimization problem.

2.4 GRADIENT SEARCH METHODS

Gradient-based search methods utilize gradient information to drive a solution to a better point in the search space. They start from an initial point in the search space, and then by making small movements and collecting gradient data at the same time, they try to move towards a higher point in the search space. So, gradient information is needed which somewhat complicates the problem. One has to calculate sensitivities of different parameters and how they affect the outputs. Moreover, since the search of a better result depends on gradient information, it is quite possible to end up at a local optimum point, and since there is no information about the global optimum one cannot determine how much the solution found is close to global optimum [5]. There are other different applications of the method that try to avoid local optimum and seek for the global optimum.

In a recent study, Sancibrian, García, Viadero and Fernández proposed a general procedure relying on gradient search method [1]. Method is based on local optimization. They have used exact gradients instead of numerically calculated gradients to get a more accurate “search direction”. The proposed method is capable of solving various synthesis problems and it allows calculation of gradients for any arrangement of linkages. Using exact gradients in search has its pros and cons. While they require differentiable expressions, at the end they give more precise results.

2.5 HEURISTIC METHODS

Tabu Search (TS), Simulated Annealing (SA) and Genetic Algorithm (GA) are among the most popular heuristic algorithms.

2.5.1 Tabu Search

Tabu search algorithm makes movements in the search space to find optimum points. First few moves are made in the local proximity of the current position and then it decides on a move that will drive the current location near to an optimum point. Tabu search keeps a list of recent moves and do not allow these to avoid repeating recent moves so it can search beyond local optimums [4].

2.5.2 Simulated Annealing

Simulated annealing algorithm seeks optimum solution analogous to annealing of metals. A temperature parameter controls direction of search. At the beginning of iterations, temperature is high and any search direction is acceptable. So at the beginning phase of the algorithm search is near to a random search. With increasing number of iterations temperature decreases and random movements in the search space are restricted. Only good moves are allowed at low temperatures [4]. Since the search gradually changes from a random search to a more structured search, simulated annealing method is able to avoid local optimum points.

2.5.3 Differential Evolution

Differential evolution is another evolutionary algorithm in which difference of two population vectors are subtracted and then added with a third population vector. Depending on the outcome of this operation, each individual is decided weather or not to survive to the next generation [6]. Main difference from genetic algorithm is that selection procedure is not a separate procedure.

Shiakolas, Koladiya and Kebrle has worked on a differential evolution algorithm which used “Geometric Centroid of Precision Positions Technique” to set the initial

boundaries for design parameters [6]. They tested the algorithm by applying it to a six-bar mechanism synthesis for dwell problem. The problem is handled in three test cases. For the first case, six-link mechanism is divided in to two loops, and at first the four-bar mechanism is synthesized then it is extended to a six-link mechanism. 18 precision points are used for this first case. For the second case, synthesis of the six-bar mechanism is made at one stage. Again 18 precision points are used. For the last case, number of precision points is reduced to 10 and single stage synthesis is made. For all three cases an accuracy level of 0.005 units is attained, however, first case needs the least number of iterations, while second case requires the most number of iterations. Their reasoning for this result is that, dividing the synthesis process into two stages reduces the total number of design variables for each stage so it is possible to reach the desired accuracy level with less iterations. Reducing the number of precision points to 10 also reduces the required number of iterations for the same accuracy, however it is not as efficient as the first case where two-stage synthesis is applied.

2.5.4 Genetic Algorithm

Genetic algorithm is an evolutionary search method that is influenced by natural genetics. It processes a population of solutions to a problem by its genetic operators and assigns a fitness value to each individual. By combining better solutions with each other it is able to drive the individuals of the population to a better point in the search space.

2.5.4.1 Terminology

Genetic algorithms have a terminology adopted from natural genetics. Following paragraphs will list these terms and their explanations [21].

String: A string is simply a possible solution to the given problem. It may be composed of binary numbers or it may be composed of real numbers according to the type of genetic algorithm. Parameters of the design are listed in a string. Strings can be thought of as individuals in a population.

Population: A population is a collection of strings that form a solution set to the problem. They are individuals spread over the search space. It may also be called "Generation".

Gene: Each number in a string is called a Gene. Genes can have real or binary numbers.

Fitness: Fitness is a measure of goodness of a string in the population. Strings with higher fitness values satisfy the required output(s) from the system more accurately.

2.5.4.2 Basic Genetic Algorithm

Genetic algorithm uses an evaluation function to evaluate a set of possible solutions to a problem and assigns a fitness value to each of the possible solutions. Genetic Algorithms rely on survival of the fittest principle [5]. Individuals are selected according to their fitness values to reproduce next generation of individuals. Since individuals with higher fitness are assigned a greater chance to be selected for reproduction, they tend to move towards a better position in the search space. Reproduction operators are inspired from natural genetics. Evaluation and selection procedures are also inspired from the survival of the fittest theorem of nature. GA's work on a set of solutions and they require an initial set of solutions to start the process. They also require an evaluation function to evaluate the solutions. They do not, however, require any other information about the problem and this makes GA's very simple to implement.

Basic Genetic Algorithm is composed of 4 program subroutines, which are evaluation, selection, crossover and mutation.

Evaluation

At this stage each String in the Population is evaluated by the fitness function and assigned a fitness value. Normally, this stage is the only stage that Genetic Algorithm uses information about the problem itself. It has to be able to analyze the system and decide whether it is a good design or not based on its fitness value.

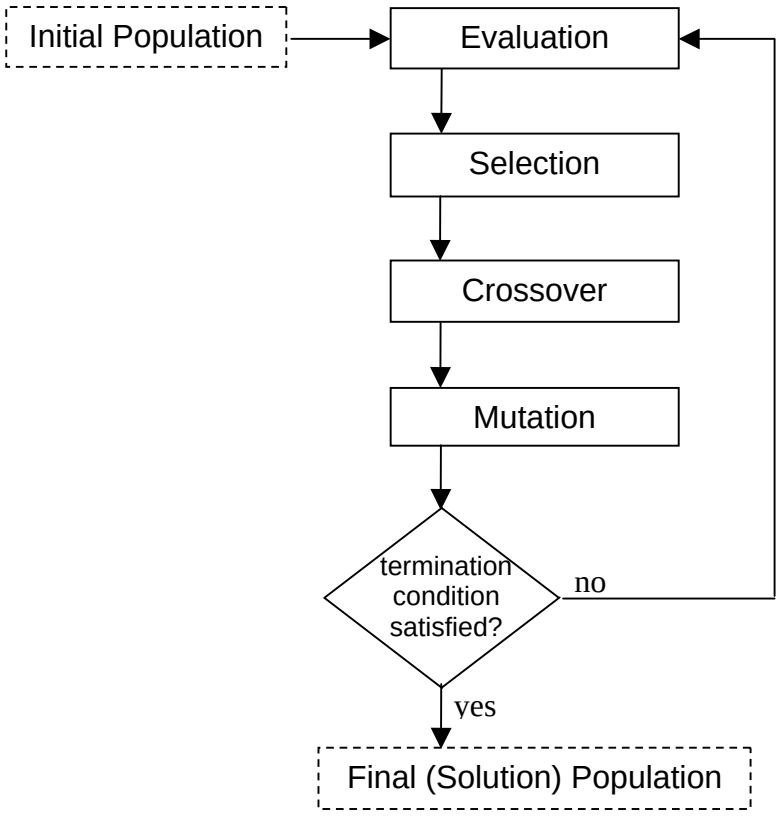

Figure 2.1 – Flow chart of a basic Genetic Algorithm

Selection

At this stage some of the strings in the population are selected according to their fitness values for crossover. As inspired from nature, the strings that have a higher fitness have more chance to be selected. There are different methods to apply while selecting individuals for reproduction. One of them is roulette wheel selection method [21]. A biased roulette wheel is used. Each string has a slot in the roulette wheel whose size is proportional to its fitness. Then the roulette wheel is rotated arbitrarily and one of the strings in the population is selected. This process is repeated to form an intermediate population with the selected strings. Clearly, the strings with higher fitness will have more chance to be selected; even they may be selected more than once. This intermediate population is used for crossover.

Crossover

The intermediate population formed in selection process is used to create a new population that will form the next generation. First the strings in the intermediate population are coupled with each other randomly and then each couple (parents) is crossed over to form two new strings (children). Crossover operation depends on the selected representation method of strings. If binary representation is used, it is done by swapping the genes between the couples from a randomly selected position along the string. If real number representation is used, a biased mean of the corresponding genes of the coupled strings are calculated according to a random number for each child.

Mutation

In order to keep the diversity in the population and to avoid convergence to a local optimum point some randomly selected strings undergo mutation. There are many

different possibilities to apply at this stage. Usually a few trials are needed to determine the best strategy and ratio for mutation.

2.5.4.3 Applications and Other Works on Genetic Algorithms

Genetic Algorithm has found many different application areas since its appearance. Its successful application to many different problems is an indication of its robust optimization power, which does not require problem specific insight. As it can be seen in [12-19], it has been successfully applied to problems of optimization, design, planning, scheduling, control, pattern matching, artificial learning and others.

Some work has been done to make a comparison among these algorithms. According to one of such studies by Habib Youssef, Sadiq M. Sait and Hakim Adiche [4], performance of an iterative algorithm depends highly on the type of problem so one has to find the suitable algorithm for the case in hand. They have found that Simulated Annealing algorithm performed better than Genetic Algorithm and Tabu Search for the problem of floor planning of very large-scale integrated circuits.

Arun Kunjur and Sundar Krishnamurty (KK) proposed a Genetic Algorithm with modified crossover and selection procedures to aid a better search of solution space [20]. They have pointed out a problem in assigning fitness values for mechanism optimization problems. They state that, since the objective of mechanism design problem is minimization, objective function will not reflect the fitness of individuals. A simple transform function could be used to overcome this, but it may cause a single solution to dominate the population. So they preferred to use a different fitness assignment procedure that sorts the individuals according to their objective function value and assign a fitness value based on their rank. They have used real number representation and crossover is made by real numbers. Crossover is made by calculating a weighted average value depending on the parent's fitness values. As they state this introduces a problem of premature convergence, however it may be

overcome by increasing the probability of mutation. They have observed that Genetic Algorithm converges to a near optimal solution at the first few steps. They have compared their results with a Genetic Algorithm that uses binary representation and with other optimization methods like exact and central difference gradient-based methods. It is observed that real number representation gives better results than binary representation while consuming less computation time. According to their results, after exact gradient-based method, Genetic Algorithm performs second best among the other algorithms.

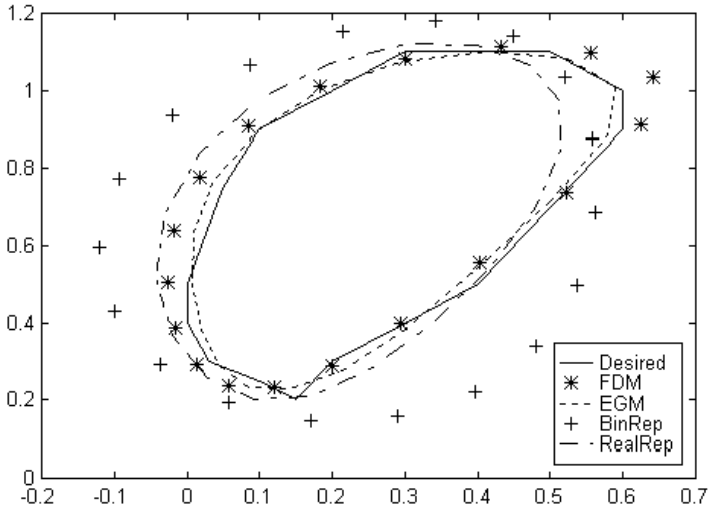

Figure 2.2 - Comparison of KK algorithm using binary representation (BinRep) and real number representation (RealRep) with Finite Difference (FDM) and Exact Gradient (EGM) methods. [20]

As Cabrera, Simon and Prado showed in their work [2], application of genetic algorithms give accurate solutions to mechanism synthesis problems. They have used

a differential evolution scheme for selection process where best individual is combined with two randomly selected individuals as shown below.

$$V = X_{best} + F(X_{r1} - X_{r2})$$

where X ’s are the vectors representing individuals and F is called disturbance used to control how much the best individual will be altered.

They have defined the problem as minimizing the squared differences of each precision point while also satisfying the Grashof condition and keeping the design variables in range, as shown.

$$\sum_{i=1}^{N} \left[\left(C_{Xd}^{i}(X) - C_{X}^{i}(X) \right)^2 + \left(C_{Yd}^{i}(X) - C_{Y}^{i}(X) \right)^2 \right]$$

where C_{Xd}^{i} is the desired x coordinate of the precision point and C_{X}^{i} is the achieved value.

Proposed algorithm is applied to two case problems, both of which are four-bar path synthesizing problems with prescribed timing. One of them has 5 precision points while the other has 18 precision points. For their case, initial mechanisms were randomly created, however the resulting solutions were similar to each other, thus they conclude that genetic algorithm converges to global optimum if enough iterations are made. They compared results of their algorithm with other methods that include a central difference method, an exact gradient method and another genetic algorithm by Kunjur and Krishnamurty. Accuracy of the solution was found to be better than other methods for small domains, for bigger domains it was close to exact gradient and central difference methods and better than the genetic algorithm proposed by Kunjur and Krishnamurty (KK).

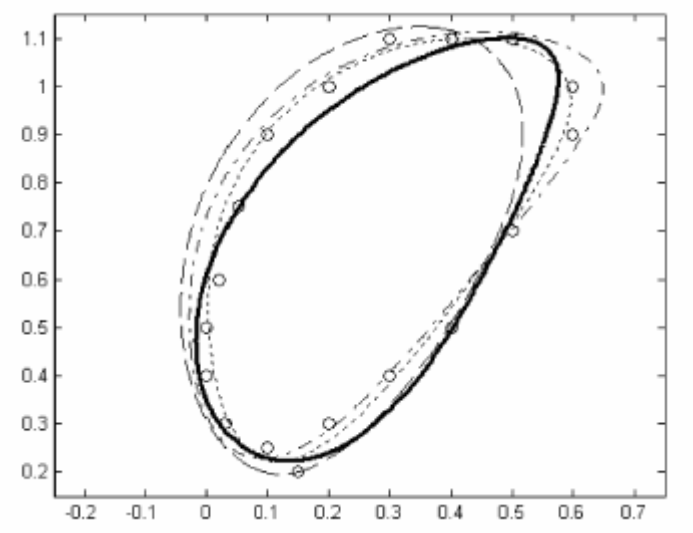

Figure 2.3 - Results for 18-point synthesis. ····· : exact gradient, −−- :KK, –·– : central difference, ▬ : Cabrera et al., o : target points. [2]

Another example of genetic algorithm in mechanism synthesis is by Laribi, Mlika, Romdhane and Zeghloul [3]. Their algorithm differs from a regular genetic algorithm that it is coupled with a fuzzy logic controller. Fuzzy logic controller gathers data about the design variables at the first run of the algorithm and then changes initial "bounding intervals". Following equation is used to obtain refined bounding values.

$$x^{*}_{\min} = x_{ave} - c_x / 2(x_{\max} - x_{\min})$$

$$x^{*}_{\max} = x_{ave} + c_x / 2(x_{\max} - x_{\min})$$

x_{ave}, x_{max} and x_{min} are obtained during the first run of the algorithm where x_{max} and x_{min} are initial bounding values of the first run and x_{ave} is the average value at the last generation of the first run. Coefficient c_x is calculated as a function of the error generated after the first run of the algorithm and variation of each parameter during the last 30 generations.

By this approach, this algorithm is able to find the global optimum avoiding local optimum points and it is claimed to have a better accuracy than classic genetic algorithms, which are not so accurate for large search spaces. Since this algorithm has to run twice to gather data at the first run, number of function evaluations is twice of the number of evaluations in a classic genetic algorithm. However, Laribi, Mlika, Romdhane and Zeghloul have shown that with their genetic algorithm, only half of the iterations are enough to reach an accuracy level of a classic genetic algorithm so their method is as fast as a classic genetic algorithm for the same accuracy level. Authors present two example runs one of which is the problem proposed by Kunjur and Krishnamurty and was also used by Cabrera et al. so it is possible to compare these 3 proposed methods.

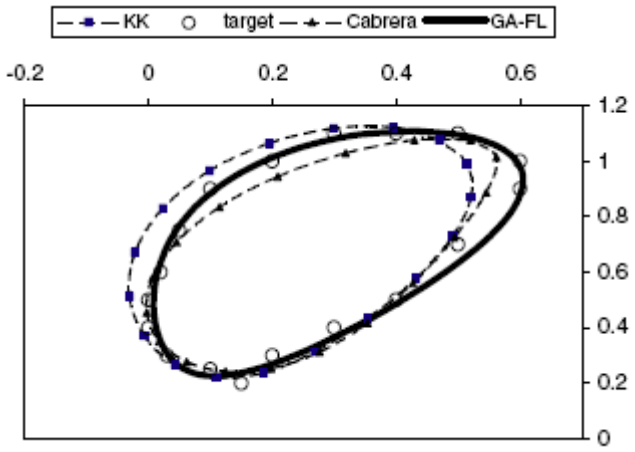

Figure 2.4 - Results for 18 point synthesis by Laribi et. al. [3]

Error reported for this GA-FL algorithm (Laribi et al.) is 0.2 whereas the errors for KK is 0.62 and for Cabrera's 0.29. To reach these error levels, Laribi's GA-FL

algorithm needed 1.32 seconds on an 1800 MHz processor, Cabrera's method needed 3.25 seconds on an 800 MHz processor and KK needed 37.03 seconds on a 33 MHz processor for this simple four-bar synthesis problem. Although the different CPU's used make it difficult to compare the algorithms in regard to computation cost, they all seem reasonably low.

Another work by Fernández-Bustos, J. Aguirrebeitia, R. Avilés, and C. Angulo, utilizes Genetic Algorithm to optimize an error function in a finite elements method that is used for synthesizing mechanisms [9].

In finite elements method, synthesis of the mechanism is done considering all the links as flexible links. So a starting mechanism is chosen and (for the path synthesizing problem) for each of the precision points, mechanism is forced to have its coupler point on the path. Resulting deformations at the linkages are added to calculate the deformation energy required to attain that position. It is suggested that when all required deformation energies for each precision points are added together, an estimation of the error of the mechanism can be gathered. The idea of this method is that, if total deformation energy can be minimized, relatively close to zero, by changing mechanism's parameters, then the resulting mechanism will be able to track the path precisely. Deformation energy is also called as "error function" as it also represents the error in the resulting path if we think of the links as rigid links.

The main concern of this work ([9]) is, as the authors suggested, developing a new error function that can be optimized with a Genetic Algorithm. Authors have used the following expression for strain energy:

$$\phi(\{x\}) = \frac{1}{2}\sum_{i=1}^{b} k_i (l_i(\{x\}) - L_i)^2$$

where index *i* denotes the link number, b is the total number of linkages, l_i is the deformed and L_i is the non-deformed length. k_i's represent the corresponding

stiffness values. If the strain energy $\phi(\{x\})$ can be minimized by changing vector x, a solution can be found.

However, since some configurations may have low stiffness with respect to others, they may have a lower deformation energy value although they are not better considering the resulting path. So, deformation energy as an indication of path error is just an estimation and not always so precise. This creates a problem such that, even the deformation energy of a mechanism turns out to be low, it may not precisely track the path closely. So authors have developed a new method for calculating error function. They based the error on the distance from the desired path rather than deformations. However, they still used the deformation energy concept to find the closest configuration of a mechanism to each of the precision points. Once the closest position to the path is found, the square of the distance in between is used as an indication of the error. Error function is formed by sum of all these distances. There are also some other suggested solutions regarding some special configurations of mechanisms. Authors also executed two sample runs and got two different mechanisms with the same inputs (initial conditions). For example, one of the mechanisms was synthesized in a time of 1 hour, with a population size of 20000 and total number iterations of 250 with a 2 GHz CPU.

A. Kanarachos, D. Koulocheris and H. Vrazopoulos proposed a combination of two different search algorithms [10]. One of them is Evolutionary Algorithm (EA) and the other is BFGS (Broyden, Fletcher, Goldfarb and Shanno) method. With this approach they replace the stochastic mutation operator of the EA with a deterministic one derived from BFGS method, so that they will be able to combine the better parts of two methods. For the EA part of the algorithm they used “panmictic” recombination, which allows more that two individuals to combine to form an offspring. Also, recombination is of type “intermediate” instead of “discrete” to obtain versatility among the population. Normally, in Evolutionary Search, mutation is applied to all of the population with a coefficient of probability of mutation. It is claimed that this approach causes loss of some useful information as individuals

undergo mutation. So, the deterministic mutation operator proposed in this work is applied to only a number of worst individuals of the population. While this process increases computation time because of the additional sorting procedure applied before mutation, it provides better convergence and may decrease the number of required number of iterations for a given accuracy level.

Authors have tested their method with the Fletcher and Powell test function and a four-bar path generation problem. In both of these porblems they got better results than the classical Genetic Algorithm and other conventional evolutionary methods.

H. Zhoua and Edmund H.M. Cheung used a modified Genetic Algorithm to optimize a hybrid five-bar mechanism in order to minimize the required maximum driving torque [11]. Hybrid five-bar mechanism is a chain of linkages and driven by two different kind of motors one of which is RTNA (real-time non-adjustable) type and drives the crank of the mechanism. The other motor is of type RTA (real-time adjustable) and it controls the position of the last link in the chain to control the path created by the coupler point. So, it is possible to get different paths with this hybrid mechanism by adjusting the position of the last linkage. Modified Genetic Algorithm is applied to a case problem where it is required to get a set of eliptic curves and the mechanism is effectively optimized to minimize the driving torque.

CHAPTER 3

APPLICATION OF GENETIC ALGORITHM

Genetic Algorithm used in this thesis is a general purpose Genetic Algorithm that is further altered to suit the needs of the mechanism synthesis problem. Flow chart of the algorithm is shown in Figure 3.1. First an initial population is created and then fitness test is applied. According to results of fitness test elimination, mutation and crossover are applied. And this loop continues until termination condition is satisfied. Termination condition is met if either one of the individual reaches a fitness value predefined by user input.

3.1 Problem Description

The problem can be described as finding a vector of parameter values that will increase the value of an objective function with regard to a starting solution. Parameter vector is as written below and it can be seen on Figure 4.2 as well.

$$x = \begin{pmatrix} \alpha_{m_1}, \alpha_{m_2}, \alpha_{k_1}, \alpha_{k_2}, \alpha_{k_3}, \alpha_{t_1}, \alpha_{d_1}, \alpha_{b_1}, \alpha_{b_2}, \\ m_1, m_2, k_1, k_2, k_3, k_4, l_1, l_2, t_1, t_2, d_1, d_2, b_1, b_2, b_3, h_{A_0} \end{pmatrix}$$

By changing the values of these parameters it is desired to increase the value of the objective function, which is:

$$f(x) = w_{BB}F_{BB}(x) + w_{AB}F_{AB}(x) + w_{LC}F_{LC}(x) + w_{BAH}D_{BAH}(x) + w_{BAL}D_{BAL}(x)$$

where, $F_{BB}(x)$, $F_{AB}(x)$, $F_{LC}(x)$, $D_{BAH}(x)$ and $D_{BAL}(x)$ are the bucket breakout force, arm breakout force, lift capacity, angle of bucket at lowest rack-back position and angle of bucket at highest rack-back position respectively and w_{BB}, w_{AB}, w_{LC}, w_{BAH} and w_{BAL} are the weighting factors for each. It is also desired to meet the constraints defined by:

$$DD(x) < DD_0$$
$$DH(x) > DH_0$$

where, $DD(x)$ is the digging depth and $DH(x)$ is the dump height. DD_0 and DH_0 are the limits on digging depth and dump height and they are taken as –100 mm for digging depth and 3300 mm for dump height.

3.2 Data Structure

Data structure of the program is made up of nested arrays. One of the arrays is the population array that holds all the information about the current generation. The elements of the population array consists of arrays of real and integer numbers. Examples to real and integer number parameters can be counted as crossover ratio, mutation ratio, population size, gnome length, fitness limit and generation index. These population parameters are defined during the initialization subroutine via user input. Arrays nested in the population array are individual array and parents array. These two arrays are identical to each other. Both have the same real number parameters and a gnome array that holds mechanism parameters. During crossover, individual data are written over to parents array and after crossover resulting individuals are written back on individual array. Data belonging to previous generations are erased as new generations are formed.

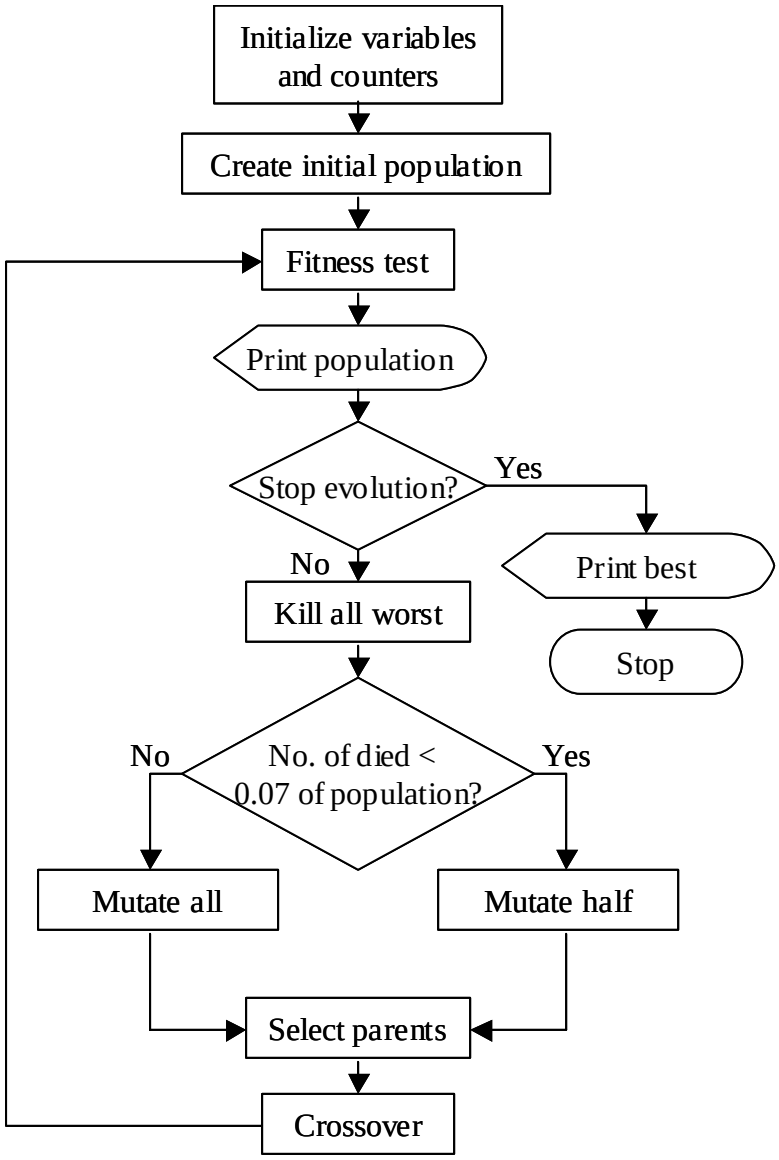

Figure 3.1 - Flow Chart of Genetic Algorithm

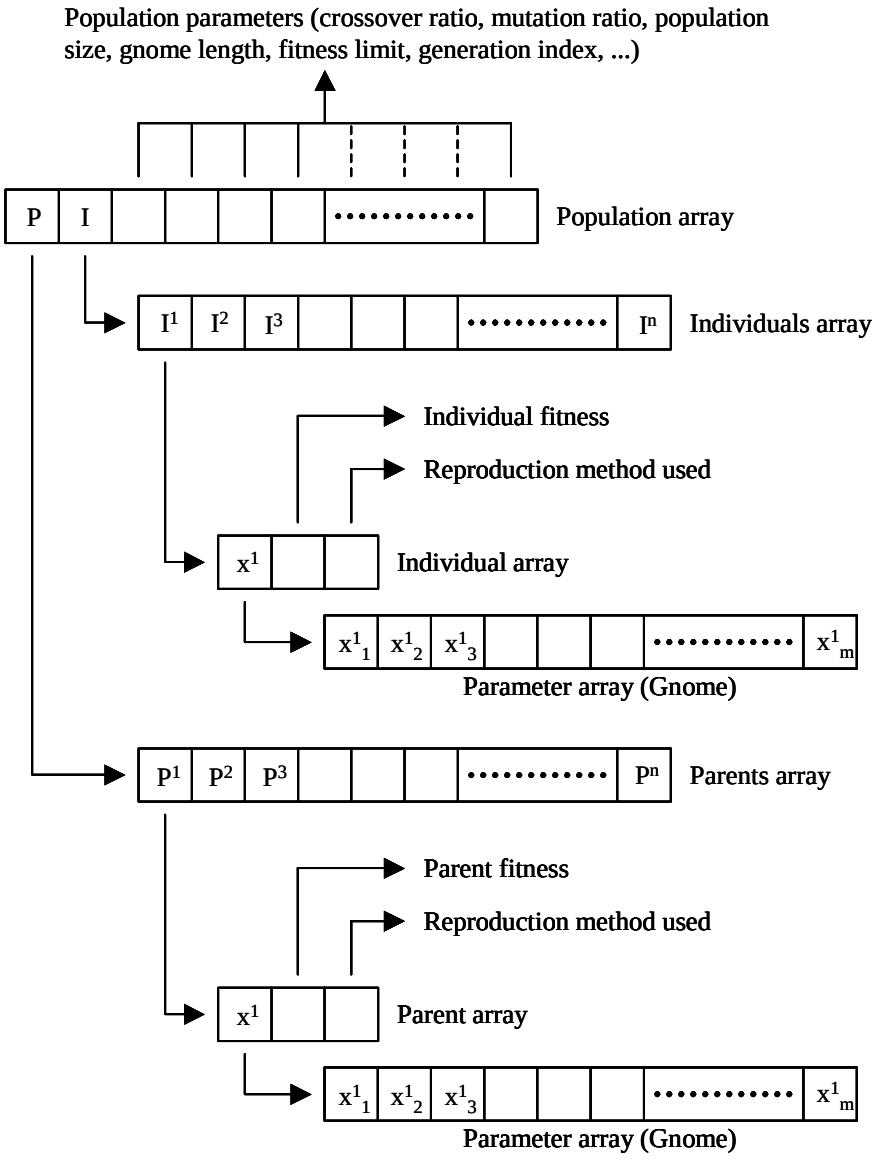

Figure 3.2 - Data Structure of Genetic Algorithm

3.3 Initialization

At the initialization stage of the algorithm initial parameter values for the first generation are selected randomly between selected lower and higher limits. There are 25 parameters and for each, initial values are determined by a previously synthesized mechanism that is close to the shape of the expected resultant mechanism. After finding the initial values, a range is determined for each parameter then minimum and maximum values for each parameter are calculated. Ranges are defined regarding the physical constraints on the mechanism size, and constraints on positions of the pivot points. So, some of the constraints are implied by these ranges.

3.4 Fitness Test

After a new generation of individuals is created, a fitness test is applied on each individual to assign a fitness value. Fitness value is calculated via various outputs of the mechanism, which are breakout and lifting forces calculated according to SAE J732, bucket angle at highest and lowest rack-back position, digging depth and dump height. Formula used to incorporate the different design evaluations into fitness value is as below:

$$F_i = \left(w_{BB} Fit_i^{BB} + w_{AB} Fit_i^{AB} + w_{LC} Fit_i^{LC} + w_{BAH} Fit_i^{BAH} + w_{BAL} Fit_i^{BAL}\right) \cdot Fit_i^{C} \quad (3.1)$$

F_i is the fitness value of the ith individual. w_{BB}, w_{AB}, w_{LC}, w_{BAH} and w_{BAL} are the corresponding weight factors of bucket breakout force, arm breakout force, lift capacity, bucket angle at highest and lowest rack-back positions respectively. They add up to 1 and determined according to the importance of each output. For this case they are set equal to each other. Fit_i^C is for digging depth and dump height constraints and take a value of either 1.0 or 0.25 depending on the satisfaction of the constraints. If the constraints are both satisfied, it becomes 1.0, else 0.25. Fit_i^{BB},

Fit_i^{AB}, Fit_i^{LC}, Fit_i^{BAH} and Fit_i^{BAL} are respective fitness values for bucket breakout force, arm breakout force, lift capacity, bucket angle at highest and lowest rack-back positions. They are calculated by taking the ratio of the current output to the desired output and the result is formulated to be always between 0 and 1 as shown below.

$$Fit_i^k = \begin{cases} \varphi_{i,desired}^k \Big/ \varphi_{i,current}^k & \text{if } \varphi_{i,desired}^k < \varphi_{i,current}^k \\ \varphi_{i,current}^k \Big/ \varphi_{i,desired}^k & \text{if } \varphi_{i,desired}^k > \varphi_{i,current}^k \end{cases} \tag{3.2}$$

$$k = BB, AB, LC, BAH, BAL$$

$\varphi_{i,current}^k$ values are the output of the generated mechanisms and $\varphi_{i,desired}^k$ values are the desired values. For bucket angles, desired value is input by user. For breakout forces and lift capacity, since the aim is to maximize those outputs, a high value is set as the desired value so that the relevant output may never reach that desired value, which are 75000 N for breakout forces and 35000 N for lift capacity.

Also, if a mechanism cannot satisfy closure requirements or its force analysis cannot be done because it is at a singular position, it is assigned a low fitness value so that it has much lower chance to be selected for crossover but still exist so that the chance it may result in a better individual when crossed is not eliminated. Its fitness value is multiplied by 0.25.

3.5 Selection

Selection of individuals for crossover involves both random selection and elitist selection together. After the elimination process, some of the individuals are deleted from the population array. Empty places in the population array are filled up with crossover, so number of individual pairs to be selected is equal to the number of

individuals that are deleted at the previous elimination subroutine. Half of the parents are selected by random number generation, the other half is sorted according to their fitness values and higher fitness ones are selected. Also the highest fitness individual is always selected for crossover by the algorithm.

3.6 Crossover

Real number representation is used in this algorithm. Crossover is also made by real numbers. Real number representation was found to yield more accurate results and it decreases the computation time with respect to binary representation according to the works of Kunjur and Krishnamurty [20]. Selected parents are crossed by calculating a biased mean of the corresponding parameters of each individual as shown in Figure 3.2. Random number R_i is different for each couple of parents and it can either be set to be same for each parameter of an individual or remain same for all parameters through user interface.

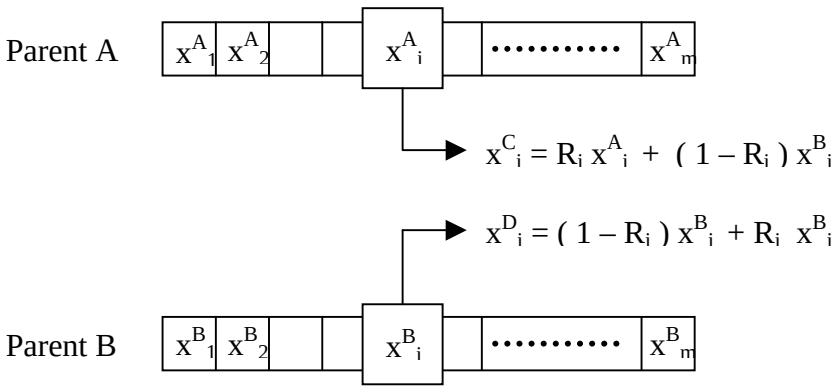

Figure 3.3 – Crossover operation

3.7 Mutation

Mutation is done by altering randomly selected parameters of some randomly selected individuals. Mutation probability is controlled by user input. If the population has more low fitness individuals then these low fitness individuals are mutated, otherwise one of the high fitness individual is mutated. Also according to the number of died individuals either half of the population is mutated or all of them are mutated. If the number of died individuals are less than 7% of the population, only half of the population undergoes mutation procedure. Less number of died individual means most of the population can achieve imposed constraints, so mutation is applied only to half of the population in order not to cause more individuals to be spoiled by mutation. Otherwise if number of died individuals is high, most of them already cannot satisfy constraints so there is not much loss by mutation of all of them.

3.8 Elimination

Individuals that have a fitness value below the mean of the current generation are deleted. After this process, population size is restored to its original value by reproduction.

3.9 Implementation and user interface

This genetic algorithm is implemented in Microsoft Excel® using built in Visual Basic for Applications editor (VBA). User input and output operations are managed via Excel interface. Figure 3.4 below shows a screenshot of parameter input and main controls page. Population output is positioned below these as seen in Figure 3.5.

	min	max				
alfa m1	10	15	deg	Max. Fitness To Reach		99,5
alfa m2	19	20	deg	Population Size		20
alfa k1	8	9	deg	Crossover Probability		0,7
alfa k2	172	175	deg	Mutation Probability		0,5
alfa k3	172	175	deg			
alfa t1	145	150	deg	desired bucket angle at ground level		45
alfa d1	2	6	deg	desired bucket angle at full height		58
alfa b1	62	68	deg			
alfa b2	7	12	deg			
m1	630	640	mm			
m2	240	250	mm	Build Population	Print Population	Calc. Fitness
k1	450	460	mm			
k2	1390	1400	mm	Evolve		
k3	760	770	mm			
k4	1210	1220	mm			
l1	400	410	mm		98,05037	Show
t1	260	270	mm	Show Results For Individual	2	
t2	350	360	mm			
d1	560	570	mm	bucket angle at ground level	45,10427	
d2	660	670	mm	bucket angle at full height	57,83576	
l2	890	900	mm			
b1	500	510	mm			
b2	930	940	mm			
b3	150	160	mm			
h A0	1695	1700	mm			

Figure 3.4 – Parameter input and controls page

PrntRng	1	2	3	4	5	6
alfa m1	10	15	12	14	10	15
alfa m2	20	20	20	20	19	20
alfa k1	9	9	8	9	8	8
alfa k2	173	175	173	173	175	173
alfa k3	173	174	173	174	174	172
alfa t1	146	150	147	148	150	147
alfa d1	6	2	4	6	5	4
alfa b1	67	67	67	68	67	66
alfa b2	11	11	8	7	8	9
m1	638	635	633	639	638	630
m2	249	242	245	244	250	242
k1	455	458	457	456	459	452
k2	1394	1392	1396	1395	1392	1392
k3	765	764	761	761	761	763
k4	1211	1217	1214	1218	1216	1218
l1	407	403	408	409	400	410
t1	263	267	261	262	265	260
t2	357	354	355	358	353	356
d1	568	565	564	569	562	569
d2	668	670	662	661	665	669
l2	896	898	899	893	898	891
b1	501	507	500	508	501	507
b2	937	936	930	930	935	938
b3	152	160	160	158	152	156
h A0	1700	1700	1695	1697	1697	1695
Fitness	88,13273	81,43429	90,08909	71,5687	79,03966	0

Figure 3.5 – Population output with fitness values

Within the same Excel file there is also mechanism position and force analysis that computes breakout forces and necessary angles. They are explained in the next chapter.

CHAPTER 4

ANALYSIS OF THE LOADER MECHANISM

Analysis of the mechanism is the part of the genetic algorithm where fitness test is conducted. Kinematic and force analyses are made to obtain outputs of the mechanism such as bucket angles at certain positions, arm and bucket breakout forces and lift capacities.

Analyses are formulated in an Excel sheet and as soon as Genetic Algorithm part of the Excel file inputs calculated mechanism parameters for fitness test, outputs are calculated in their respective cells. Then these outputs are read by genetic algorithm and evaluation of the mechanism is done.

4.1 Loader Mechanism

Loader is a two-degree of freedom mechanism with two slider inputs, which are formed by links 5, 6 and links 7, 8 in Figure 4.1. There are 11 linkages, 12 revolute joints and 2 prismatic joints as seen in Figure 4.1, which, according to General Degree-of-freedom equation [24], yields to:

$$F = \lambda \cdot (l - j - 1) + \sum_{i=1}^{j} f_i$$

$$l = 11$$

$$j = 12R + 2P = 14$$

$F = 3 \cdot (11 - 14 - 1) + 14 = 2$ degrees of freedom.

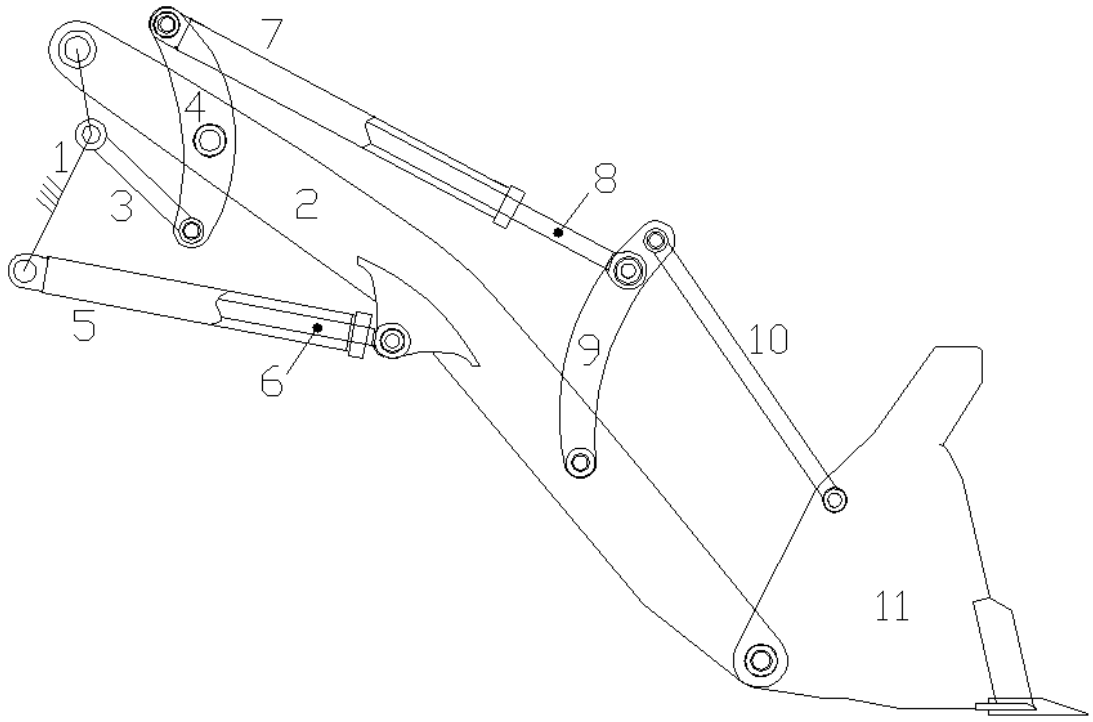

Figure 4.1 – Loader mechanism topology

4.1.1 Position Analysis

Mechanism parameters are marked on Figure 4.2. Inputs are the lengths s_1 and s_2. Required outputs are the bucket angle and angles of each other linkages for force analysis.

In order to ease position analysis, some pre-defined functions are used. First one is $MagCos(a,b,\theta)$ which finds the length of the edge opposite to angle θ. Other is

$AngCos(a,b,c)$ that calculates the positive angle opposite to edge c. $Angle(x,y)$ calculates angle of a line with respect to x-axis given its x and y coordinates. And lastly $FrBar(a,b,c,d,Cl,\theta)$, calculates the output angle of the four-bar whose link lengths are a, b, c, d, with closure condition Cl, and input angle θ. Each function is defined below.

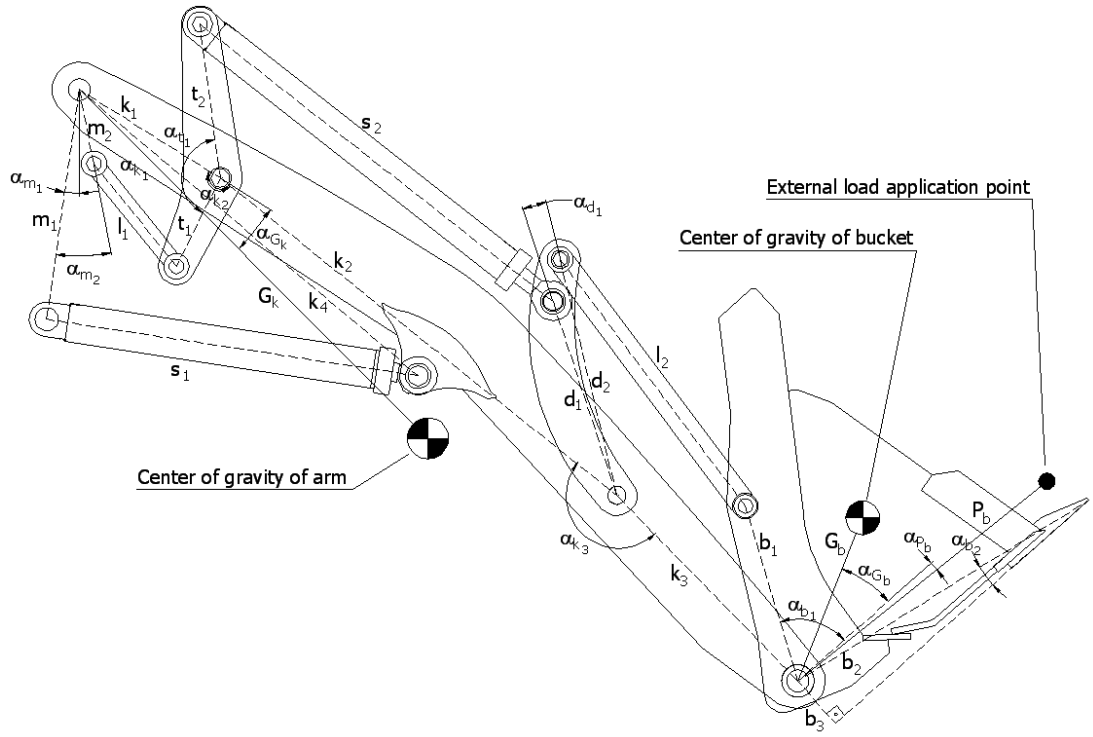

Figure 4.2 – Parameters of the loader mechanism

FrBar function calculates the output angle of the specified four-bar using the previously defined functions *MagCos* and *AngCos*. Formulation according to Figure 4.4 is as follows.

$$z_x = -d + a \cdot \cos\theta$$

$$z_y = a \cdot \sin\theta$$

$$z = \sqrt{z_x^2 + z_y^2}$$

$$\phi = Angle\left(z_x, z_y\right)$$

$$\psi = AngCos(c, z, b)$$

$$FrBar(a, b, c, d, Cl, \theta) = \phi - Cl \cdot \psi$$

where Cl defines the closure of the four-bar and is either 1 or –1 and *AngCos* is defined as

$$AngCos(a, b, c) = \left|\arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right)\right|$$

$$MagCos(a, b, \theta) = \sqrt{a^2 + b^2 - 2ab\cos\theta}$$

a, b and c are side lengths of an triangle and θ is the angle opposite to length c as shown in Figure 4.3.

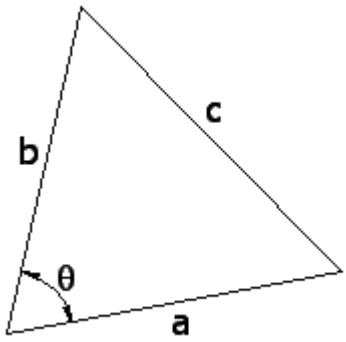

Figure 4.3 – Finding angle θ by cosine theorem

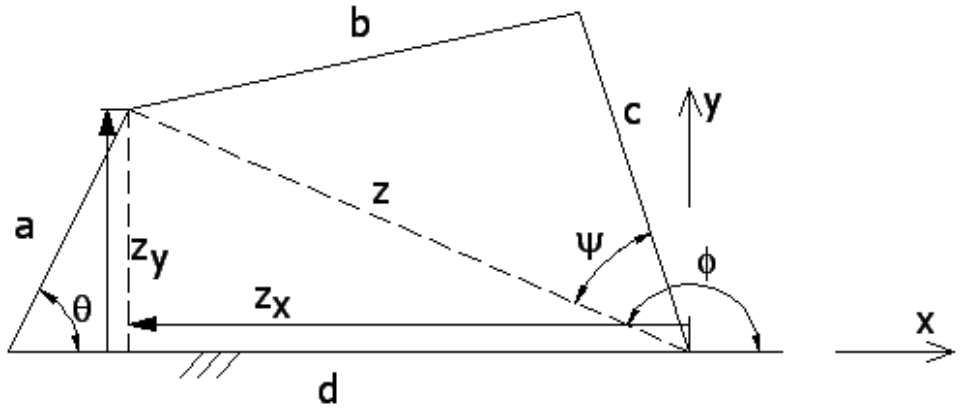

Figure 4.4 – Solution to four-bar

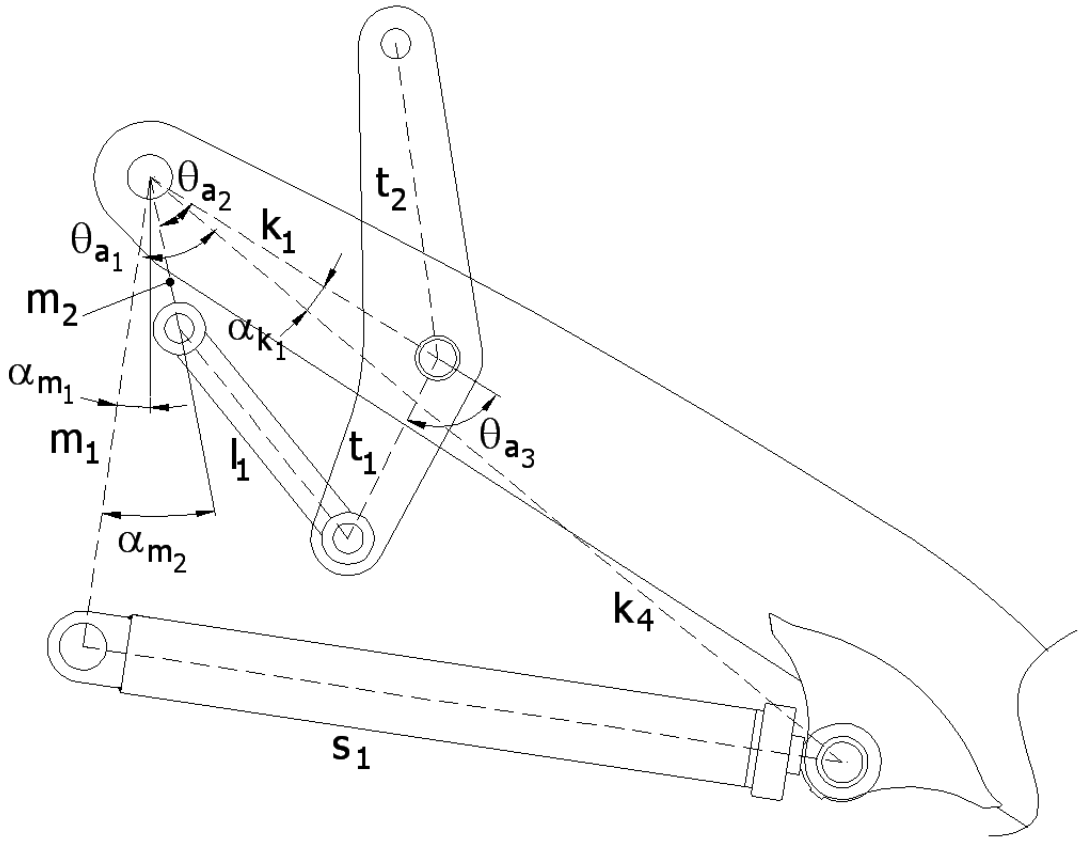

Figure 4.5 – Inverted slider-crank formed by m_1, k_4, s_1 and the four-bar formed by k_1, m_2, l_1, t_1

Given the input s_1, it is possible to find θ_{a1} by cosine theorem.

$$\theta_{a_1} = \arccos\left(\frac{m_1^2 + k_4^2 - s_1^2}{2m_1 k_4}\right)$$

and by using parameters α_{m_2} and α_{k_1}, input angle, θ_{a_2}, of the four-bar formed by k_1, m_2, l_1, t_1 can be found as

$$\theta_{a_2} = \theta_{a_1} - \alpha_{m_2} + \alpha_{k_1}$$

Output angle of four-bar k_1, m_2, l_1, t_1 is

$$\theta_{a_3} = FrBar(m_2, l_1, t_1, k_1, 1, \theta_{a_2})$$

Input angle of the next four-bar, k_2, t_2, s_2, d_1 can be calculated using the output angle of the first four-bar and the other parameters as follows.

$$\theta_{b_1} = \pi - \theta_{a_3} - \alpha_{t_1} - \alpha_{k_2}$$

Output angle of the four-bar k_2, t_2, s_2, d_1 is then calculated using the *FrBar* function defined before.

$$\theta_{b_2} = FrBar(t_2, s_2, d_1, k_2, 1, \theta_{b_1})$$

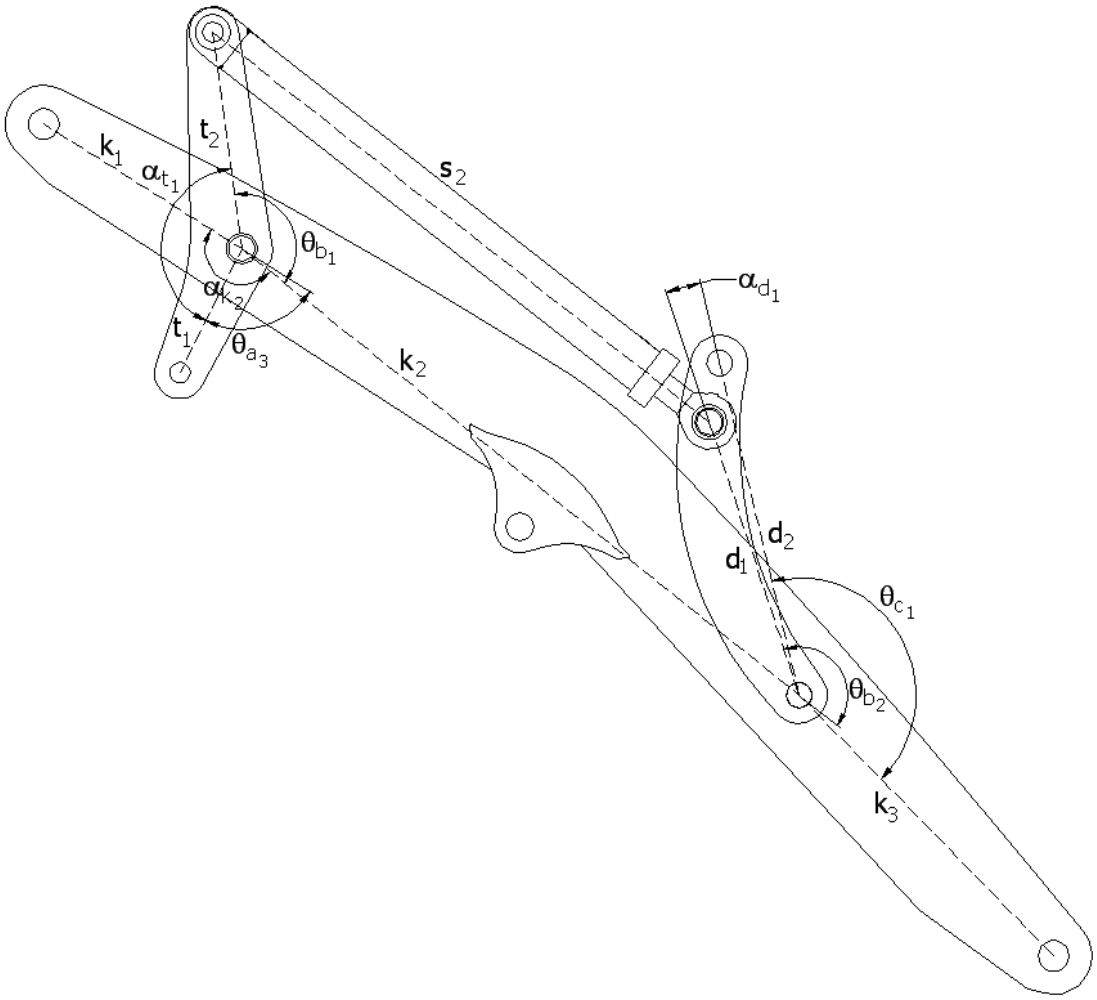

Figure 4.6 – Second four-bar on the arm of the loader formed by k_2, t_2, s_2, d_1

Input angle of the last four-bar, k_3, d_2, l_2, b_1, θ_{c_1} is calculated as follows.

$$\theta_{c_1} = \theta_{b_2} - \alpha_{d_1} + \pi - \alpha_{k_3}$$

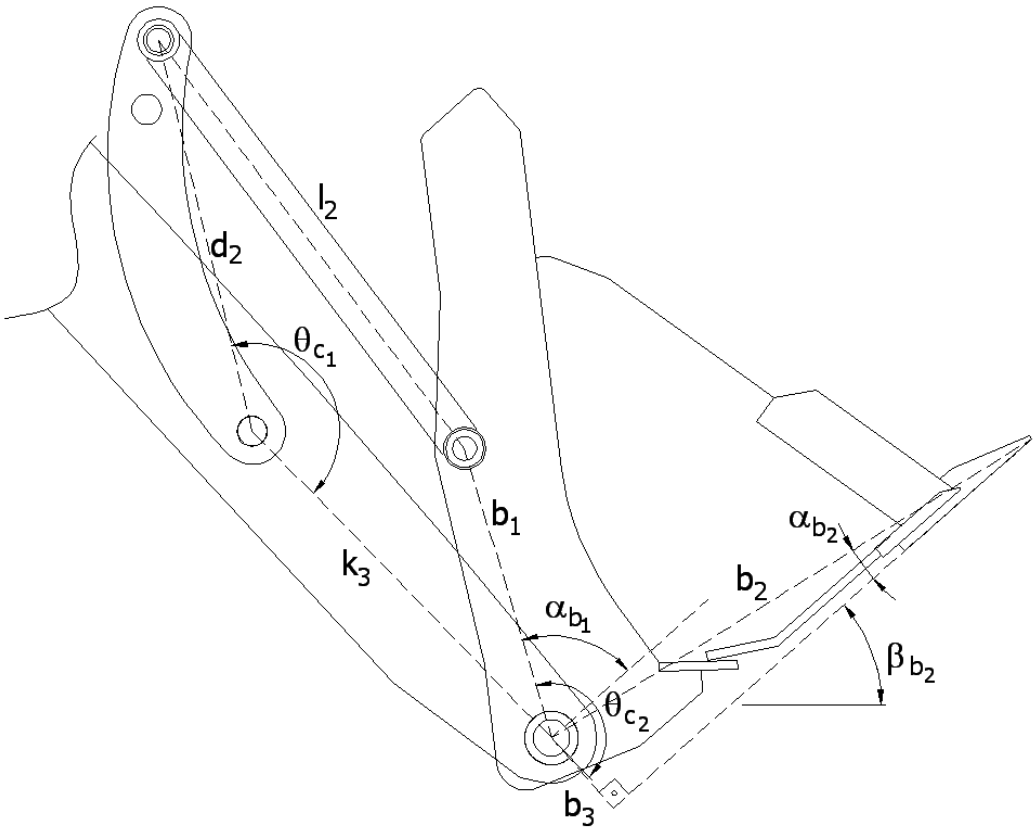

Figure 4.7 – Last four-bar on the loader arm formed by k_3, d_2, l_2, b_1

Finally output angle of the last four-bar, θ_{c_2} , is calculated as follows.

$$\theta_{c_2} = FrBar(d_2, l_2, b_1, k_3, 1, \theta_{c_1})$$

In order to calculate the bucket angle with respect to x-axis and to calculate each linkage's angle with respect to x-axis, to aid force analysis, additional calculations are to be made. They are given below.

$$\beta_{s_1} = \frac{\pi}{2} - \alpha_{m_1} - AngCos(m_1, s_1, k_4)$$

$$\beta_{k_1} = \alpha_{k_1} + \theta_{a_1} - \theta_{m_1} - \frac{\pi}{2}$$

$$\beta_{k_2} = \beta_{k_1} - \pi + \alpha_{k_2}$$

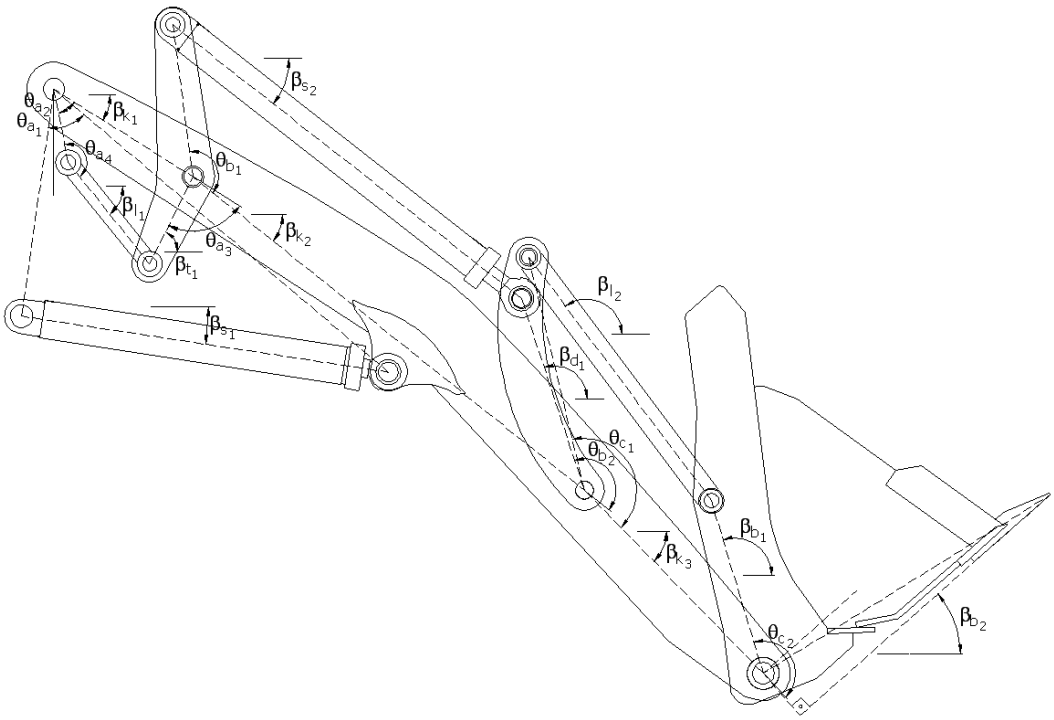

Figure 4.8 – Orientation angles of each linkage with respect to x-axis for use in force analysis

$$\beta_{k_3} = \beta_{k_2} - \pi + \alpha_{k_3}$$

$$\beta_{l_1} = \beta_{k_1} - \theta_{a_2} + \pi - \theta_{a_4}$$

$$\beta_{t_1} = \beta_{k_1} - \theta_{a_3} + \pi$$

$$\beta_{s_2} = \beta_{t_1} - \alpha_{t_1} + AngCos(s_2, t_2, MagCos(k_2, d_1, \pi - \theta_{b_2}))$$

$$\beta_{d_1} = \beta_{k_3} + \theta_{c_1} + \alpha_{d_1}$$

$$\beta_{l_2} = \beta_{k_3} + \theta_{c_1} + AngCos(d_2, l_2, MagCos(k_3, b_1, \pi - \theta_{c_2}))$$

$$\beta_{b_1} = \beta_{k_3} + \theta_{c_2}$$

$$\beta_{b_2} = \beta_{b_1} - \alpha_{b_1}$$

where β_{b_2} is one of the output parameters that is tried be made to match the user input values for defined positions of the mechanism. Remaining β angles are required for force analysis.

4.1.2 Force Analysis

In order to calculate bucket breakout, arm breakout forces and lift capacity, force analysis is to be conducted on the mechanism. Method of free body diagrams is used for force analysis. To keep number of equations low thus keep the matrix size smaller, weights of two-link members are distributed among the joints. Resulting matrix equation is solved in Excel and linkage forces are obtained. Breakout forces are then calculated according to SAE J732 standard. Free body diagrams for each member and equations are given below.

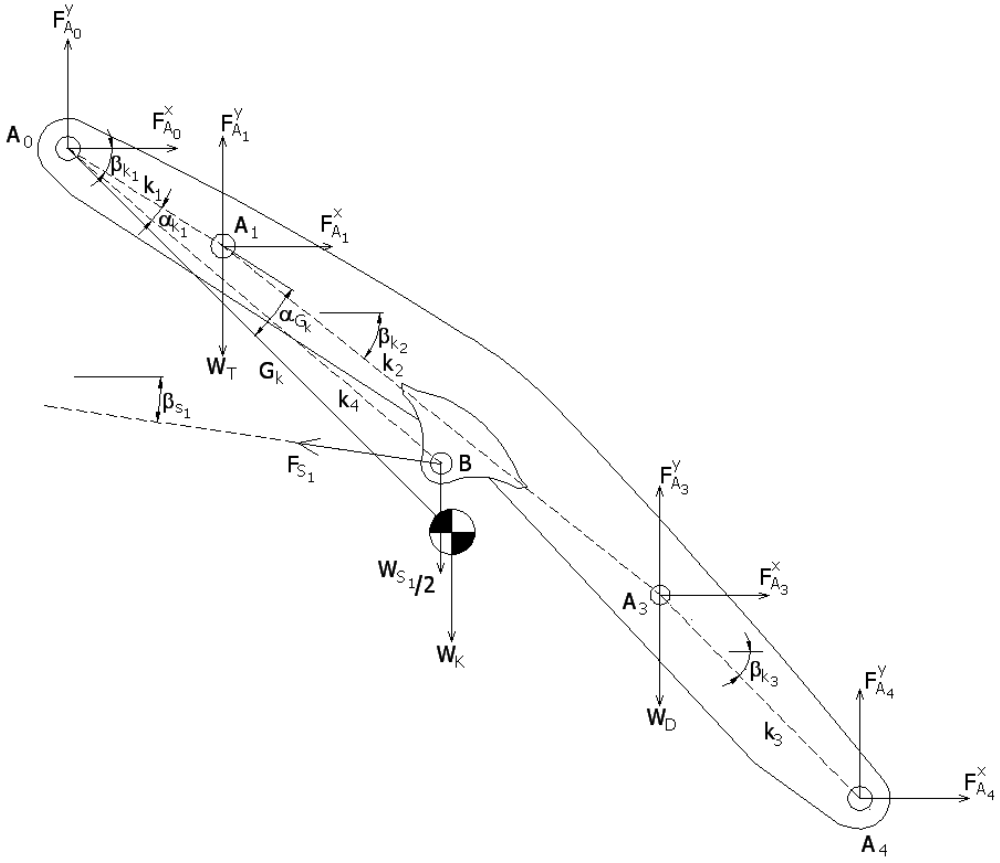

Figure 4.9 – Free body diagram of the loader arm

$$\sum F^x = 0 \Rightarrow F^x_{A_0} + F^x_{A_1} + F_{S_1}\cos(\beta_{s_1} + \pi) + F^x_{A_3} + F^x_{A_4} = 0$$

$$\sum F^y = 0 \Rightarrow F^y_{A_0} + F^y_{A_1} + F_{S_1}\sin(\beta_{s_1} + \pi) + F^y_{A_3} + F^y_{A_4} = W_T + \frac{W_{S_1}}{2} + W_K + W_D$$

$$\sum M_{A_0} = 0 \Rightarrow F^x_{A_1} k_1 \sin(-\beta_{k_1}) + F^y_{A_1} k_1 \sin(\frac{\pi}{2} - \beta_{k_1})$$
$$- F^x_{A_3}\left(k_1\sin(\beta_{k_1}) + k_2\sin(\beta_{k_2})\right) + F^y_{A_3}\left(k_1\cos(\beta_{k_1}) + k_2\cos(\beta_{k_2})\right)$$
$$- F^x_{A_4}\left(k_1\sin(\beta_{k_1}) + k_2\sin(\beta_{k_2}) + k_3\sin(\beta_{k_3})\right)$$
$$+ F^y_{A_4}\left(k_1\cos(\beta_{k_1}) + k_2\cos(\beta_{k_2}) + k_3\cos(\beta_{k_3})\right) - F_{S_1}k_4\sin(\beta_{s_1} - (\beta_{k_1} - \alpha_{k_1}))$$
$$= -k_1 W_T \sin(\frac{3\pi}{2} - \beta_{k_1}) - k_4\frac{W_{S_1}}{2}\sin(\frac{3\pi}{2} - (\beta_{k_1} - \alpha_{k_1}))$$
$$- G_k W_K \sin(\frac{3\pi}{2} - (\beta_{k_1} - \alpha_{G_k})) + W_D\left(k_1\cos(\beta_{k_1}) + k_2\cos(\beta_{k_2})\right)$$

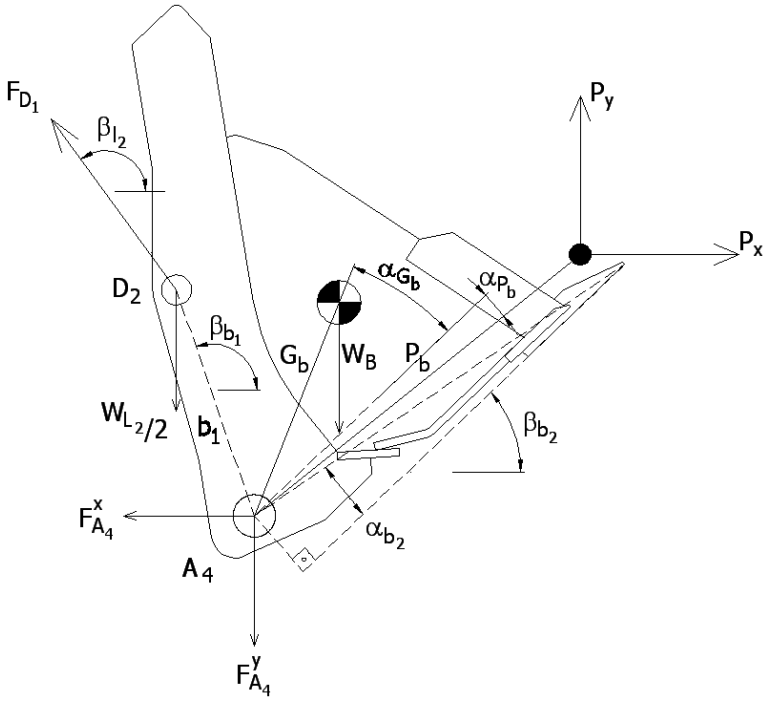

Figure 4.10 – Free body diagram of the bucket

$$\sum F^{x} = 0 \Rightarrow -F_{A_4}^{x} + F_{D_1}\cos(\beta_{l_2}) = -P_x$$

$$\sum F^{y} = 0 \Rightarrow -F_{A_4}^{y} + F_{D_1}\sin(\beta_{l_2}) = \frac{W_{L_2}}{2} + W_B - P_y$$

$$\sum M_{A_4} = 0 \Rightarrow F_{D_1} b_1 \sin(\beta_{l_2} - \beta_{b_1}) = \frac{W_{L_2}}{2} b_1 \cos(\beta_{b_1}) + W_B G_b \cos(\beta_{b_2} + \alpha_{G_b})$$
$$+ P_b P_x \sin(\beta_{b_2} - \alpha_{P_b}) - P_b P_y \cos(\beta_{b_2} - \alpha_{P_b})$$

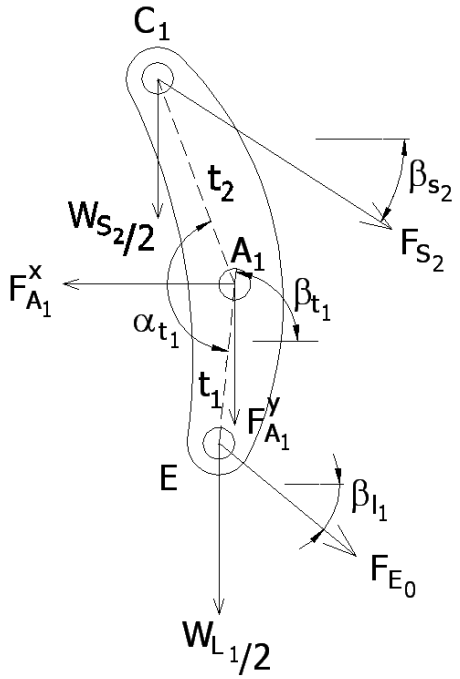

Figure 4.11 – Free body diagram of the linkage “t”

$$\sum F^x = 0 \Rightarrow -F_{A_1}^x + F_{E_0}\cos(\beta_{l_1}) + F_{S_2}\cos(\beta_{s_2}) = 0$$

$$\sum F^y = 0 \Rightarrow -F_{A_1}^y + F_{E_0}\sin(\beta_{l_1}) + F_{S_2}\sin(\beta_{s_2}) = \frac{W_{S_2}}{2} + \frac{W_{L_1}}{2}$$

$$\sum M_{A_1} = 0 \Rightarrow F_{E_0} t_1 \sin(\beta_{l_1} - (\beta_{t_1} + \pi)) + F_{S_2} t_2 \sin(\beta_{s_2} - (\beta_{t_1} + \pi - \alpha_{t_1}))$$

$$= W_{S_2} t_2 \cos(\beta_{t_1} + \pi - \alpha_{t_1}) - \frac{W_{l_1}}{2} t_1 \cos(\beta_{t_1})$$

$$\sum F^x = 0 \Rightarrow -F_{A_3}^x + F_{D_1}\cos(\beta_{l_2} - \pi) + F_{S_2}\cos(\beta_{s_2} + \pi) = 0$$

$$\sum F^y = 0 \Rightarrow -F_{A_3}^y + F_{D_1}\sin(\beta_{l_2} - \pi) + F_{S_2}\sin(\beta_{s_2} + \pi) = 0$$

$$\sum M_{A_3} = 0 \Rightarrow F_{D_1} d_2 \sin(\beta_{l_2} - \pi - (\beta_{d_1} - \alpha_{d_1})) + F_{S_2} d_1 \sin(\beta_{s_2} + \pi - \beta_{d_1})$$

$$= \frac{W_{L_2}}{2} d_2 \cos(\beta_{d_1} - \alpha_{d_1}) + \frac{W_{S_2}}{2} d_1 \cos(\beta_{d_1})$$

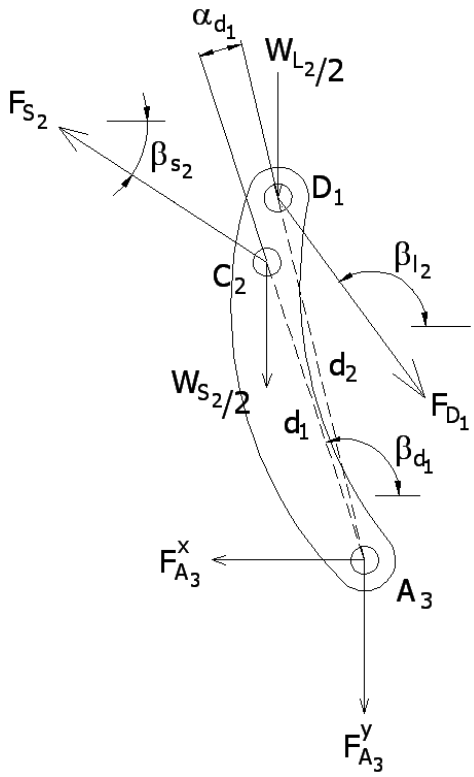

Figure 4.12 – Free body diagram of the linkage “d”

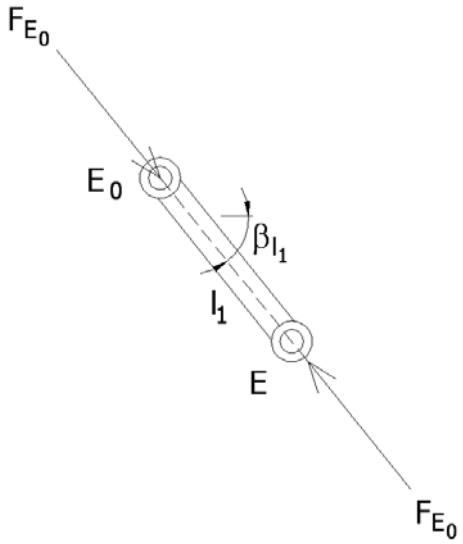

Figure 4.13 – Free body diagram of the two-force member “l_1”

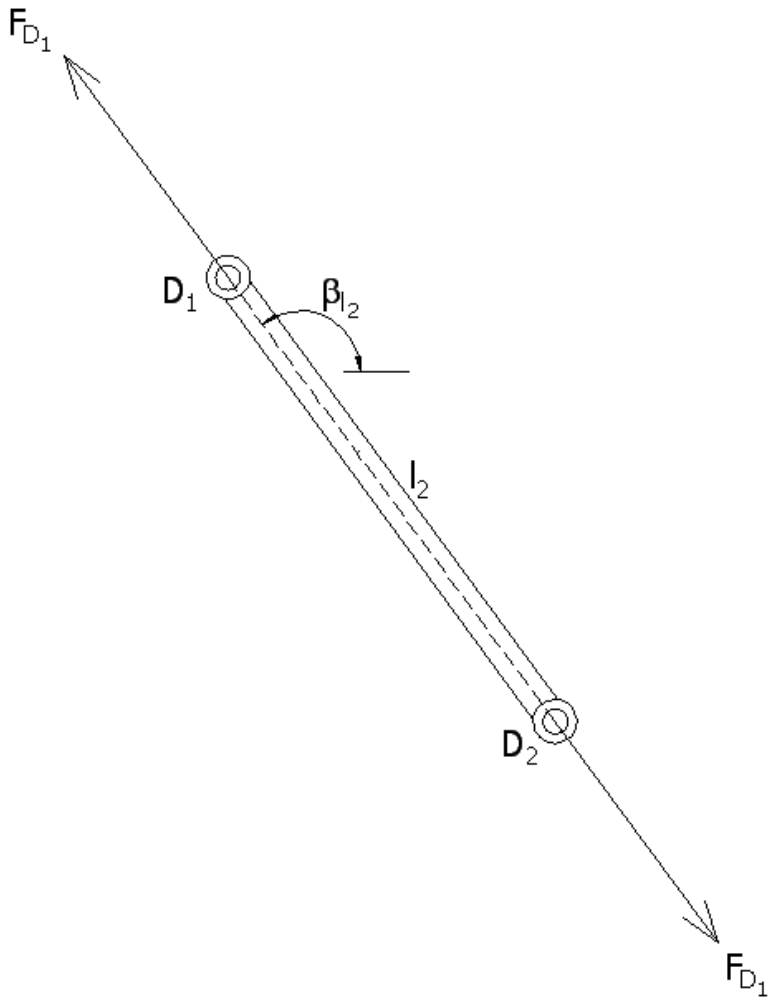

Figure 4.14 – Free body diagram of the two-force member ”l_2”

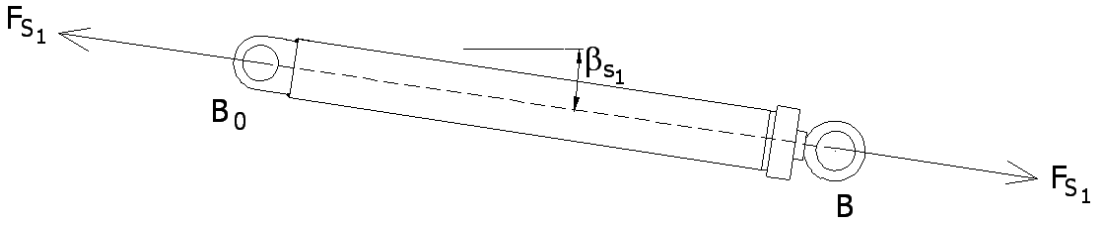

Figure 4.15 – Free body diagram of the hydraulic cylinder s_1

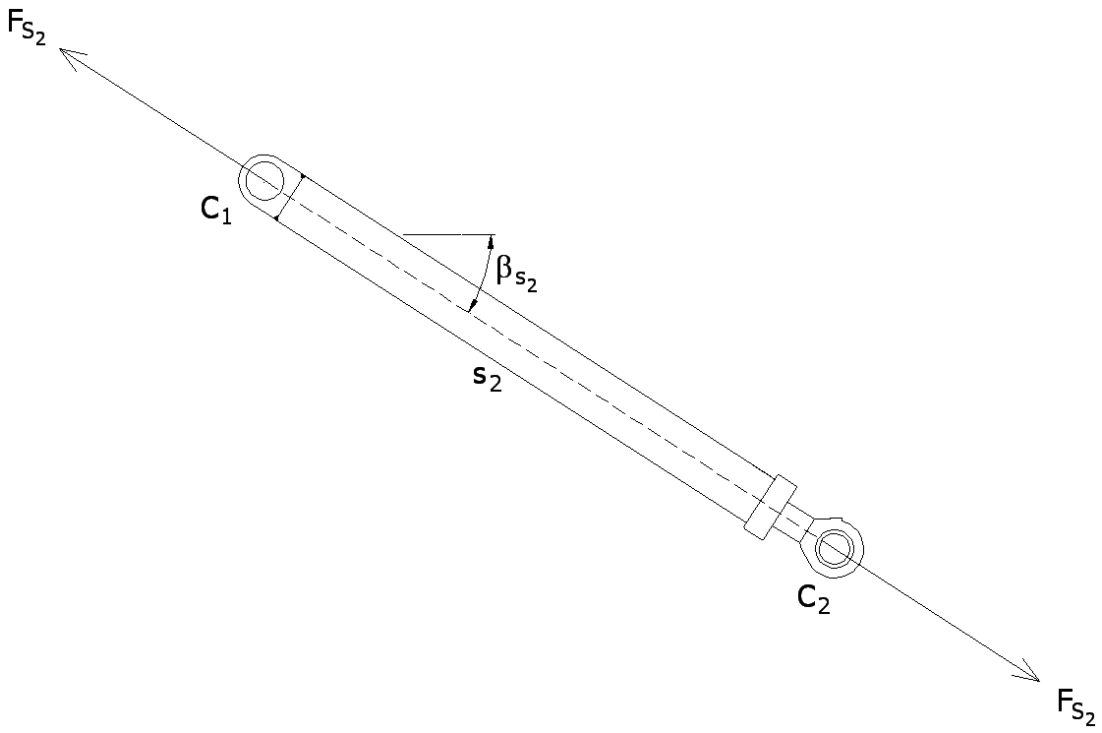

Figure 4.16 – Free body diagram of the hydraulic cylinder s_2

F A0 x	F A0 y	F A1 x	F A1 y	F A3 x	F A3 y	F A4 x	F A4 y	F E0	F D1	F S1	F S2
1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00	-0,76	0,00
0,00	1,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	0,00	-0,65	0,00
0,00	0,00	-205,76	402,94	-711,23	1703,81	-900,13	2446,58	0,00	0,00	-456,29	0,00
0,00	0,00	-1,00	0,00	0,00	0,00	0,00	0,00	0,89	0,00	0,00	0,87
0,00	0,00	0,00	-1,00	0,00	0,00	0,00	0,00	0,46	0,00	0,00	0,49
0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	224,55	0,00	0,00	-356,01
0,00	0,00	0,00	0,00	-1,00	0,00	0,00	0,00	0,00	0,99	0,00	-0,87
0,00	0,00	0,00	0,00	0,00	-1,00	0,00	0,00	0,00	0,11	0,00	-0,49
0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-558,31	0,00	546,00
0,00	0,00	0,00	0,00	0,00	0,00	-1,00	0,00	0,00	-0,99	0,00	0,00
0,00	0,00	0,00	0,00	0,00	0,00	0,00	-1,00	0,00	-0,11	0,00	0,00
0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	450,67	0,00	0,00

Figure 4.17 – An example shot of the coefficient matrix $[A]$, in Excel sheet

Above equations can be written in matrix form.

$$[A]\cdot[x]=[C]$$

where $[A]$ is the coefficient matrix. $[C]$ is the load matrix, which is composed of the terms at the right hand side of the force equilibrium equations, and $[x]$ is the array of unknown forces. Solution to this matrix equation can be found by multiplying each side with $[A]^{-1}$.

$$[A]^{-1}\cdot[A]\cdot[x]=[A]^{-1}\cdot[C]$$
$$[\mathrm{x}]=[\mathrm{A}]^{-1}[\mathrm{C}]$$

Therefore unknown forces array can be found, which gives the following joint forces.

$$[x]=\begin{bmatrix}F_{A_0}^{x} & F_{A_0}^{y} & F_{A_1}^{x} & F_{A_1}^{y} & F_{A_3}^{x} & F_{A_3}^{y} & F_{A_4}^{x} & F_{A_4}^{y} & F_{E_0} & F_{D_1} & F_{S_1} & F_{S_2}\end{bmatrix}^{T}$$

Bucket Breakout Force	
78886	N
8041	kg-f

Arm Breakout Force	
73127	N
7454	kg-f

Max Load Capacity	
45881	N
4677	kg-f

Figure 4.18 – An example shot of the Excel sheet where bucket, arm breakout forces and maximum lifting capacity values are calculated.

Using forces F_{S_1} and F_{S_2}, which are forces acting on hydraulic cylinders, pressures in the cylinders can be calculated. Then, arm breakout and bucket breakout forces are calculated according to SAE J732 [23].

In some cases calculation of breakout forces may not be possible if the mechanism is in a singular position. If the mechanism is in a singular position, determinant of matrix A will be equal to zero, thus the matrix equation cannot be solved. Or, in some cases even position analysis may not be possible if linkage lengths do not allow assembly of the mechanism. In those cases, Excel sheet outputs an error message to the relevant output cell, and their fitness values are lowered by a coefficient as explained in Chapter III.

CHAPTER 5

CASE STUDY

In this chapter, results of four sample runs will be presented. First two of them start with the same initial parameter values whereas the third one starts with a different set of initial parameter values. Fourth one is a special case where the value of parameter d_2 is set equal to the value of d_1 and value of α_{d_1} is set to zero. Therefore, the two joints on linkage "d" merge to form one single joint and a different type of mechanism is formed as shown in Figure 5.1.

Results will be compared with the initial set regarding arm breakout force, bucket breakout force and lifting capacity then they will be checked for linkage interference. Results of the initial mechanism are given in Table 5.1.

Table 5.1 – Results for the initial mechanism

Arm Breakout Force	58664 N
Bucket Breakout Force	68964 N
Lifting Capacity	28792 N
Bucket angle at ground level	41.3 degree
Bucket angle at full height	58.7 degree

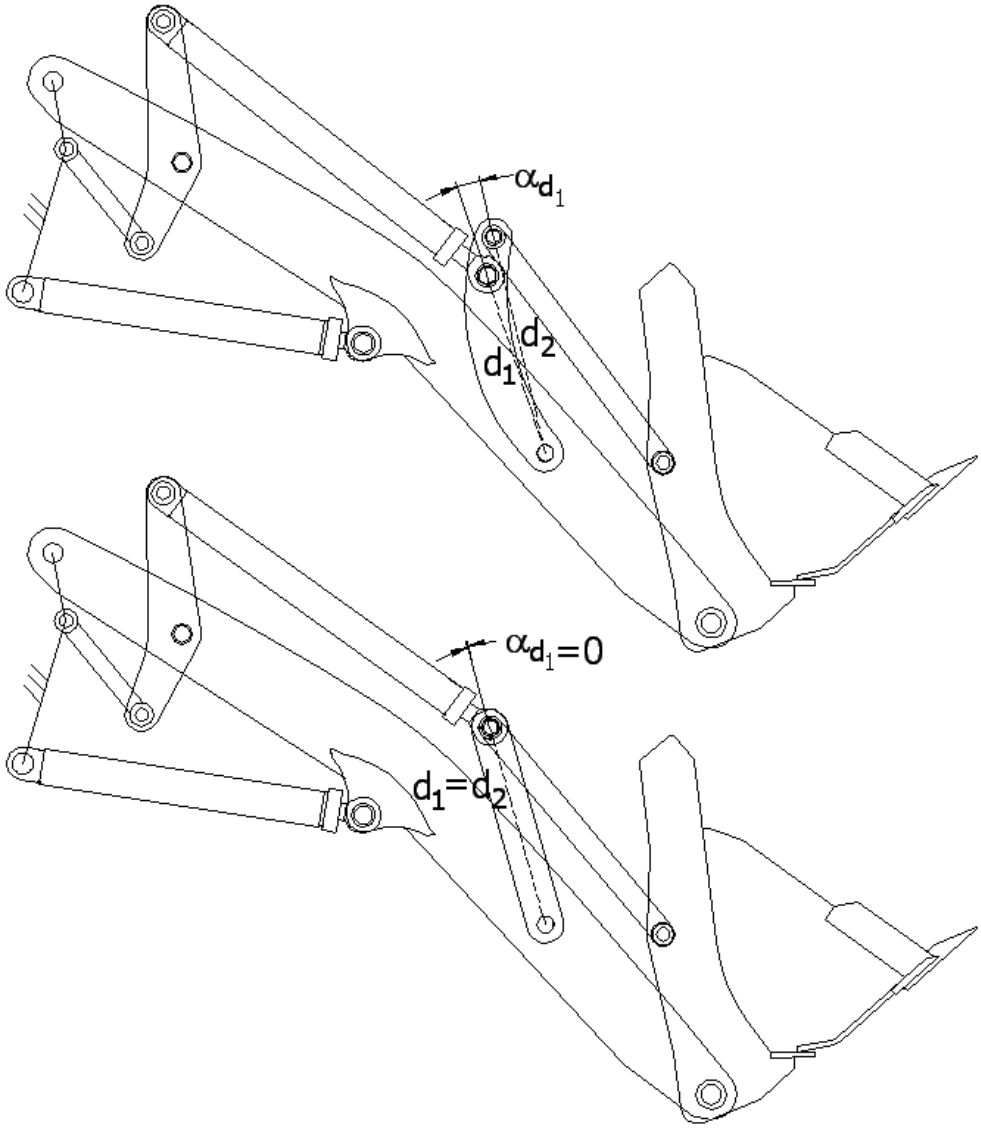

Figure 5.1 – Reduction of number of joints for case #4

For all runs of the program a maximum fitness value of 98 is aimed. Population size is set to 30. Crossover and mutation probabilities are 1 and 0.35 respectively. Desired bucket angle at ground level and full height are 44 and 59 degrees respectively for the first two runs, 41 and 59 for the last two runs.

Results are presented in Table 5.2.

Table 5.2 – Results for the sample runs

	#1	#2	#3	#4
Arm Breakout Force	66324 N	63956 N	61388 N	66678 N
Bucket Breakout Force	73244 N	70654 N	68459 N	73628 N
Lifting Capacity	29685 N	29137 N	29668 N	29573 N
Bucket angle at ground level	44.0 degree	44.3 degree	42.1 degree	42.0 degree
Bucket angle at full height	59.4 degree	59.0 degree	58.9 degree	60.1 degree

Percent improvements with respect to the initial mechanism are given in Table 5.3.

Table 5.3 – Percent improvements in breakout forces and lift capacity

	#1	#2	#3	#4
Arm Breakout Force	13.1 %	9.0 %	4.6 %	13.7 %
Bucket Breakout Force	6.2 %	2.5 %	-1.0 %	6.8 %
Lifting Capacity	3.1 %	1.2 %	3.0 %	2.7 %

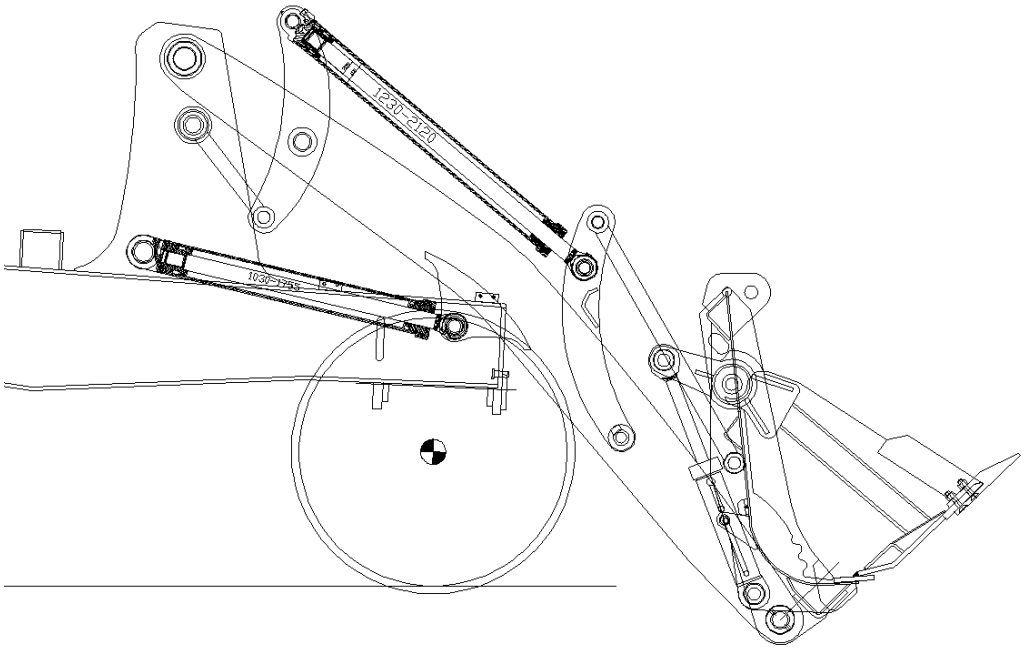

Figure 5.2 – Mechanism #1 at its lowered position.

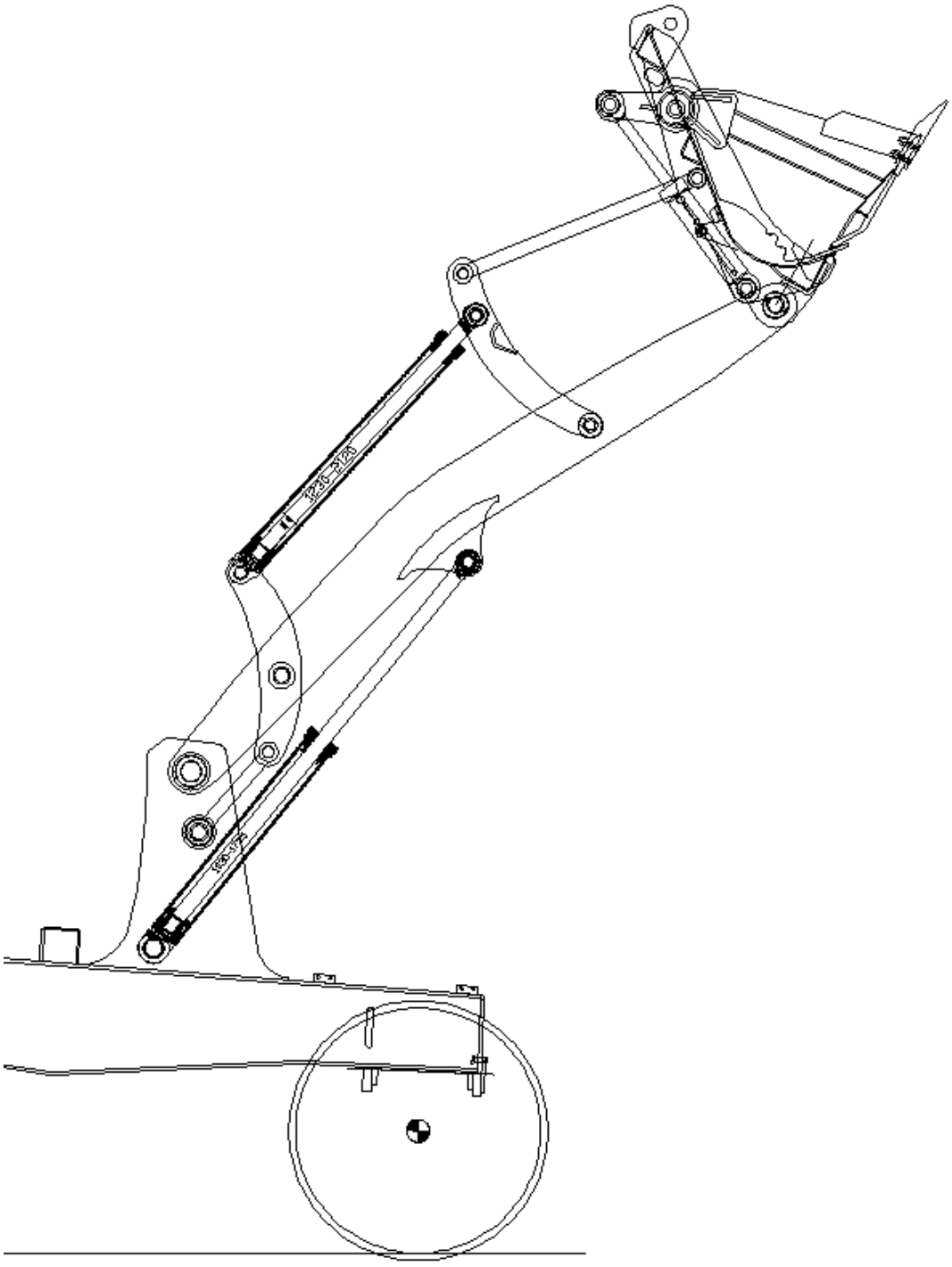

Figure 5.3 – Mechanism #1 at dumping position.

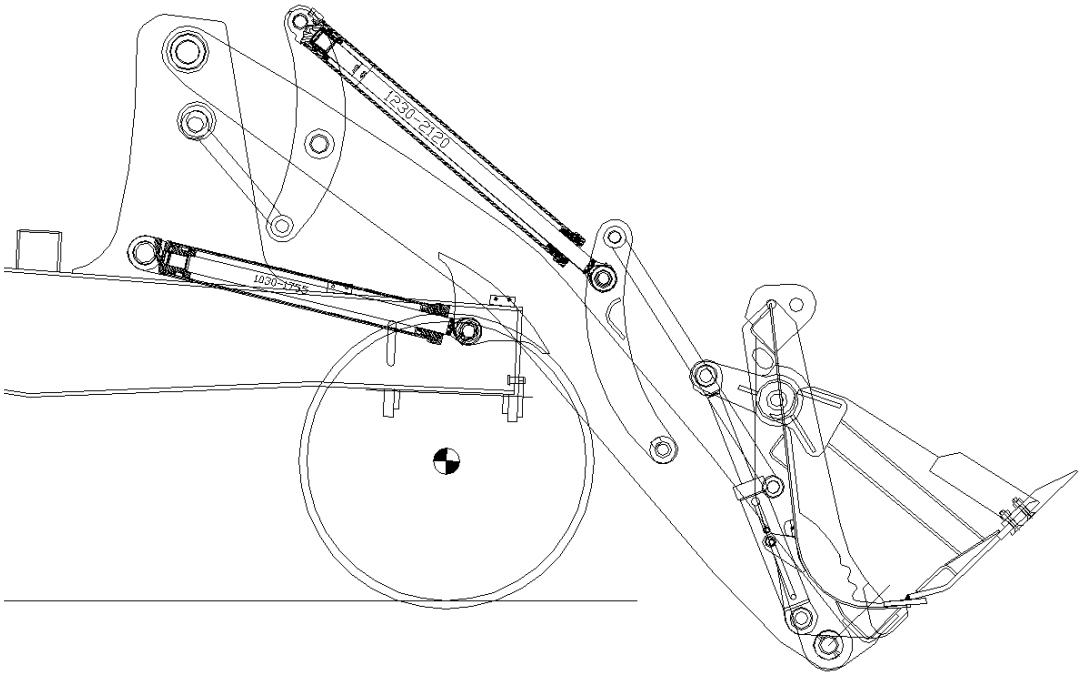

Figure 5.4 – Mechanism #2 at lowered position.

As seen in figures 5.2 to 5.9, generated mechanisms do not have any interfering linkages. They all satisfy dumping height requirements and digging depth requirements. Comparisons of the found solutions with the initial solution are presented in Figures 5.10 to 5.13.

A single iteration of a population with 30 individuals approximately takes 3 seconds on a P4 2.4GHz CPU and iterations to reach a fitness value of 98 were approximately 510 for the fist, 950 for the second, 800 for the third and 1350 for the last case as seen on Figures 5.10, 5.11 and 5.12. These indicate a total run time of 25.5 minutes for the first case, 47.5 minutes for the second case, 40 minutes for the third case and 67.5 minutes for the fourth case.

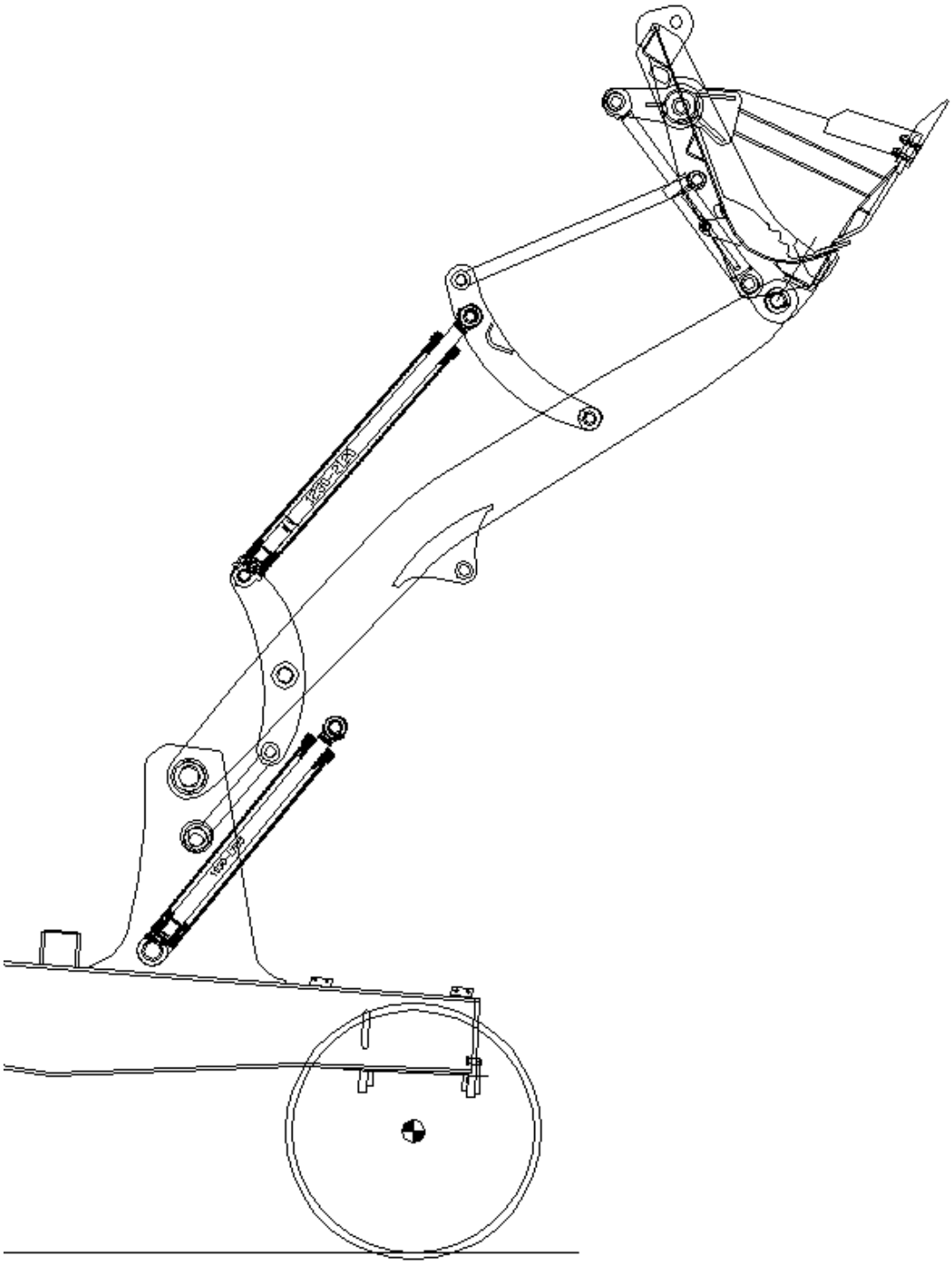

Figure 5.5 – Mechanism #2 at dumping position.

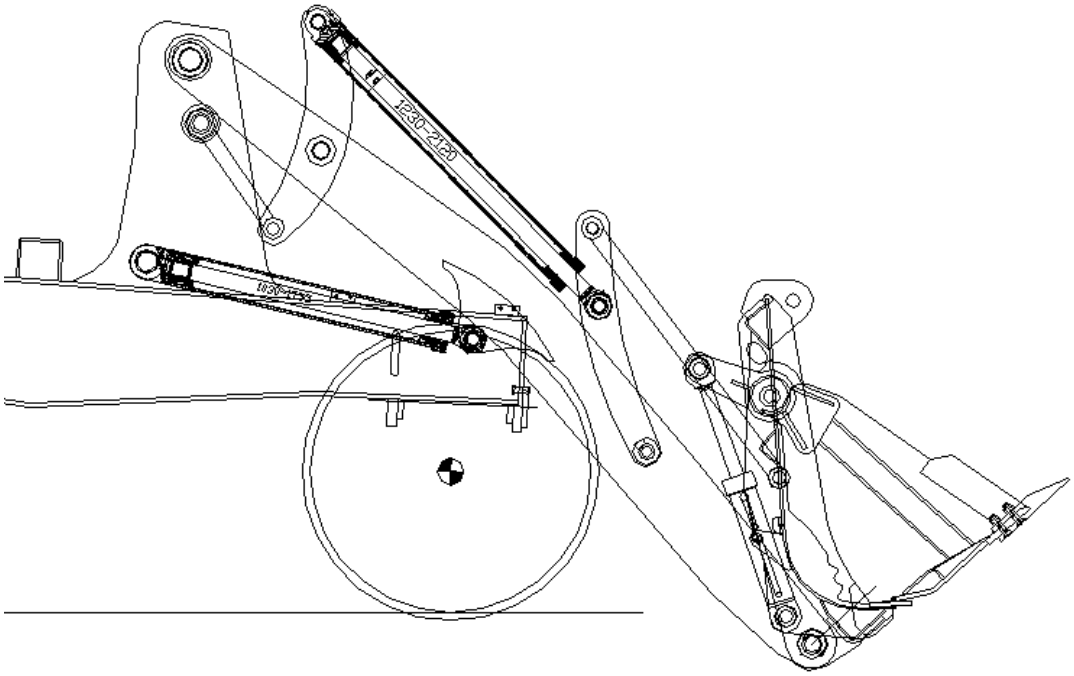

Figure 5.6 - Mechanism #3 at lowered position.

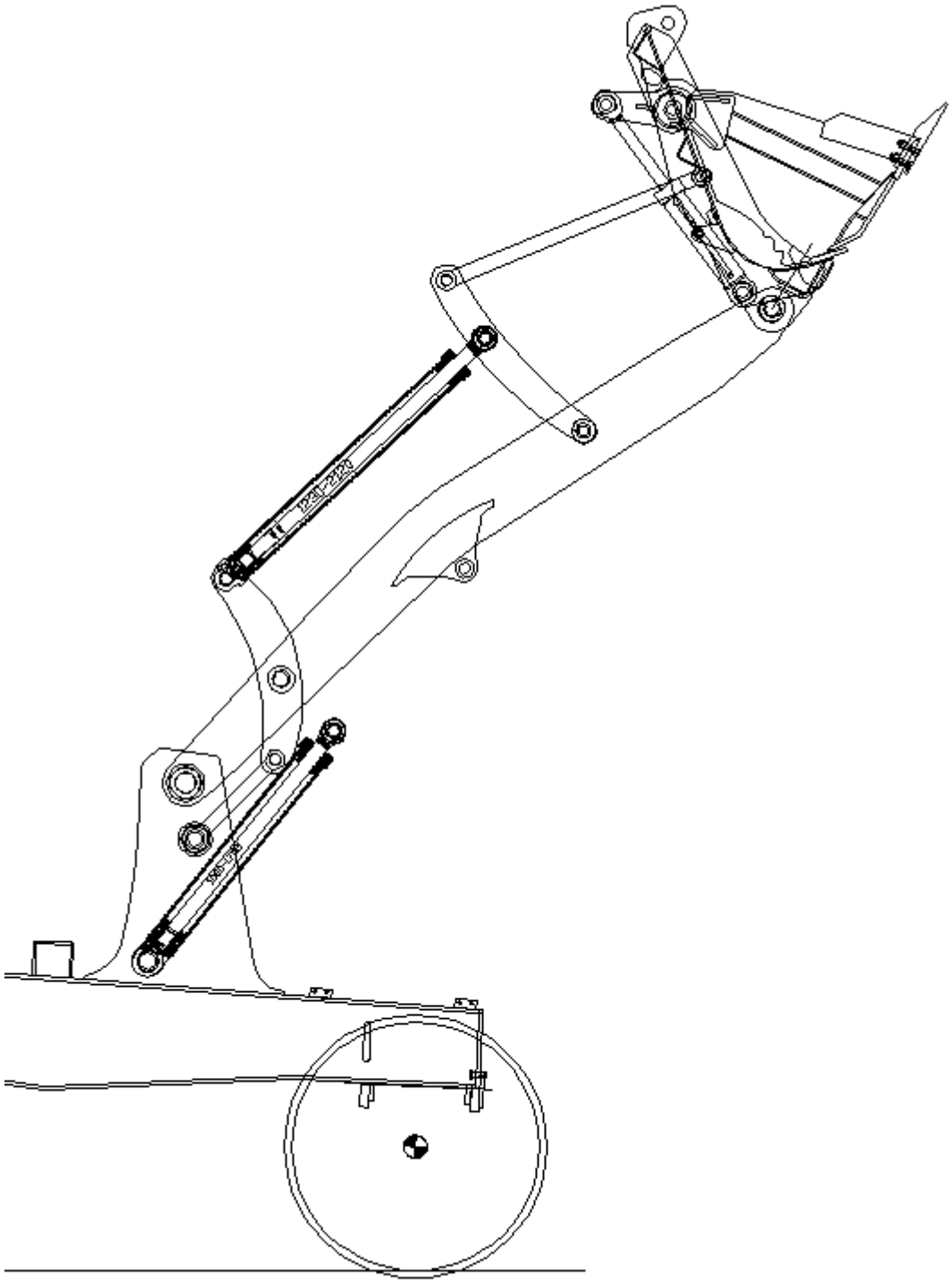

Figure 5.7 - Mechanism #3 at dumping position

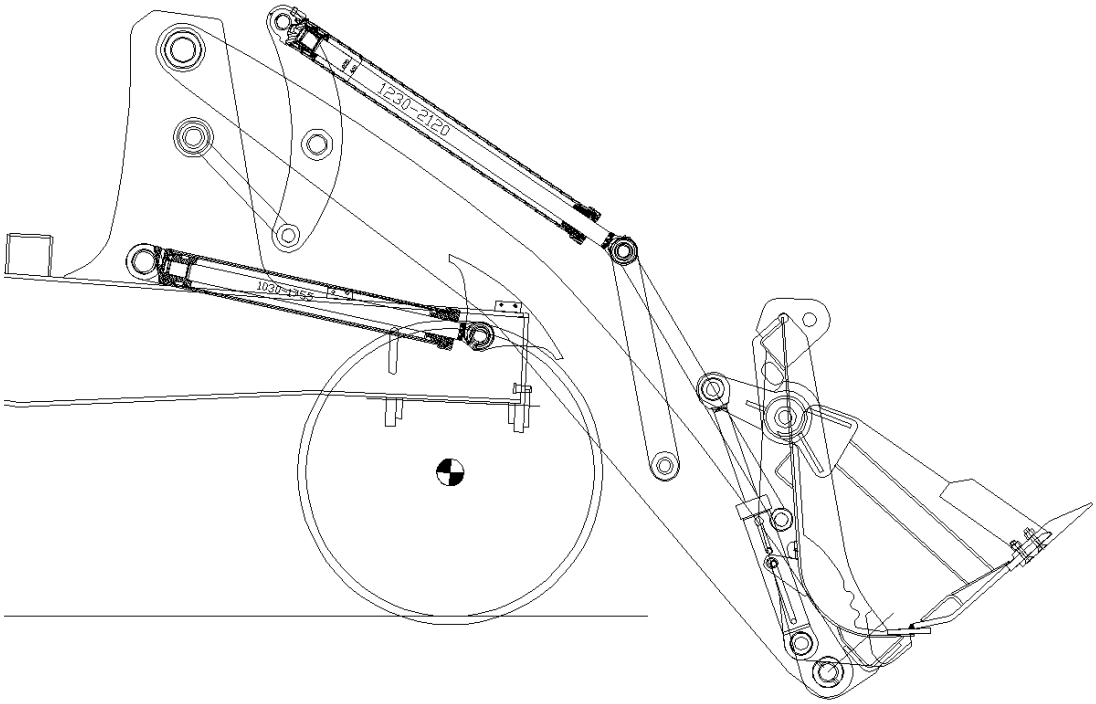

Figure 5.8 - Mechanism #4 at lowered position

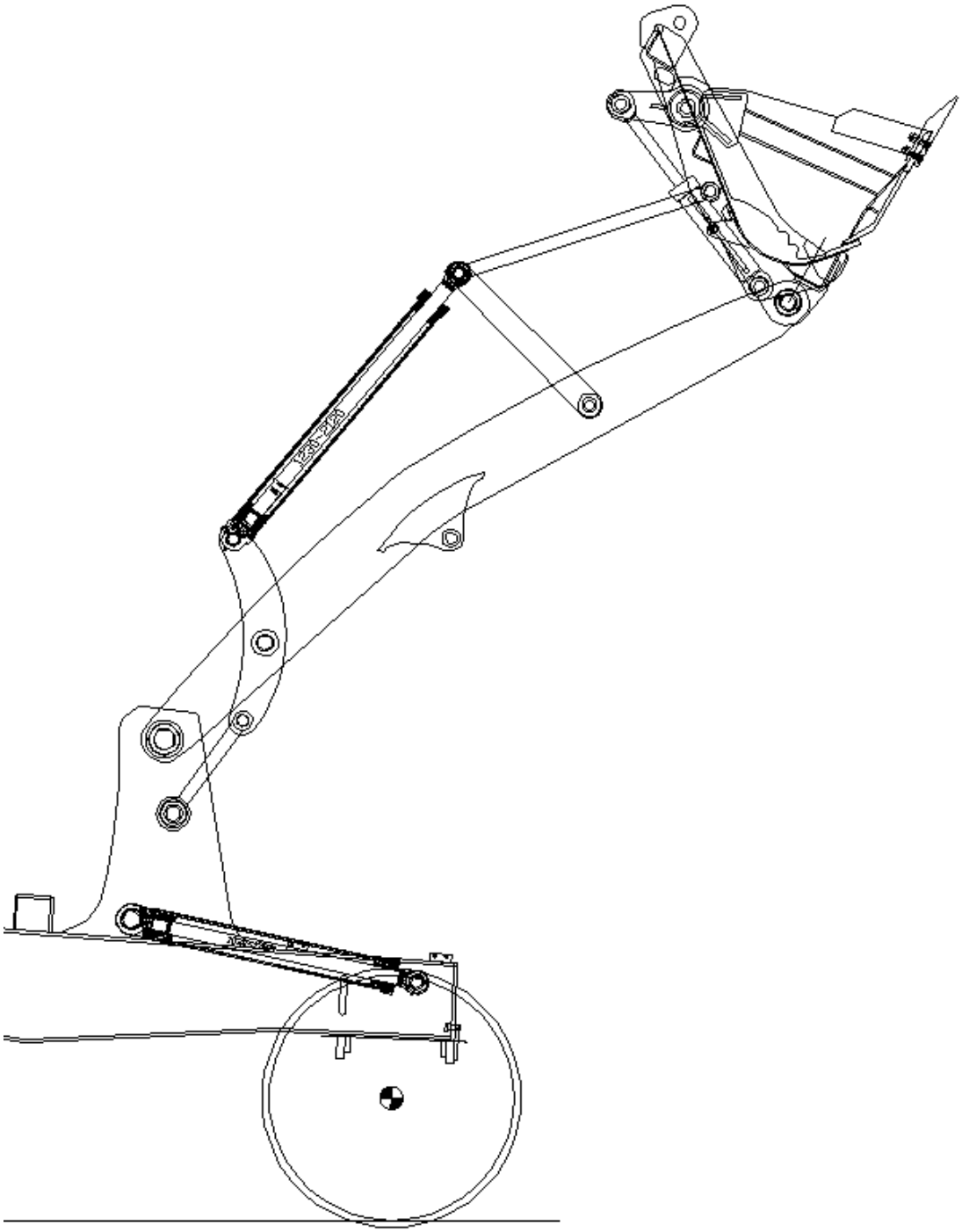

Figure 5.9 - Mechanism #4 at dumping position

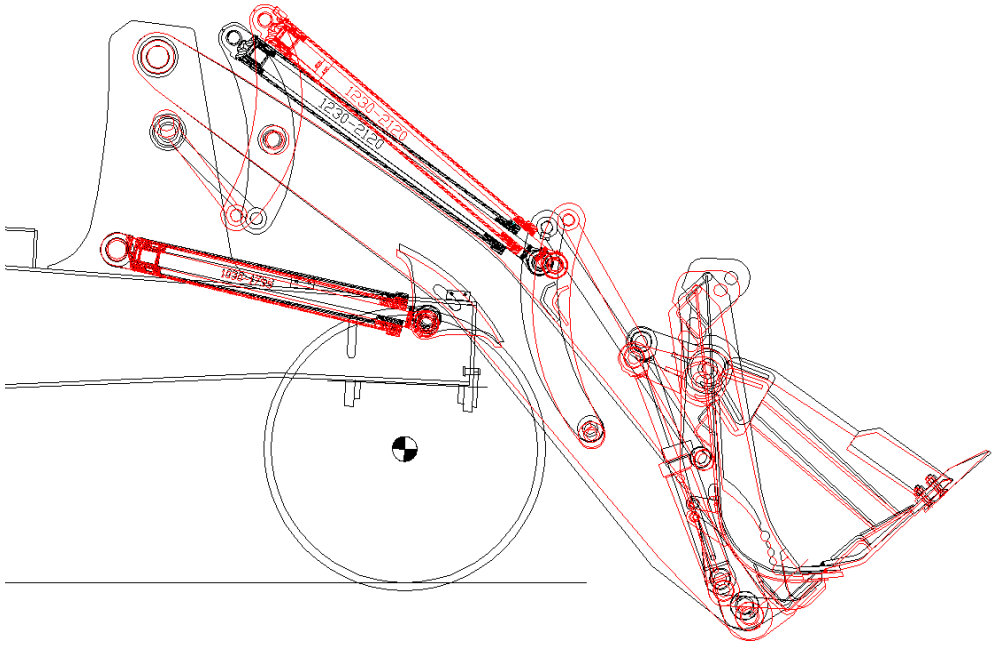

Figure 5.10 – Comparison of initial mechanism with #1

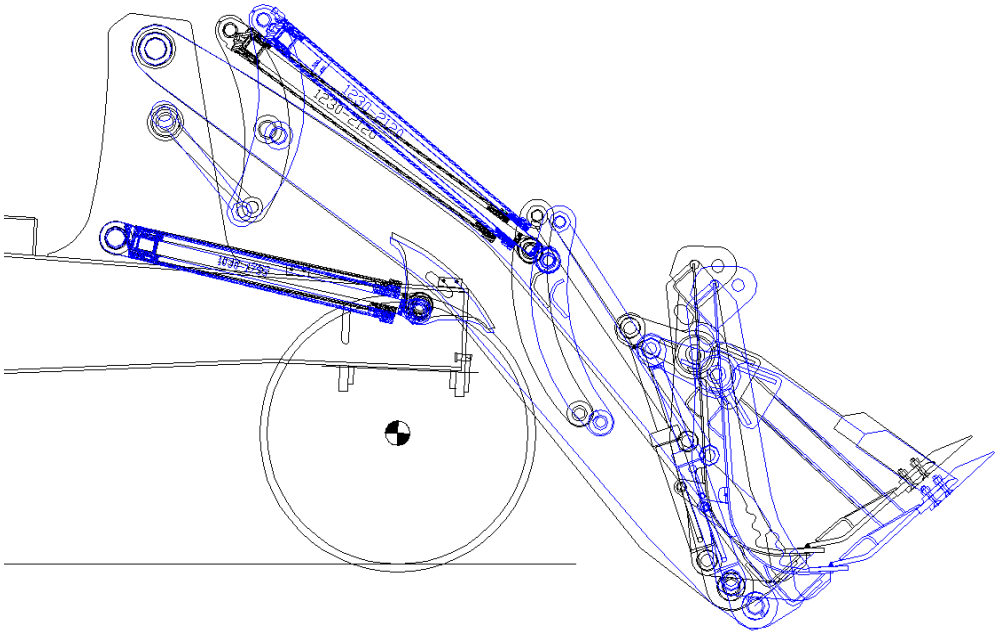

Figure 5.11 – Comparison of initial mechanism with #2

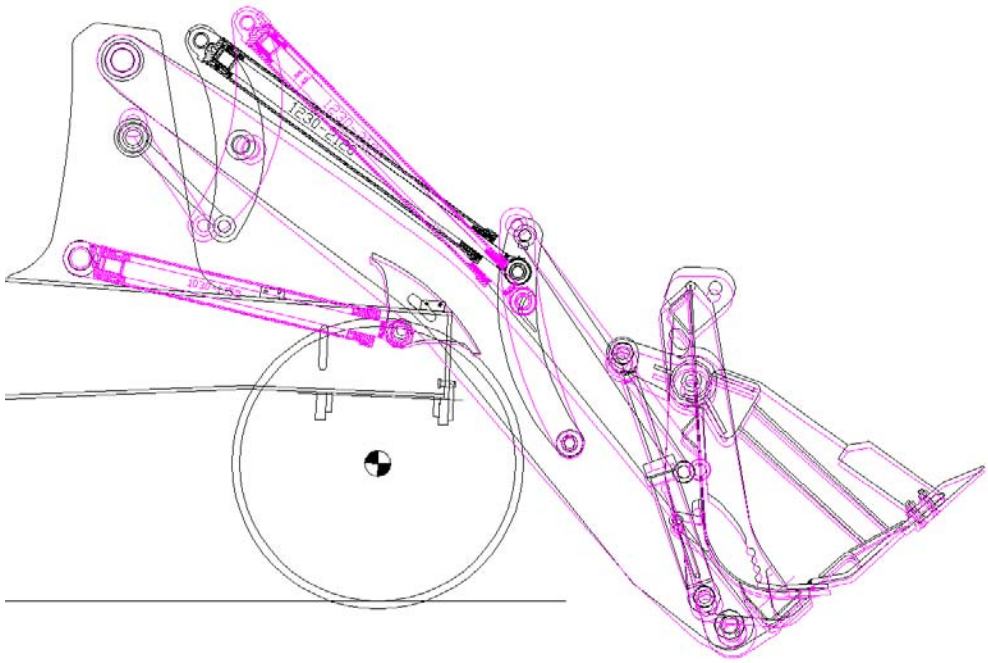

Figure 5.12 – Comparison of initial mechanism with #3

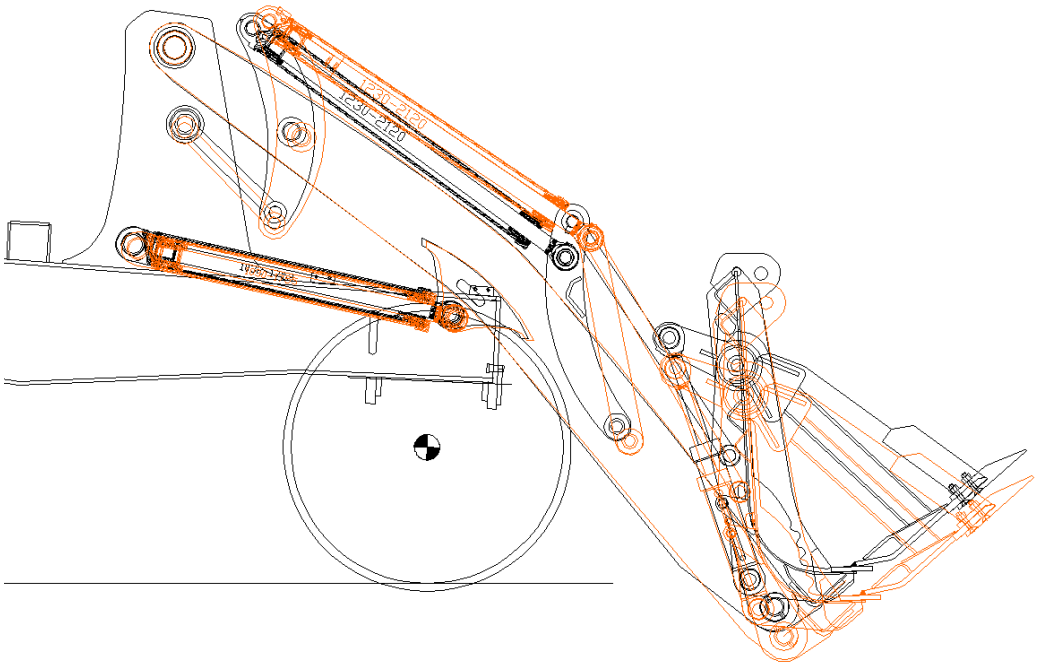

Figure 5.13 – Comparison of initial mechanism with #4

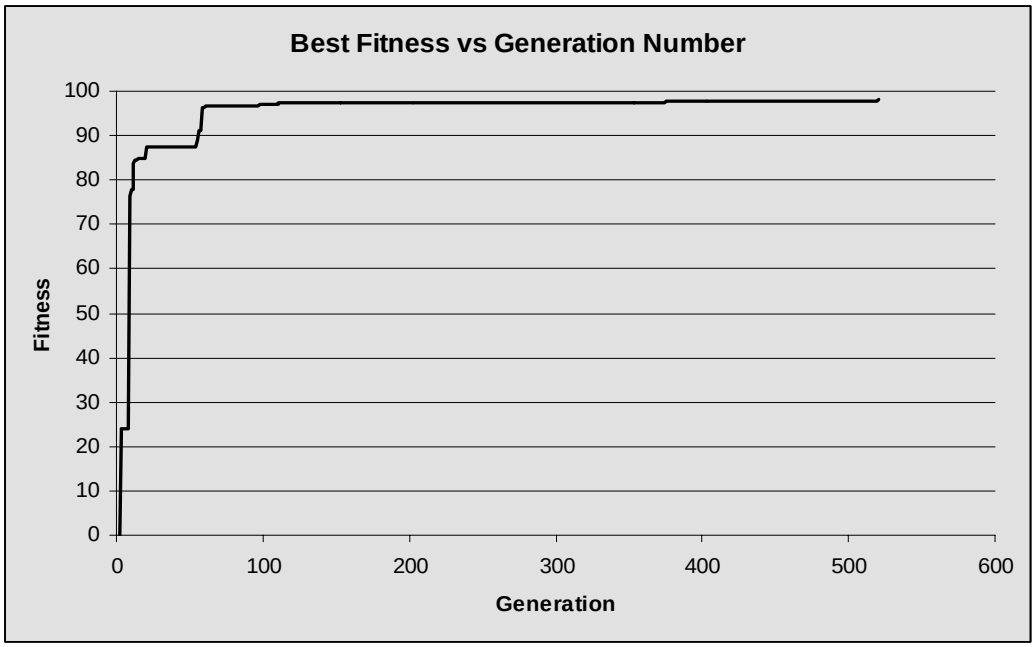

Figure 5.14 – Best fitness vs. generation number plot for #1

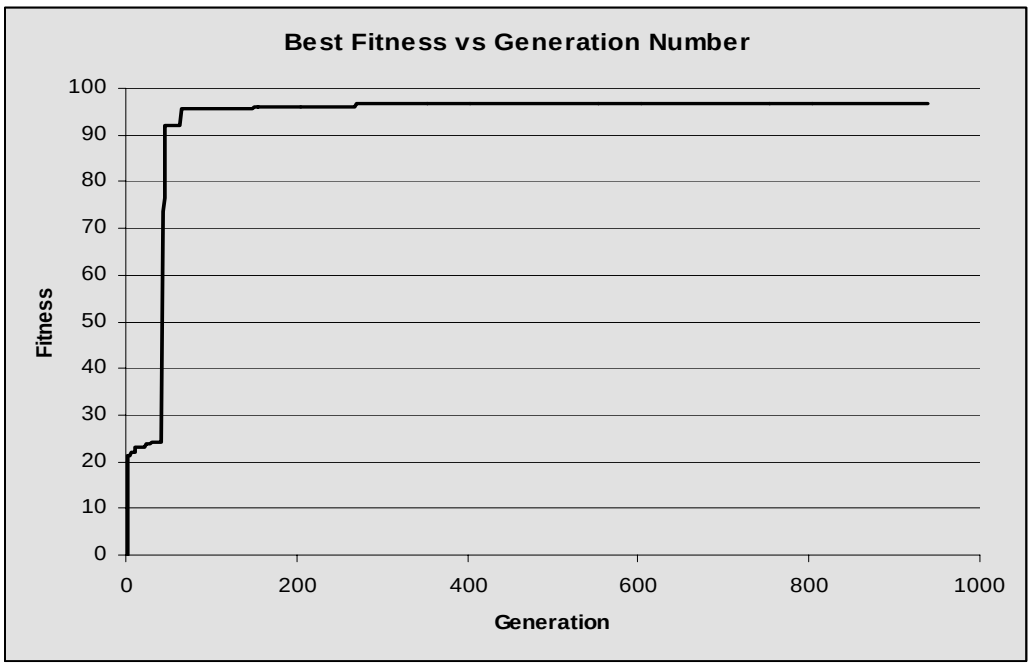

Figure 5.15 – Best fitness vs. generation number plot for #2

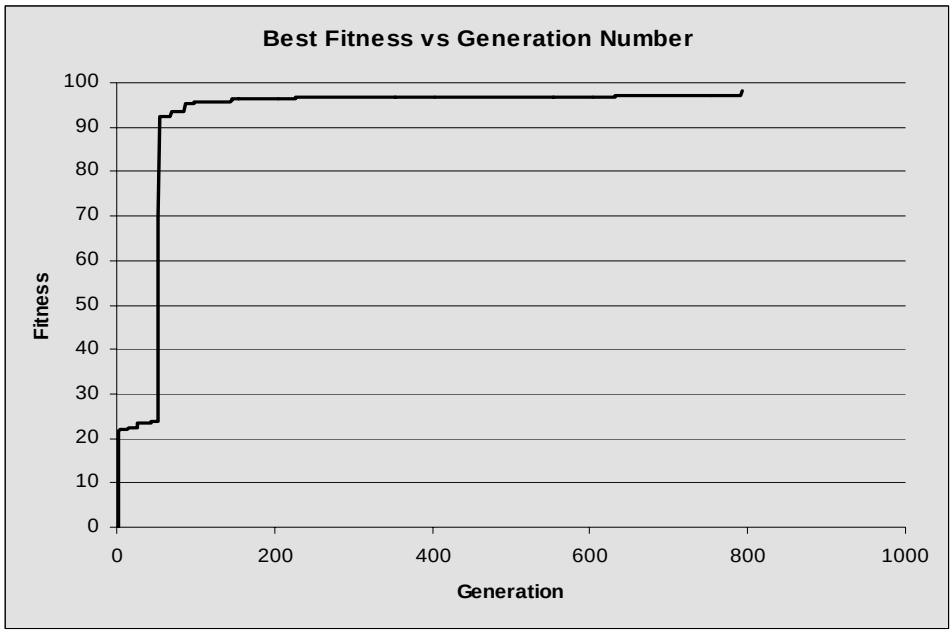

Figure 5.16 – Best fitness vs. generation number plot for #3

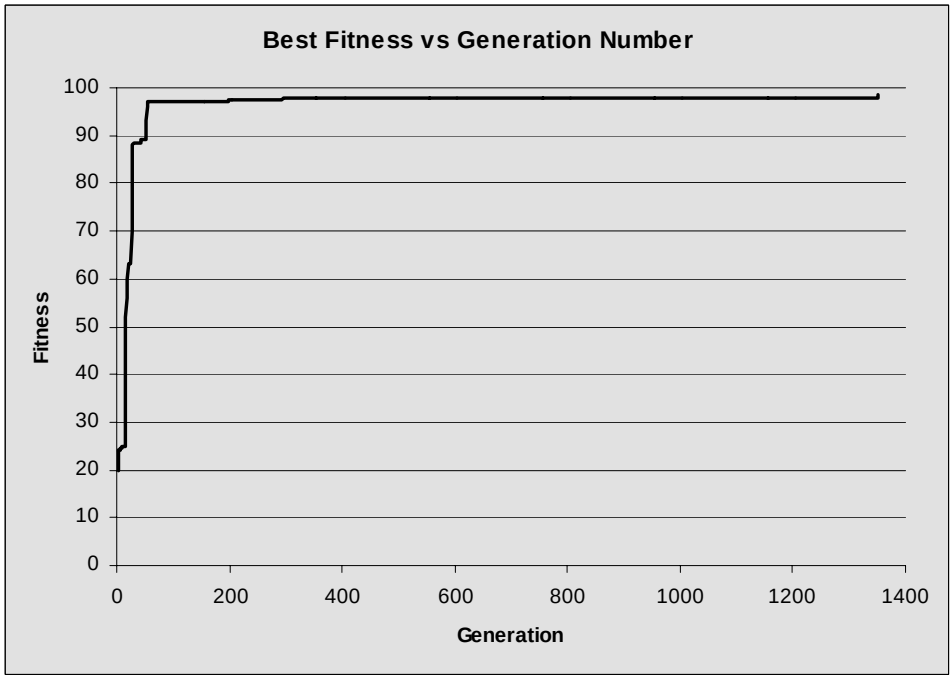

Figure 5.17 – Best fitness vs. generation number plot for #4

CHAPTER 6

DISCUSSION AND CONCLUSIONS

In this study a Genetic Algorithm in an Excel workbook by means of Excel macros is implemented to improve loader mechanisms.

Loader mechanism is an 11 link complex mechanism and it is desired to improve more than one output while also satisfying constraints. So, classical methods of synthesis cannot be applied. Genetic Algorithm is selected because it is suitable for multi-criteria optimization problems with constraints.

In order to calculate fitness values of mechanisms generated by the Genetic Algorithm an Excel worksheet is prepared. Given the values of the mechanism parameters, this worksheet calculates all necessary output values for the mechanism such as arm breakout force, bucket breakout force, lifting capacity, digging depth and loading height. Genetic Algorithm interacts with this Excel worksheet to input generated mechanism parameter values and thereafter read the calculated outputs in order to calculate fitness values for each mechanism generated.

Some of the constraints on the mechanism are implemented by adjusting the range of the random numbers generated for the parameter values of the first population; others are implemented in the fitness function by assigning a low fitness value to those mechanisms which violate one or more of the constraints. By this approach mechanisms that violate the constraints are not deleted immediately and still have a

low chance to be selected for crossover, so they may still yield a high fitness mechanism after a crossover operation.

A case study has been conducted to test the program and sample runs are made. Initial parameter values used for these runs are read from the current backhoe-loader machine of Hidromek Ltd. Even though this mechanism was well studied and improvements were made in time, it was still possible to improve it with the Genetic Algorithm of this work.

Among the results of the sample runs four mechanisms are selected that were able to move without linkage interference. The best one of the four runs has achieved 13.7 % increase in arm breakout force, 6.8 % increase in bucket breakout force and 2.7 % increase in lifting capacity while still achieving the required digging depth and dumping height constraints.

REFERENCES

[1] R. Sancibrian, P. García, F. Viadero, A. Fernández, A general procedure based on exact gradient determination in dimensional synthesis of planar mechanisms, Mechanism and Machine Theory Mechanism and Machine Theory 41 (2006) 212–229.

[2] J.A. Cabrera, A. Simon, M. Prado, Optimal synthesis of mechanisms with genetic algorithms, Mechanism and Machine Theory 37 (2002) 1165–1177.

[3] M.A. Laribi, A. Mlika, L. Romdhane, S. Zeghloul, A combined genetic algorithm–fuzzy logic method (GA–FL) in mechanisms synthesis, Mechanism and Machine Theory 39 (2004) 717–735.

[4] Habib Youssef, Sadiq M. Sait, Hakim Adiche, Evolutionary algorithms, simulated annealing and tabu search: a comparative study, Engineering Applications of Artificial Intelligence 14 (2001) 167–181.

[5] Gábor Renner, Anikó Ekárt, Genetic algorithms in computer aided design, Computer-Aided Design 35 (2003) 709–726.

[6] P.S. Shiakolas, D. Koladiya, J. Kebrle, On the optimum synthesis of six-bar linkages using differential evolution and the geometric centroid of precision positions technique, Mechanism and Machine Theory 40 (2005) 319–335.

[7] Ashim Bose, Maria Gini, Donald Riley, A case-based approach to planar linkage design, Artificial Intelligence in Engineering 11 (1997) 107-119.

[8] A. Kunjur, S. Krishnamurty, A Robust multi-criteria optimization approach, Mech. Mach. Theory Vol. 32, No. 7, pp. 797-810, 1997.

[9] Fernández-Bustos, J. Aguirrebeitia, R. Avilés, C. Angulo, Kinematical synthesis of 1-dof mechanisms using finite elements and genetic algorithms, Finite Elements in Analysis and Design 41 (2005) 1441–1463.

[10] A. Kanarachos, D. Koulocheris, H. Vrazopoulos, Evolutionary algorithms with deterministic mutation operators used for the optimization of the trajectory of a four-bar mechanism, Mathematics and Computers in Simulation 63 (2003) 483–492.

[11] H. Zhoua, Edmund H.M. Cheung, Analysis and optimal synthesis of hybrid five-bar linkages, Mechatronics 11 (2001) 283–300.

[12] Euan W. McGookin, David J. Murray-Smith, Submarine manoeuvring controllers' optimisation using simulated annealing and genetic algorithms, Control Engineering Practice 14 (2006) 1–15.

[13] A. Sadegheih, Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance, Applied Mathematical Modelling 30 (2006) 147–154.

[14] Zhang Wei Boa, Lu Zhen Huaa, Zhu Guang Yu, Optimization of process route by Genetic Algorithms, Robotics and Computer-Integrated Manufacturing 22 (2006) 180–188.

[15] D.M. Peng, C.A. Fairfield, Optimal design of arch bridges by integrating genetic algorithms and the mechanism method, Engineering Structures 21 (1999) 75–82.

[16] Xuejun Tan, Bir Bhanu, Fingerprint matching by genetic algorithms, Pattern Recognition 39 (2006) 465 – 477.

[17] J. Lampinen, Cam shape optimisation by genetic algorithm, Computer-Aided Design 35 (2003) 727–737.

[18] A.E. Baumal, J.J. McPhee, P.H. Calamai, Application of genetic algorithms to the design optimization of an active vehicle suspension system, Comput. Methods Appl. Mech. Engrg. 163 (1998) 87-94.

[19] Angel Kuri, An alternative model of genetic algorithms as learning machines, Expert Systems with Applications 15 (1998) 351–356.

[20] Arun Kunjur, Sundar Krishnamurty, Genetic Algorithms in mechanism synthesis, www.ecs.umass.edu/mie/labs/mda/mechanism/papers/genetic.html, last accessed on 04.2006.

[21] David E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company Inc., USA, 1989.

[22] Jasbir S. Arora, Introduction to Optimum Design, McGraw Hill Book Co., Singapore, 1989, p. 45.

[23] 1995 SAE Handbook, Volume 3 On Highway Vehicles and Off-Highway Machinery, Society of Automotive Engineers, Inc., USA, 1995, p. 40.71

[24] Eres Söylemez, Mechanisms, Middle East Technical University Press, Ankara, 1999, p. 16.