Organization Design Optimization Using Genetic Programming

Bijan KHosraviani[1], Raymond E. Levitt[1], and John R. Koza[2]

[1] Stanford University
Department of Civil and Environmental Engineering
Stanford, California 94305
{bijan, Ray.Levitt}@stanford.edu
[2] Stanford University
Stanford Biomedical Informatics Program, Department of Medicine
Stanford, California 94305
koza@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. We use GP as a postprocessor optimizer for the project organization design simulator Virtual Design Team (VDT). Decision making policy and individual/sub-team properties, activity assignments and percentage allocation for each activity 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 compare favorably 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 best fit their particular projects. This traditional method of project organization 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 Virtual Design Team (VDT) simulation system, based on the information processing theories of Galbraith (1977) and March and Simon (1958), was a successful attempt to develop an analysis tool for project organization design (Jin & Levitt, 1996). 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 engineering 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 at-

tempt 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. VDT relies on the expertise of the human user, and offers no guarantee of optimality. The problem has many degrees of freedom, so the search space for better solutions is vast, and exploring it manually is daunting.

In this paper, we demonstrate how we have designed and used a post-processor for VDT that uses genetic programming, an evolutionary computing method, to generate near optimal project organization designs.

2 Motivation and Points of Departures

Over the past 50 years, optimizers have been successfully developed and deployed for a variety of analysis tools aimed at predicting the behavior of physical systems such as structures, engines, or semiconductors. These optimizers, in conjunction with mature and extensively validated analysis tools, have enhanced the productivity of engineers by orders of magnitude, and have expanded the range and enhanced the quality of products created in many fields of technology. In contrast, organizational analysis tools that can be used by managers to predict performance outcomes of alternative organizational configurations have only begun to emerge over the past decade. Starting with the Virtual Design Team research in the mid-1990s and the pioneering work of Burton and Obel (2004) in the late 1990s, there are now several agent-based computer models and rule-based diagnostic tools that help managers analyze candidate organizational configurations for a given set of task requirements and environmental constraints. However, to the best of our knowledge not much work has been done that can claim to optimize organization designs for real world organizations. One of the attempts we are aware of is a fuzzy multicriteria framework for the comparison of alternative organization structures of post corporations (Kujacic & Bojovic, 2003.)

Some attempts have been made in the past at developing a post-processor for VDT; however, those were of limited power and generality. For example, William Hewlett (2000) designed a rule-based "expert system" post processor for VDT that analyzes the outputs of a VDT simulation, and recommends small, incremental changes in the design of modeled organizations. Hewlett's post processor was tested in a design *charrette* on a group of Stanford students; it showed that they were able to create better organizations when they used the post-processor than without it. However, there were many limitations of this initial post-processor. First, the post-processor did not solve for the optimal organization; it was only a small piece of an optimization strategy. It primarily focused on team sizes and suggested reallocating personnel between teams. Thus, after running the VDT simulator, a user had to take advice suggested by the post-processor, make changes in the original design, run the simulator again, and observe whether the optimization was beneficial. Then the user had to repeat this optimizing loop until the desired output was achieved. As a result, this process was an exhaustive, never ending search. Second, although this process was shown to be beneficial for some students with less project management design experience, it provided less benefit for more experienced managers.

An ongoing research effort by Michael Murray, another PhD student in the Department of Civil and Environmental Engineering at Stanford, is beginning to address a few selected aspects of the organization design optimization problem. The focus of Murray's research, like Hewlett's, is on the scheduling and resources of the project organization. This optimization tool combines operations research techniques (linear programming and branch and bound search) with artificial intelligence techniques (constraint propagation (Baptiste et al., 2001) and heuristic search (Cheng & Smith, 1994)). The tool optimizes the macro resource sizing and scheduling to eliminate the most serious backlogs for project participants while respecting project priorities. Murray (2002) also conducted a brief investigation of the application of genetic algorithms to engineering design project scheduling problems.

During the last few years, evolutionary computational methods have been used to optimize various kinds of systems in ways that rival or exceed human capabilities. For example, GP has produced optimization results for a wide variety of problems involving automated synthesis of controllers, circuits, antennas, genetic networks, and metabolic pathways (Koza et al., 2003). Prof. John H. Miller (2001) and his group at Carnegie Mellon University have done similar work, in terms of evolving organizations, but for simpler structures than our proposed research. In their research, they show that simple adaptive mechanisms allow for the creation of superior organizational structures. In addition, they conclude that, while they do not have proofs of optimal structures, the genetic algorithm was designed to solve difficult, nonlinear problems, and thus the structures that emerge from the algorithm should contain valuable hints about optimal form.

2.1 Virtual Design Team (VDT)

The Virtual Design Team (VDT) is a project organization modeling and simulation tool that integrates organizational and process views of strategic, time-critical projects. The vision behind VDT is to provide a method and tool 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 study of his project or projects and run simulations to predict project outcomes. Simulations also identify organizational 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 model components, a modeler can experiment with different solutions to determine which one meets his program quality, cost, and scheduling objectives.

Using VDT's a graphical interface, project managers design the organizational structure—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.

The user sets other organization attributes such as skill levels of each actor (individual or subteam) and decision making policies. The skill level of each actor can be set to low, medium, or high. The higher the skill level of individuals, the faster the task gets done, and the lower the rate of exceptions generated. 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 collocation" of the various specialists in an organization by setting the probability workers will attend to communications. The above decision-making policies can also be set to low, medium, or high.

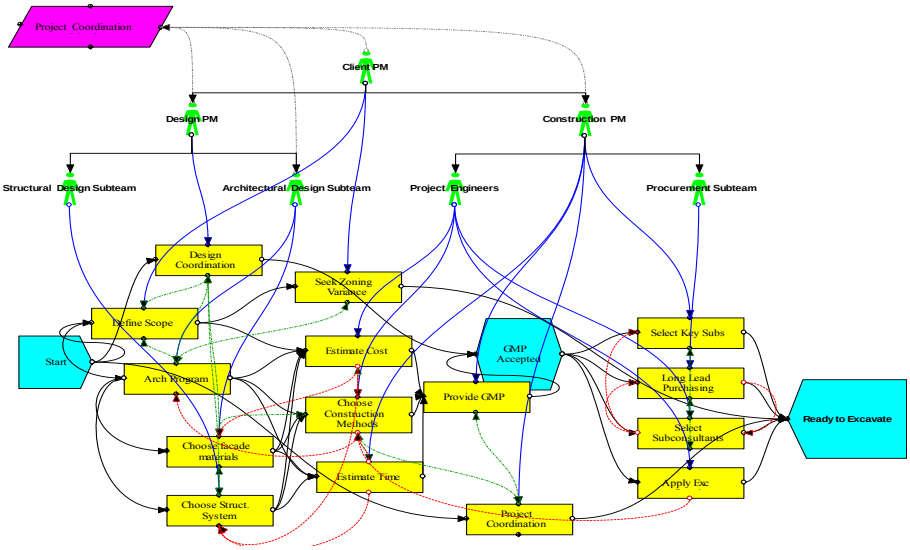

Fig. 1. User Interface of the VDT Simulator - Each project participant fills a position in the project organizational hierarchy and works on one or more activities. The organizational structure and the interdependence between activities define coordination requirements among individuals

Once the above attributes and topologies are set, the Monte Carlo discrete event simulation can be run (usually 50-100 trials 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.

Fig. 2. Sample 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

Once we implemented the first version of our postprocessor optimizer, we applied it to a case study that has been used for several years in a project management course taught at Stanford. The results produced by our GP were then compared against the best solution discovered by student groups and senior project manager groups over the last 6 years. In this case study, student and project manager groups are given a biotech plant project organization and asked to modify some of the individual/sub-team attributes and organizational policy structure in order to reduce the project schedule duration as much as possible, while maintaining acceptable levels of quality risk.

4 Methodology

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 The Representation

The remainder of this section explains the setup for the genetic programming tree and how it was 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, Assign, Aloc
Fitness Cases	15 total – 1 for simulation duration, 1 for FTE, 13 for each activities
Raw Fitness	SPD + TFTE * FTEW + $\sum$ (FRI_i * $FRIW_i$ + PRI_i * $PRIW_i$ + CR_i * CRW_i) (see section 4.4 Fitness Evaluation)
Standardized Fitness	Same as raw fitness
Parameters	Population size M = 3000 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

4.2 Function and Terminal Sets

Terminal set P1 through P7 represents actors in the group. CFM stands for Centralization, Formalization and Matrix Strength. Functions Up, Down, Same can have different meaning depending on the Terminals that they connect to and whether there are FTE, Assign or Aloc functions in between. For example, the function FTE increases or decreases the number of FTE for each actor depending on the number of Up/Down functions preceding it in the genetic Tree. The Assign function assigns an activity to an actor, and Aloc specifies percentage allocation for each activity.

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 transforming tree produced by this genetic operation. In this configuration, for example, the skill attributes of P3 (person 3) in the organization structure changes based on the type of its parent and grandparent nodes. So, in this case, P3's first skill level (e.g. Project Management) remains the same, her second skill level (e.g. Software Engineering) increases, and her third skill level (e.g. Design Coordination) decreases. In the sample tree below CFM's parents and grandparents are Same, Up, and Down. So, in this case, the genetic program tree suggests that centralization should remain the same, formalization should increase by one level, and matrix strength should decrease to optimize the overall project outcome.

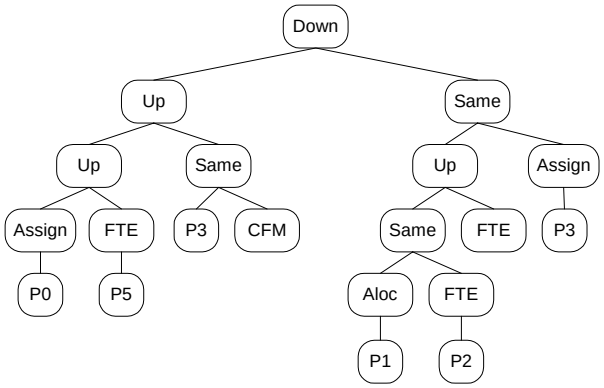

Fig. 3. Sample of a Transforming Genetic Tree. Program trees created by genetic operations modify the structure and attributes of a project organization. The genetic tree above transforms an organization design (not shown here) proposed by a project manager to a near optimal one

4.3 Genetic Tree Constraints

Since FTE, Assign , and Aloc functions can only appear next to the bottom of the genetic tree (i.e., Terminal sets should be the only children of them), constraints were added to the ECJ parameter files to enforce such limitations.

4.4 Fitness Evaluation

In each successive generation, evaluation of the fitness function is calculated on three primary inputs. First, the total simulation duration, or number of days to complete the project. Second, the quality risk values including communication risk, functional risk, and project risk. Third, the total FTEs, since there was a constraint for the maximum number allocated to the project. The formula 1 below shows how weighting factors are applied to these inputs to calculate the total fitness value. Note that the weighting factors are designed such that the fitness function heavily penalizes an increase in quality risk or FTEs.

$$SPD + TFTE * FTEW + \sum_{i=1}^{M} (FRI_i * FRIW_i + PRI_i * PRIW_i + CR_i * CRW_i) \quad \textbf{(1)}$$

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

4.5 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.

5 Results

We divided our case study experiment into two phases. In Phase I, we defined a simplified GP. In this process, we varied only the levels of the actors' skills. Then, we compared the results found by the GP with the known optimal solution. In Phase II, we kept skill levels constant, and varied the number of Full Time Equivalent FTEs (i.e., human resources) added to different positions. We also varied organizational policy attributes such as the levels of centralization, formalization and matrix strength and the assignment of activities to actors using GP. We then compared the GP results against the best solution found by previous student and manager groups. In the next two sections, we discuss the findings of this experiment.

5.1 Varying Actors' Skill Levels

There are seven positions (actors) in this project organization, and each one of these positions has two to eight different skills. The skills range from biotechnology to design coordination to mechanical/electrical, etc. There are a total of 29 skills for all seven positions. Each one of these skills can be set to three levels of low, medium, and high. Therefore, the total number of combinations that one could try to find an optimal solution exhaustively is $3^{29} = 6.8 * 10^{13}$. Thus, the sample space is vast and an exhaustive search is infeasible.

It should be obvious that the more skilled the actors, the faster the tasks get done, and the fewer the exceptions (i.e., when an actor requires additional information or a decision to complete part of a task, or the actor generates an error that may need correcting.). In this case where we are not concerned about cost, the optimal solution would be when the skill levels of all actors are set all to high. Knowing the above fact, in one scenario we set skill levels of all actors to "high", ran the VDT simulation and compared the results with the base results where we had the skill levels of all actors set to "medium". At the base level, we found that the simulated schedule end was March 28, 2001, and when we set all skill levels to "high", the project duration was reduced by 69 days and the simulation showed that the project schedule end would be Jan 17, 2001. Then, we ran the simulation again using the suggested solution by GP and we found identical results as when all skill levels were set at high.

5.2 Varying Actors' FTEs and Organization's Policy

In Phase II, we allowed the GP to vary the assignment of activities to actors, percentage allocation for each activity, the Full Time Equivalent's (FTE) of each actor in 0.5 FTE increments, and organizational policy properties such as levels of centralization, formalization and matrix strength using GP.

The best individual found by GP in generation 21, and it is shown below in a lisp-type format:

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

Then, we compared the results with the best results obtained by more than 40 teams of students and managers over the past six years.

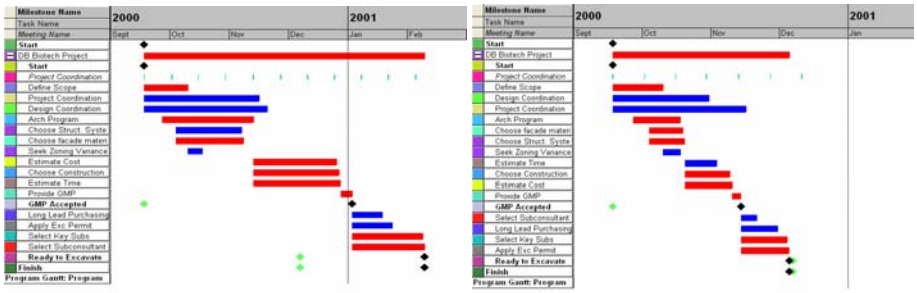

Fig. 4. Comparison of Gant Charts before *(Left)* and after *(Right)* Evolutionary Process. GP reduced end date from Feb 20, 2001 to Dec 5, 2000. This GP solution is better than the best solution (Dec 7) found by >40 student and manager teams for this problem over the last 6 years

The best individual found by GP in generation 21 beats the best human-discovered solution by 2 days. The best human solution reduced project completion from Feb 20, 2001 to Dec 7, 2000; the GP-suggested solution reduced the project end date to Dec 5, 2000. This is shown in Figure 4 above. In addition the quality risks such as communication risks were improved as shown in figure 5 below.

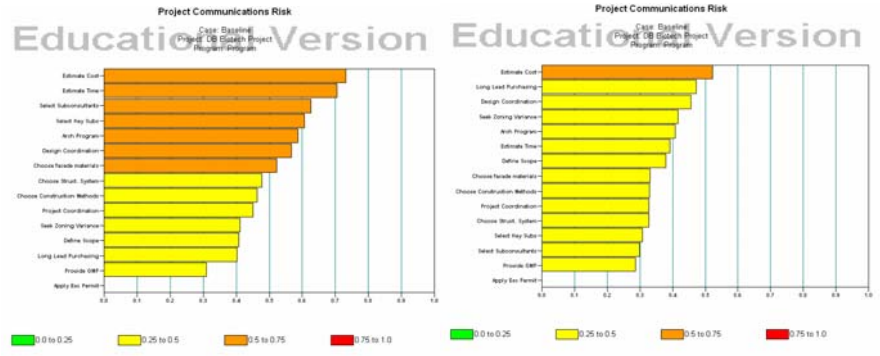

Fig. 5. Comparison of Quality Risks Before *(Left)* and After *(Right)* Evolutionary Process. Originally 7 out of 14 activities had quality risks higher than acceptable 0.5 thresholds (orange bars). With the suggested organizational changes, quality risks for all activities improved

6 Discussion of Results

We compared the solution produced by our genetic programming methodology both against the theoretical optimal solution and the best human generated solution. In both cases, the results were promising.

In the first case, as mentioned above, the outcome results found by GP were identical with the optimal case. However, interestingly, the suggested solution found by

GP was not identical to the optimal solution. (i.e., there were multiple solutions that yielded identical optimal outcome.) Unlike the optimal case scenario, GP did not have to set all skill levels to "high". In fact, there were situations where the levels of some actors' skills were reduced from "medium" to "low", and still the outcome matched the optimal solution. For example, the "General" skill of the "Structural Design Sub-team" was reduced to "low", and the "Mechanical" skill of the "Construction PM" was kept at "medium". This showed that GP could generate different solutions to a problem, so that a project manager can better decide which solutions to pick. For example, different solutions suggested by GP could advise that we could reduce skill levels of certain actors and increase skill level of others. So the project manager has the choice between different alternatives to pick a solution that better fits her/his specific project and available resources.

In the second case, where GP was allowed to vary the number of FTEs added to actors, the assignment of activities to actors, percentage allocation for each activity, and the policy attributes of the organization project, the solution found by the evolutionary process surpass the best Student/Manager (S/M) solution ever found. The number of FTEs and where they were added in GP solution matched exactly the best S/M solution. The S/M solution had suggested for swapping two activities between two actors. GP solution suggested an additional reassignment of an activity and change in percentage allocation of a couple of activities, in addition to the swaps suggested by the S/M solution. Also the S/M solution had suggested increasing both formalization and matrix strength by one level, whereas the GP solution suggested adding a level only to matrix strength.

As shown in the results section, GP has been able to improve the final outcome in both cases and meet the given constraints. The greatest improvement is seen in the project schedule, where the simulated duration was reduced by 77 days, followed by substantial improvement in communication, functional and project quality risks.

We also considered looking at the trend of improvements through different generations. Figure 6 below shows that the greatest improvement of the best individual fitness value was made between generations 1 and 6 (solid pink line). This figure also displays the mean fitness improvements during different generations (yellow dotted line). As shown, although the fitness value of best individual is not improved from generation 21, the mean fitness value has improved. This means that GP has produced more individuals within a generation that match or are near the solution found by the best individual. This can be an opportunity for a project manager to select a solution that better matches her/his specific project needs.

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

Although the transforming genetic tree in its current form was shown to produce improved results, it might not be the most efficient. The tight dependencies of function set FTE, Assign, and Aloc on their 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 for the skill levels and FTEs. Thus, after many generations, the right side of the genetic tree was more active than the left side.

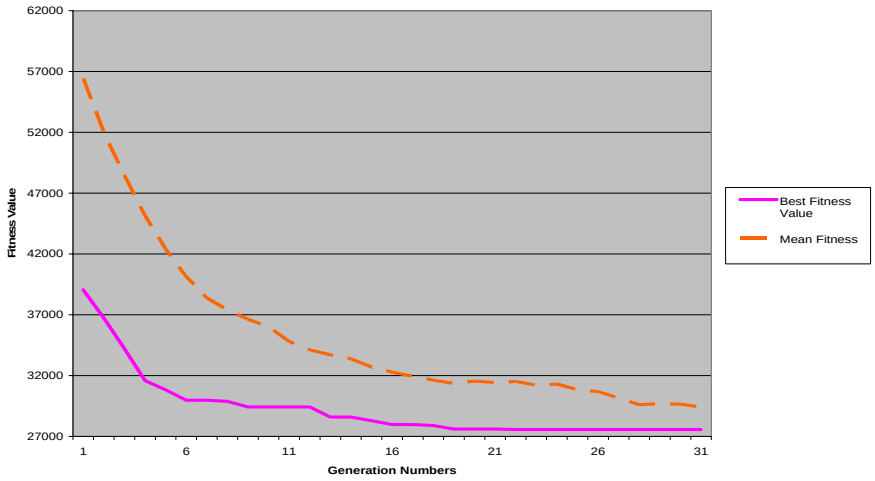

Fig. 6. Improvement of Fitness Value Generations 1 Through 30 – Although the best fitness value (solid line) does not improve after generation 21 (best solution found), the mean fitness value is improving. This means there are more alternative solutions that a manager can pick among the best solutions

7 Conclusions

This research has made successful first steps towards the optimization of project organization designs. Instead of redesigning the project organization from scratch, a human generated design was used as a baseline. Several input attributes such as decision making policies, individual / sub-team properties, activity assignments, and actors' attention allocation—were adjusted using genetic programming to evolve the project organization design against a fitness function representing the goals for the project. The effects of the evolutionary process on simulated project duration and quality risks were noticeable. We compared the results produced by the GP with those generated by humans. Our GP post processor for VDT beats the best human trial-and-error performance of > 40 teams for this realistic problem.

8 Future Research

This work is only the beginning use of genetic programming in optimizing organizational designs. 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. The topology, communication, and other individual and team properties—such as who reports

to whom, team experience, and application experience—are some of the other parameters that could be added to this model. Eventually, the evolutionary post-processor should be integrated within the VDT model.

9 Acknowledgments

We would like to thank the developer of ECJ (A Java-based Evolutionary Computation and Genetic Programming Research System) Sean Luke, for providing the easy-to-use programming environment that was used for this project. We would also like to thank Mark Ramsey, for helping us to connect the genetic tree to the VDT model.

References

1. Baptiste P., Le Pape, C., Nuijten, W.: Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems, Kluwer Academic Publishers, Boston, MA. (2001)
2. Burton, R., Obel, B.: Strategic Organizational Diagnosis and Design: Developing Theory for Application, second edition, Academic Publishers, Boston, MA (2004)
3. Cheng-Chung C., Stephen F. S.: Generating feasible schedules under complex metric constraints. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), volume 2, pages 1086--1091, Seattle, Washington, USA, AAAI Press/MIT Press (August 1994)
4. Galbraith J.R.: Organizational Design, Reading, MA: Addison-Wesley (1977)
5. Hewlett W.R.: Design and Experimental Results of a Post-Processor of VitéProject, B.S. Honors Thesis, Symbolic Systems Program, Stanford University (2000)
6. Jin Y., Levitt RE. The Virtual Design Team: A computational Model of Project Organizations. Computational and Mathematical Organization Theory; 2(3):171-196 (1996)
7. Koza J.R., Keane M.A, Streeter M.J., Mydlowec W., Yu J., Lanza G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence, Kluwer Academic Publishers (2003)
8. Koza, John R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: The MIT Press (1992)
9. Kujacic, M., Bojovic N.: Organizational Design of Post Corporation Structure Using Fuzzy Multicriteria Decision Making, Computational & Mathematical Organization Theory, 9, 5-18, Kluwer Academic Publishers, Manufactured in the Netherlands (2003)
10. March, J.G., Simon H.A.: Organizations. New York: John Wiley and Sons (1958)
11. Miller J. H.: Evolving Information Processing Organizations – In Dynamics of Organizations Computational Modeling and Organization Theories: edited by Alessandro Lomi and Erick R. Larsen, MIT Press / AAAI Press, Ch-10 pp 307-327 (2001)
12. Murray, M.: On the Application of Genetic Algorithms to Scheduling Engineering Design Projects. In Koza, John R. (editor). Genetic Algorithms and Genetic Programming at Stanford 2002. Stanford, CA: Stanford University Bookstore (2002)
13. Tatum C.B.: Decision Making in Structuring Construction Project Organizations. Ph.D. Dissertation Thesis (1983)