Organization Design Optimization using Genetic Programming

Bijan KHosraviani
Department of Civil and Environmental Engineering
Stanford University
Stanford, California 94305
bijan@stanford.edu

ABSTRACT

This paper describes how we use Genetic Programming (GP) techniques to help project managers find near optimal designs for their project organizations. Our GP model is a postprocessor optimizer for Virtual Design Team (VDT), an organization project design simulator also developed at Stanford. Decision making policy and individual/sub-team properties are varied by GP, and the effect on quality, and duration of the project is compared via a fitness function. The solutions found by GP are compared with the best human generated designs.

1. Introduction and Overview

In the complex and rapidly changing business environment of the early 21st Century, designing an effective and optimized organization for a major project is a daunting challenge. Project managers have to rely on their experience and/or trial and error to come up with organizational designs that fit their particular projects. The Virtual Design Team (VDT) simulation system, based on the information processing theories of Galbraith (77) and March and Simon (58), was a successful attempt to develop an analysis tool for project organization design (Levitt 96). VDT now enables project managers to model and analyze project organizations before implementing them in practice. After extensive ethnographic research in engineering organizations to calibrate its parameters, VDT can predict the schedule, cost and quality performance for a user-specified organization and work process.

However, like the analysis tools that support many other design processes, VDT has no inherent ability to improve or optimize current designs automatically. The user must experiment in a "What if?" mode with different alternatives in an attempt to find better solutions that can mitigate the identified risks for a given project configuration. Based on her or his expertise, the user must set up the model, run the simulator, analyze the output, make changes to the input, and repeat these steps until an acceptable output is achieved. The problem has many degrees of freedom, so the search space for better solutions is vast. Exploring this space manually is daunting. VDT relies on the expertise of the human user, and offers no guarantee of optimality.

So, our goal is to design and develop a post-processor for the VDT that uses evolutionary methods such as genetic programming to generate the optimal or near optimal project design.

2. Background

Organizational design is a complex global optimization problem involving both continuous and discrete variables. For example, an organizational designer must size functional capabilities, assign staff to tasks, and set communication and control policies.

Traditionally, project managers design project organizations intuitively. Success of project design has been a function of the manager's experience and how effectively the person can apply relevant past experience. Project managers' only means to discover whether or not a project organization will be successful has been to implement their projects and then evaluate their success or failures by trial and error.

This traditional method of project management design is very costly. Based on Tatum's empirical research, managers adapt personal experience as the primary process in organizational structuring. They repeat successes, avoid failures, and make adjustments as required by project situation (Tatum, 1983). The introduction of the Virtual Design Team (VDT) system has helped project managers to analyze their project design by means of

simulation in advance of implementation and thereby to construct better organizations. Instead of applying their design to the actual project, they can ”run the projects” virtually and see the outcome of their design on their computer screen.

Although Tools like VDT represent a big step forward in helping managers design their project organizations, it remains a challenging task for a project manager to find the optimal or near-optimal organization design for a specific project. In other words, usefulness of the VDT tool still depends on the experience of the modeler, and there is no guarantee that an optimized solution can be found.

2.1. VDT Background

Virtual Design Team (VDT) is a project modeling, design, and simulation tool that integrates organizational and process views of strategic, time-critical projects. The vision behind VDT is to provide a tool and method to design an organization the way an engineer designs a bridge, that is, by first creating and analyzing a virtual model, and then implementing the organization that has predictable capabilities and known limits.

Using VDT a user can develop a case of his project or projects and run simulations to see project time lines. Simulations also identify risks to development quality, schedule, and cost. Software simulation helps the user to set up, monitor, and troubleshoot a large project or a program of projects successfully. By altering the VDT planning components, a modeler can experiment with different solutions to determine which one meets his program quality, cost, and scheduling objectives.

As shown in figure 1, VDT provides a user-friendly interface by which a project manager can graphically design the organizational structure, namely its size, the number of people in the group, and its topology, who reports to whom. The project manager also graphically assigns one or more activities for each individual within the group, as well as the dependencies between the activities.

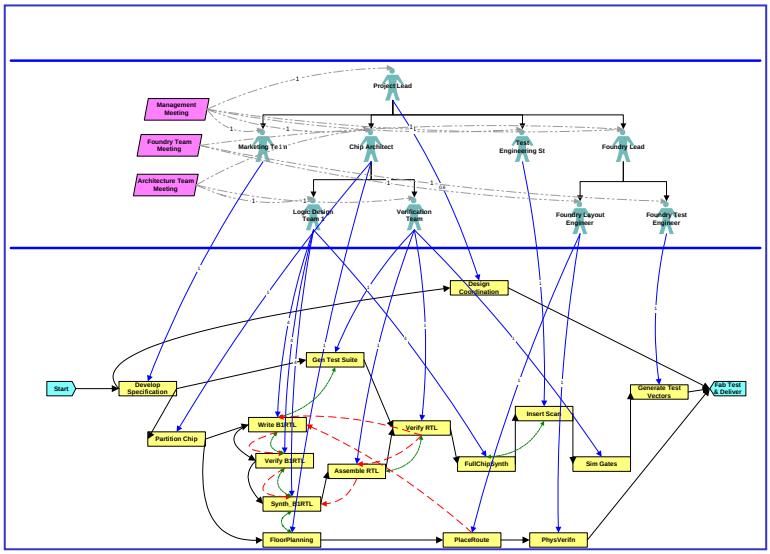

Figure 1 User Interface of the VDT simulator – Each project participant fills a position in the project organizational hierarchy and works on one or more activities. The organizational structure and the interdependence between activities define relationships among individuals.

A user can also set other organization attributes such as skill level of each Actor (individual within the organization) and decision making policies. The skill level of each Actor can be set to three levels of low, medium, and high. The higher the skill level of individuals, the faster the task can get done. Decision making policies include *centralization, formalization* and *matrix strength. Centralization* reflects whether decisions are made by senior management positions or decentralized to first level supervisor or worker positions. *Formalization* is the relative degree to which communication among positions takes place through formal meetings and memos *vs.* informally. *Matrix strength* models the “degree of connectedness” of the various specialists in an organization by setting the probability that workers will attend to information. The above decision making policies can also be set to three levels -- low, medium, and high.

Once the above attributes and topologies are set, the discrete event simulation can be run (usually 50-100 run is sufficient) and the model produces a set of output as shown in Fig. 2. The user can then manually adjust the input parameters to obtain the desired output.

Figure 2 Sample of VDT outputs

Gant charts, quality risks, and person backlogs are among number of graphical outputs that VDT can produce. Gant Chart displays project, tasks, and milestone in rows with duration-represented bars. Quality Risk shows the task or projects at greatest risk of exception-handling failures. Actor backlog shows the backlog for each person in the model, which indicates predicted overload of positions over time.

3. Statement of the Problem

As we see from the above, even if we don't consider the reporting structure of the organization, the number of combinations that one needs to try to find the optimal solution for a project organization with only 10 positions would be 3^3 x $3^{(2x10)}$ = 282,429,536,481different cases. (i.e. 3^4 because Centralization, Formalization, Matrix Strength can each take three values of low, medium, and high; and $3^{(2x10)}$ considering each individual/sub-team on average has two skills, which skill level of each can take three values of low, medium, and high.) Thus, it takes an enormous amount of time and computing power if one wants to use a random search to find the optimal solution.

Since the number of different input attributes that can be changed is enormous, we decided to limit the focus of this project to optimizing a case study of a biotech plant that is currently used in one of the project courses at Stanford. The problem is defined as reducing the schedule time-line for biotech project pre-construction activities, while improving the risk qualities of the project. The following constraints should be considered:

1. Adding a total of up to 3 Full Time Equivalent's (FTE) in increments of not less than 0.5 FTE to any combination of actors.
2. Increasing the skill level (from low to medium, or medium to high) for any one skill for any one actor.
3. Changing levels of Centralization, Formalization, or Matrix strength
4. Reassigning tasks to different actors or change the assignment % that an actor is allocated to an activity.

The objective is to shorten the simulation duration while maintaining acceptable quality risk. Because of the time limitation of this project, reassigning tasks, item #4 above was not implemented.

4. Methods

The method used in this paper is similar to what has been used in designing an improved version of Astrom-Hagglund PID controller (Koza 2003). Instead of redesigning a project organization from scratch, we used an existing design done in the traditional human generated way and tried to adjust different attributes, so the final outcomes of the project could be improved. In order to do this, the genetic transforming tree produces a solution that in fact is an instruction of what, in the given project organization, needs to be changed and by how much.

4.1 Software/Hardware used

The runs reported in this paper were designed based on the standard genetic programming paradigm as defined in Koza (1992). The problem was coded in Java using the ECJ 10, the Evolutionary Computation and Genetic Programming System by Sean Luke. The runs were executed on a PC with a Pentium 4 - 2.8GHz processor.

4.2 The Genetic Transforming Tree

The remainder of this section explains the setup for the genetic programming tree and how it used to produce set of solution to the given problem. The standard genetic programming tableau appears in Table 1.

Table 1 Tableau for the project organization design optimization problem

Objective:	Find the changes need to be made to the current project organization in order to reduce the project simulated duration, reduce cost and improve quality of the final outcome
Terminal Set	P1, P2, P3, P4, P5, P6, P7, CFM
Function Set	Up, Down, Same, FTE
Fitness Cases	15 total – 1 for simulation duration, 1 for FTE, 13 for each activities
Raw Fitness	SPD + TFTE * FTEW + ∑ (FRI(i) * FRIW(i) + PRI(i) * PRIW(i) + CR(i) * CRW(i)) (see description in section 4.5 Fitness Evaluation)
Standardized Fitness	Same as raw fitness
Parameters	Population size M = 1000 Maximum number of generations, G = 100 Crossover = 90% Mutation = 3% Reproduction = 7%
Success Predict	None – search for the shortest simulation duration with the given quality and FTE constraints

4.3 Function and Terminal Sets

Terminal set P1 through P7 represents Actors in the group. CFM stands for Centralization, Formalization and Matrix Strength. Function sets Up, Down, Same can have different meaning depending on the Terminals set that they connect to and whether there is an FTE function set in between. Function FTE increases or decreases the FTE amount of each individual depending on the number Up/Down preceding it in the genetic Tree.

- Function sets Up/Down increase or decrease the Actor's skill level by one level correspondingly. For example, from medium to high or from medium to low.
- Function set **Same** does not make any changes in Actor's skill level.
- Depending on where the above three function sets are located in the hierarchy, they affect the 1st, 2nd, or 3rd skill set of a person. For example:

Up
|
Down
|
Same
|
P2

Increases the skill set 1, decrease the skill set 2, and does not change the skill set 3 of Actor P2 accordingly.

- Function Set **Swap**, swaps the task assignments between the two Actors. If Actors have more than one task assigned to them, both get swapped.
- Function Set **FTE** (Full-Time Equivalent) increases or decreases the Actor FTE by 0.5 per each Up or Down preceding the FTE. For example:

Up
|
Up
|
FTE
|
P5

Adds 2 x .05 = 1 FTE to Actor P5 current FTE.

Terminal Set **P1 through Pn** represents all people (Actors) in the organization.

Terminal Set **CFM** stands for Centralization, Formalization and Matrix Strength. Once again Up, Down, Same will affect the above 3 parameters accordingly. For example:

Down
|
Up
|
Same
|
CFM

Would bring the Centralization one level down, increase the formalization by one level, and does not change the Matrix Strength level.

A combination of above function and terminal sets transforms the initial project organization suggested by a project manager to a near optimal one. Figure 3 shows a sample of a program tree produced by this genetic operation. In this configuration, for example, the skill attributes of P4 (person 4) in the organization structure changes based on the type of its parent and grandparent nodes. So, in this case, P4's first skill level (e.g. Project Management) increases, her second skill level (e.g. Software Engineering) decreases, and her third skill level (e.g. Design Coordination) remains the same. In another branch in the tree below function set "Swap" lie the parents of P3 and P5. In this case, the genetic tree suggests that the activities of P3 and P5 should get swapped in order to optimize project performance. In the sample tree below CFM's parents and grandparents are "Up", "Down", and "Up". So, in this case, the genetic program tree suggests that centralization should increase by one level, formalization should decrease and matrix strength should increase as well to optimize the overall project outcome.

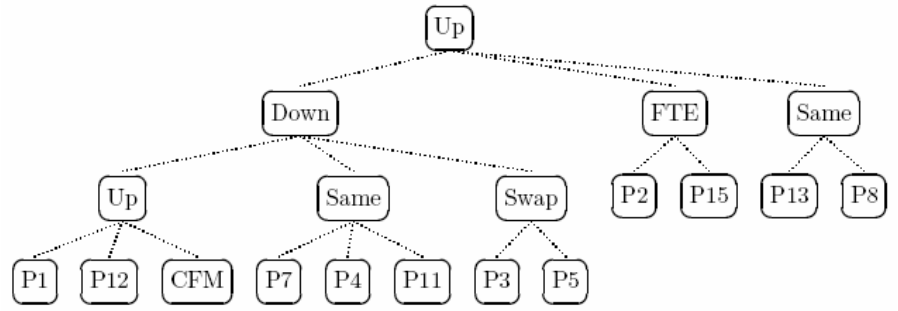

Figure 3—Sample of a transforming Genetic Tree. Program trees created by genetic operations modify the structure and attributes of a project organization. The genetic tree above trans-forms an organization design (not shown here) proposed by a project manager to a near optimal one.

4.3 Genetic Tree Constraints

Since FTE and SWAP function sets can not appear anywhere but next to the bottom of the genetic tree (i.e., Terminal sets should be the only children of the FTE and SWAP, and Up, Down, and Same can not be the children of the two), constraints were added to the ECJ parameter files to enforce such limitations. Once the constraints were added the genetic program only produced valid genetic trees, as shown above.

4.4 Integrating the system as a whole

Several small programs were written in Java, so the genetic tree can be transformed in a "useful-sense" to rules that can change the base XML file so it can be read by VDT. The modified XML file was read into VDT, a simulation was run, and the result was exported in another XML file. The fitness read the result and produced the final fitness value. Figure 4 displays a flow chart that represents how the system was integrated as a whole.

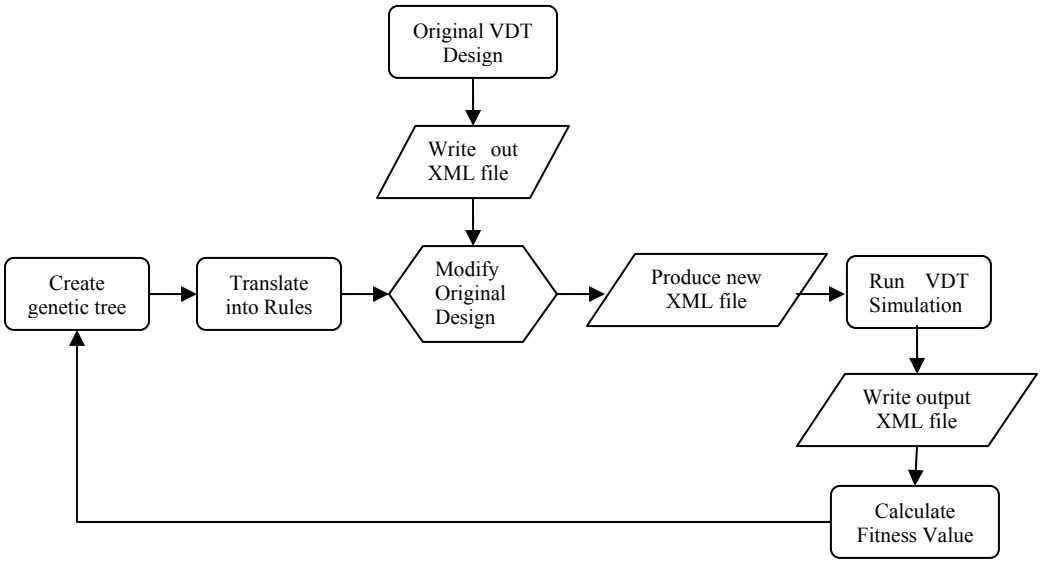

Figure 4—Flow Chart of the system as a whole

4.5 Fitness Evaluation

There are three areas of focus in calculating the fitness function. First is the total simulation duration, which is the number of days that take to finish the project. Second are the quality risk values such as communication risk, Functional Risk Index (FRI) and Project Risk Index (PRI). Third is the Total FTE since there was a constraint for the maximum number of total FTE allowed to be added. The total FTE added and the quality risk value were added to the total fitness, and they were both penalized heavily if they exceeded their given limits. For example, if total FTE exceeded 3.0 or each one of the quality risk values were over 0.5, the values were multiplied by 1000. So, the total fitness value was calculated based on the following formula:

$$\text{Fitness Value} = \text{SPD} + \text{TFTE} * \text{FTEW} + \sum_{i=1}^{M} (\text{FRI}(i) * \text{FRIW}(i) + \text{PRI}(i) * \text{PRIW}(i) + \text{CR}(i) * \text{CRW}(i))$$

Where

- SPD = Simulated Project Duration
- TFTE = the Total FTE added
- FTEW = FTE Weight (if TFTE > 3.0 => equal 1000 otherwise 1)
- FRI(i) = Functional Risk Index for activity i
- FRIW(i) = FRI weight for activity i (if FRI(i) > 0.5 => equal 1000 otherwise 1)
- PRI(i) = Project Risk Index for activity i
- PRIW(i) = PRI weight for activity i (if PRI(i) > 0.5 => equal 1000 otherwise 1)
- CR(i) = Communication Risk for activity i
- CRW(i) = CR weight for activity i (if CR(i) > 0.5 => equal 1000 otherwise 1)
- M = maximum number of activities

5. Results

The best individual of the first one hundred generations is shown below in a lisp-type format:

```
(Up (Up P3 (FTE P5 P1) (Up (Same P4 (Up CFM P2 P0) (Up P3 P2 (Up (Same P4 (Up
CFM P2 P0) P6) P5 (Down P5 P5 P4)))) P5 (Up (Same P4 (Same (Up (Same (FTE P2
P0) P0 (Down P5 P5 P4)) P1 P3) (Same (FTE P2 P0) P0 (Down P5 P5 P4)) (Up
(Same (FTE P2 P0) P0 (Down P5 P5 P4)) (Up (Down P5 P5 P4) P5 (FTE P1 P4))
P2)) P6) P5 (Down P5 P5 P4)))) (Up (Up CFM P2 P0) P5 (FTE P1 P4)) (Same (FTE
P5 P1) (Up (FTE P2 P0) P1 P3) (Same (FTE P2 P0) P0 (Down P5 P5 P4))))
```

However, looking through the best of each generation, a shorter version of the transforming genetic tree could be found that produces similar results:

```
(Up (Up P3 P2 (Up (Same P4 (FTE P2 P0) P6) P5 (FTE P1 P4))) (Up (Up CFM P5
P5) P5 (FTE P1 P4)) (Same (FTE P5 P1) (Up (Same (FTE P2 P0) P0 (Down P5 P5
P4)) P1 P3) (Same (FTE P2 P0) P0 (Down P5 P5 P4))))
```

This best individual improved the original outcome of the simulation significantly. Comparison results are shown in Figure 5 through 7 below:

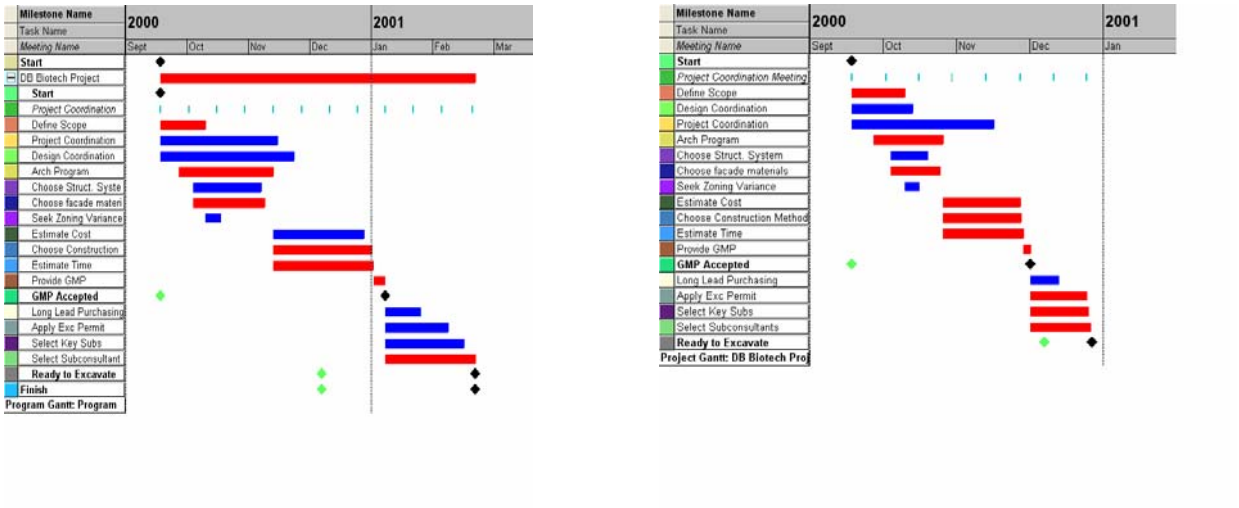

Figure 5— Comparison of Gant Charts between before (left) and after (Right) the evolutionary process. End project schedule was reduced from Feb 20, 2001 to Dec 27, 2000

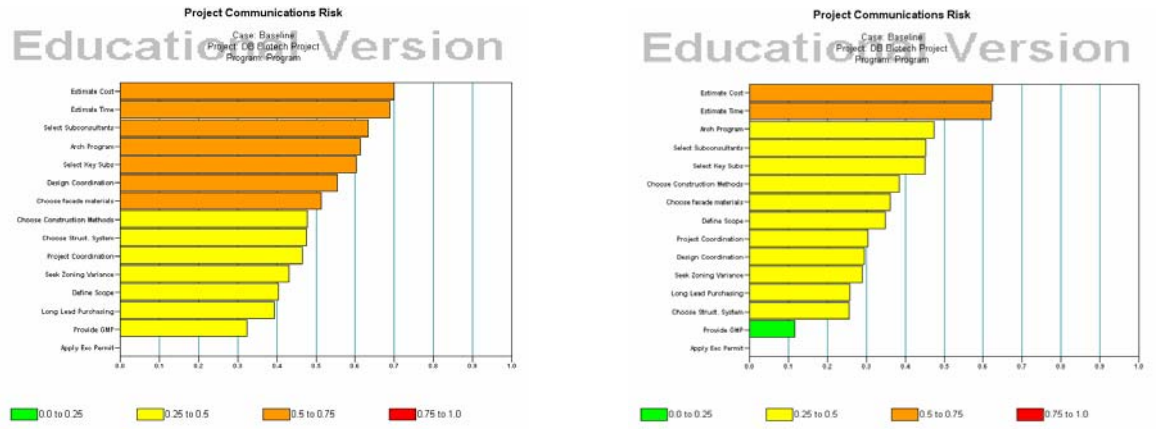

Figure 6— Comparison of Quality Risk Charts between before (left) and after (Right) the evolutionary process. Originally 7 out of 14 activities have quality risks higher in orange zone. With the suggested organizational changes quality risks for all activities improves. Two of the activities still remain the orange zone.

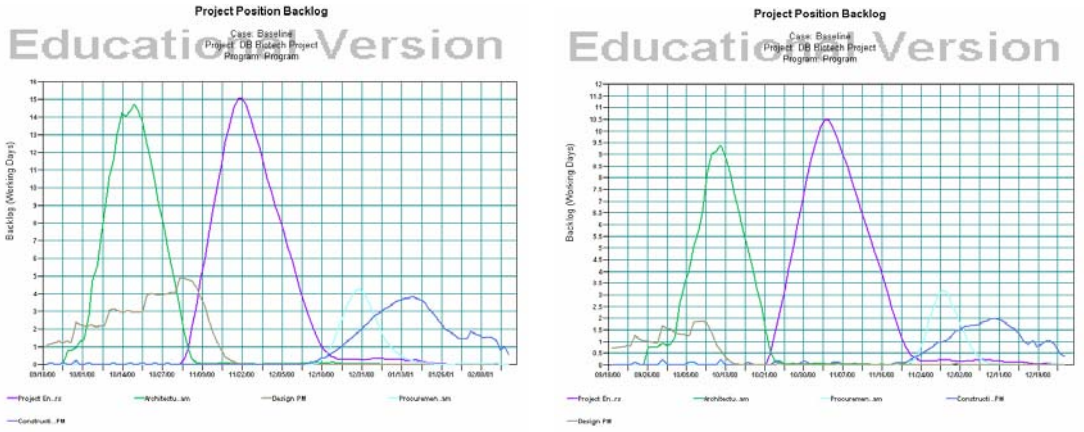

Figure 7— Comparison of backlog Charts between before (left) and after (Right) the evolutionary process. Backlog, of the majority of different positions are improved by a few days.

6. Discussion of Results

As shown in the above figures, genetic programming has been able to improve the final outcome in all three areas of concern and meet the given constraints. The greatest improvement is seen in the project schedule, where the simulated duration was reduced by 55 days. Due to the time limitation of this project, the Swap functionality was not implemented, and as a consequence, the final results could not be improved as much as expected. A solution found by a student group in a project course, for the same problem, was able to reduce the duration by 75 days. Students found that by only reassigning some tasks, they were able to reduce the simulated duration by 23 days.

However, in another trial, the suggested reassignment of the activities by the student group was used and added to the baseline design. Then, we ran the genetic operation again. The result matched the best solutions found by the students. For this trial the population size was 500, and the best individual was found after 24 generations.

Figure 8 below shows that the greatest improvement in the original problem was made between generations 1 and 2. From generations 3 to 8, fitness value almost linearly decreased. From generation 13 onward, not much improvement was made.

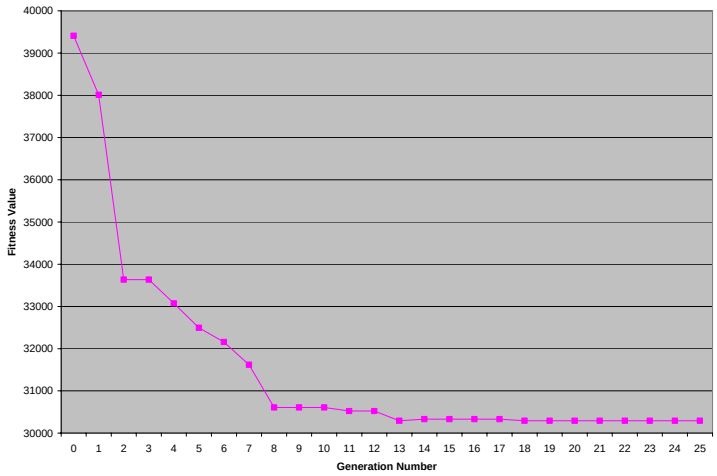

Figure 8— Improvement of fitness value generations 1 through 25

Several trials with different population sizes and different mutation and crossover rates were made, but this did not affect the final outcome significantly. Also, in one case, the number of FTE children was changed to one, and that also did not affect the results greatly.

Although the transforming genetic tree in its current form is shown to produce improved results, it might not be the most efficient. The tight dependencies of FTEs of each Actor on its preceding nodes can cause some deficiency during the crossover operations, where only part of a branch of one individual is swapped with another. Another factor is that when the recurrence of Actors occurred, the very last Actor to the right of the tree was used. Thus, after many generations, only the right side of the genetic tree was active.

7. Conclusions

This paper has demonstrated some successful first steps towards optimization of organization project designs. Instead of redesigning the organization project from scratch, a human generated design was used as a baseline. Several input attributes--such as policy properties, individual / sub-team properties, skill level, and FTE of Actors—were adjusted using genetic programming to improve the final project outcome. The effects of the evolutionary process on project simulation duration and quality risk were noticeable. A relative comparison was made between the results produced by humans and those of GP. However, a one-to-one comparison could not be applied in the main case since reassigning activities could not be implemented within the short duration of this project. For the case in which this was possible, the result produced by GP was comparable with those produced by humans.

8. Future Work

This work is only the beginning use of genetic programming in optimizing organizational design. Much time on this project was spent on writing the code to translate the transforming genetic tree and connecting the results to the VDT simulator. Now that the preliminary work is done, the next step is to add different organizational attributes to the genetic operation and see the effect on the final outcome. Reassigning tasks to different actors and changing the assignment percentage that an Actor is allocated to an activity should be among the first ones to be implemented. The communication properties—such as who reports to whom, team experience, and application experience—are some of the other parameters that should be added to the input variables. Eventually, the evolutionary post-processor should be integrated within the VDT model.

9. Acknowledgments

I would like to gratefully acknowledge the assistance and guidance of John Koza for this project. I would also like to thank the developer of ECJ, Sean Luke, for providing the easy-to-use programming environment that was used for this project, and responding to some of my specific questions regarding the genetic constraints. Last but not least, I would like to thank Mark Ramsey, for his assistance on helping me to connect the genetic tree to the VDT model.

Bibliography

Galbraith, J.R. 1977 Organizational Design. Reading, MA: Addison-Wesley.

Jin Y. and Levitt R.E. 1996 The Virtual Design Team: A computational Model of Project Organizations. Computational and Mathematical Organization Theory; 2(3):171-196

Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press.

Koza J.R. Keane M.A. Streeter, M.J. Mydlowec, W. Yu, J. Lanza, G. 2003. Genetic Programming IV-Routine Human-Competitive Machine Intelligence: Kluwer Academic Publishers.

March, J.G. and Simon H.A. 1958 Organizations. New York: John Wiley and Sons

Tatum C.B. Decision Making in Structuring Construction Project Organizations. Ph.D. Dissertation. Department of Civil and Environmental Engineering. Stanford University 1983.