AN EQUITABLE APPROACH TO THE PAYMENT SCHEDULING PROBLEM IN PROJECT MANAGEMENT

Gündüz Ulusoy

Manufacturing Systems Engineering
Faculty of Engineering and Natural Sciences
Sabancı University
Orhanlı,Tuzla, 81474 Istanbul, Turkey
Tel.: 90 216 4839503 Fax: 90 216 4839550 e-mail: gunduz@sabanciuniv.edu

Serkan Cebelli

Koç Holding Co., Automotive Group
M. Üstündağ Cad. No. 13, Kadıköy
81020 Istanbul, Turkey

Published in the European Journal of Operational Research, 127, pp.262-278, 2000.

AN EQUITABLE APPROACH TO THE PAYMENT SCHEDULING PROBLEM IN PROJECT MANAGEMENT

ABSTRACT

This study reports on a new approach to the payment scheduling problem. In this approach, the amount and timing of the payments made by the client and received by the contractor are determined so as to achieve an equitable solution. An equitable solution is defined as one where both the contractor and the client deviate from their respective ideal solutions by an equal percentage. The ideal solutions for the contractor and the client result from having a lump sum payment at the start and end of the project respectively. A double loop genetic algorithm is proposed to solve for an equitable solution. The outer loop represents the client and the inner loop the contractor. The inner loop corresponds to a multi-mode resource constrained project scheduling problem with the objective of maximizing the contractor's net present value for a given payment distribution. When searching for an equitable solution, information flows between the outer and inner loops regarding the payment distribution over the event nodes and the timing of these payments. An example problem is solved and analyzed. A set of 93 problems from the literature are solved and some computational results are reported.

1. THE PROBLEM DEFINITION

A review of the project management and scheduling literature reveals that a considerable amount of effort has been spent on the resource constrained project scheduling problem (RCPSP) with the objective of minimizing the makespan (Özdamar and Ulusoy, 1995; Kolisch and Padman, 1998). It is only relatively recently that financial considerations attract increasing attention although the first attempts for modelling financial aspects of project management appeared rather early (Russell, 1970). Financial aspects of project management have been included in the problem formulation through the use of the maximization of the *net present value* (NPV) objective which includes both the negative (disbursement) and the positive (receipt) cash flows throughout the project. A recent review of project network models with discounted cash flows is given by Herroelen *et al*. (1997).

The problem treated in this paper can be considered as an extension of *the payment scheduling problem*. The payment scheduling problem considers the amount and timing of the

payments as decision variables which can affect the financial returns of both the contractor and the client. The amount and timing of payments are an important agenda item in the negotiations between the client and the contractor. Here, the payments are assumed to occur at the event nodes.

In this study, the payment scheduling problem is attacked by taking into account not only the contractor or only the client but both the client and the contractor. The motivation behind this study is to make a first attempt to the investigation of the negotiation process between the contractor and the client. Both the contractor and the client aim to maximize their financial returns and hence, their respective NPVs. The budget including a profit margin is agreed upon by the client and the contractor. The contractor meets the costs of the activities by using the payments made by the client towards the budget. Whenever the cumulative cash flow of the contractor becomes negative, then the contractor is assumed to borrow funds at a borrowing rate larger than the regular discount rate used to calculate the NPVs. The most preferred payment schedule for the contractor is obtaining the total payment as a lump sum at the beginning of the project. After receiving the total payment at the beginning, the contractor will try to minimize his/her costs by scheduling the activities in such a manner that activities with higher cash outflow will be scheduled as late as possible. To calculate the client's NPV, it is assumed that the budget is readily available initially and the NPV of all the disbursements by the client are subtracted from this lump sum. The remainder is the gain of the client resulting from not having to make the total payment initially. The most preferred payment schedule for the client is a lump sum payment made at the completion of the project. Then the client will not worry about the scheduling of the activities but only the project duration, since this is the factor affecting the NPV for the client directly. The most preferred payment structures for both the contractor and the client will be called the *ideal solutions* for both the contractor and the client respectively.

The investigation of the negotiation process is reduced here to the search for an *equitable solution* for both the contractor and the client. An equitable solution is defined as one where both the contractor and the client deviate from their respective ideal solutions by an equal percentage so as to reach a compromise solution to overcome the schedule

disagreements arising between them. The equitable solution is equitable in the sense that both parties agree to forego an equal percent from their ideal solutions.

The payment scheduling problem from the contractor's point of view was formulated by Dayanand and Padman (1993). They proposed a zero-one integer programming formulation and suggested and tested several heuristics. Dayanand and Padman (1994) presented a two-stage procedure in which the first stage consisted of a simulated annealing algorithm and in the second stage, activities are rescheduled to improve the project NPV. They reported that the performance of this approach was significantly better than the problem-dependent heuristics proposed earlier by Dayanand and Padman (1993). Another major result was that the improvement due to the second stage was indeed relatively small. Dayanand and Padman (1995) looked into the payment scheduling problem from the client's perspective. Several mixed integer linear programming models were introduced. The analysis showed that the client obtains the greatest benefit by scheduling the project for early completion. Making payments at regular time intervals was shown to typically increase the client's expenses. Dayanand and Padman (1997) introduced several deterministic models to analyze the payment scheduling problem with the objective of maximizing the contractor's NPV. In the models, a deadline is imposed and the number of payments is fixed. Once the total payment to be received from the client is determined, it remains unchanged during the progress of the project. They suggested that the models described can be used by both the contractor and the client subject to some modifications.

Bey *et al*. (1981) demonstrated how profitability in project activity can be increased through effective timing of cash outlays and receipts. In their model, the only resource constraint imposed was cash which was taken to be a doubly constrained resource. They utilized an example problem to illustrate how to minimize a project's cost or to maximize a project's expected value. With the use of a numerical example they showed that the optimal schedule varies with the objective function selected and depending whether it is viewed from the standpoint of the contractor or from the standpoint of the client. The cost of capital was shown to have an effect on the optimal time sequencing of activities.

2. THE PROBLEM FORMULATION AND A SOLUTION APPROACH

2.1. The problem formulation

The payment scheduling problem as defined above is represented over an activity - on - arc (*AoA*) network. All aspects of the problem are taken to be deterministic. The progress payments are assumed to be paid at the event nodes where an event occurs at the completion of one or more activities. The project is to be accomplished subject to a deadline. The resources employed are renewable resources each of limited amount. Each activity can be accomplished in general in more than one way of resource combinations and usage levels. Each different way of accomplishing an activity corresponds to a mode associated with this activity. Switching from one mode to another results either in a resource-duration trade-off or in a resource-resource trade-off. Once a resource is assigned to an activity by a certain amount, it is assumed that this amount of resource will be tied up by that activity for its whole duration. Each mode of an activity will have a different cost of accomplishment. It is assumed that the cash flow associated with the cost of an activity occurs at the completion of that activity. Note that this is not a restrictive assumption since any type of cash flow associated with an activity can be discounted to an equivalent amount occurring at the end of the activity using a proper discount rate. Activity scheduling is taken to be non-preemptive.

A payment distribution is defined here as the percentages of the budget to be paid by the client to the contractor at the event nodes prespecified. The event nodes where a payment might occur can be prespecified by the client and even by the contractor such as a payment at the initial node. Also, the number of event nodes where a payment might occur can be restricted to a prespecified number.

Given a payment distribution, then the maximum NPV of the contractor can be obtained by solving a deterministic multi-mode resource constrained max-NPV problem for the contractor. Using the timing of the progress payments, the NPV of the client can be easily calculated since the client is only affected by the amount and timing of the progress payments. Thus, by taking the absolute difference between the percent deviations from the ideal NPVs,

the objective function value Z for the given payment schedule can be calculated using (Eq. 1). The objective, of course, is the minimization of Z.

$$Z = \left| \frac{NPV_{cont}^{ideal} - NPV_{cont}}{NPV_{cont}^{ideal}} - \frac{NPV_{client}^{ideal} - NPV_{client}}{NPV_{client}^{ideal}} \right| \qquad \text{(Eq.1)}$$

where NPV_{client} and NPV_{cont} are the NPVs for the client and the contractor respectively. The superscript 'ideal' refers to the ideal solution in each case. Note that, $0 \leq Z < 1$.

2. 2. The negotiation process

During the negotiation process, information flows between the client and the contractor. The client suggests to the contractor a payment distribution over the event nodes. Having received this information, the contractor prepares a project schedule so as to maximize his/her NPV and sends back to the client the information on the timing of the payments. Using this information, the client can calculate his/her NPV. The interaction between the client and the contractor is depicted in Figure 1. The objective function value is calculated using Eq.1. Based on this result, the client suggests a new payment distribution. This process repeats itself until an equitable solution is reached. During the steps leading to the equitable solution, either NPV_{client} is in the decreasing mode and NPV_{cont} is in the increasing mode or vice versa, depending on the initial solution started from.

Figure 1 about here

---

3. A DOUBLE LOOP GENETIC ALGORITHM

Genetic algorithms (GAs) were developed by John Holland (1975) as artificial adaptive systems to simulate natural evolution. Because of their effectiveness and efficiency in searching complex search spaces, they are increasingly used to attack NP-hard problems. Several GA applications have been proposed for solving RCPSP with the objective of

minimizing the makespan. Özdamar (1999) proposed a GA with priority rules encoding for the multi-mode RCPSP with both renewable and nonrenewable resources. Lee and Kim (1996) reported their results of using simulated annealing, tabu search and genetic algorithm on RCPSP. Their GA was based on priority value encoding. Mori and Tseng (1997) attack a multi-mode RCPSP with renewable resources only. They use a direct chromosome representation in which each gene corresponds to an activity and includes the mode assignment, the scheduling order and start-finish times of the corresponding activity. The GA approach is compared to a stochastic scheduling method by Drexl and Gruenewald (1993) and is found to provide superior solutions. The GA developed by Hartmann (1997) considers both renewable and nonrenewable resources. The encoding is based on a precedence feasible set of activities and their mode assignments. A local search extension is employed to improve the solutions found by the basic GA. Extensive experiments are conducted with several different variants of the GA and results are compared with three other heuristics from literature. The proposed GA outperforms the other algorithms with regard to a lower average deviation from the optimal makespan. Hartmann (1998) introduced a permutation based GA for the RCPSP and compared it with both priority value based and priority rule based GAs to find it to be superior to both. To the best knowledge of the authors, no GA application has yet been reported for the multi-mode resource-constrained max-NPV problem.

To attack the problem, a double-loop genetic algorithm is designed. An example of a double loop genetic algorithm is provided by Gravel *et al*. (1998). In the approach proposed here, the double loop genetic algorithm structure reflects the payment scheduling part and the project scheduling part through the use of an outer loop and an inner loop. The outer loop captures the information regarding the percentage payment of the overall payment at each event node. The inner loop holds the information about the scheduling of the activities given the payment structure at the event nodes.

Figure 2 about here

The basic rules applied for the management of GA for both the inner loop and the outer loop are basically the same (Figure 2). An *initial population* is created by randomly

generating feasible solutions and it becomes the current population. The *population size* is kept constant throughout the GA. The solutions in the current population constitute a generation of solutions. A new generation is generated from the current generation through three operations: reproduction, crossover, and mutation. A number N_e of individuals called *elit individuals* are reproduced. For that purpose, the chromosomes in the population are listed in nonincreasing order of fitness values and the first N_e chromosomes from the top of the list are reproduced. N_e is determined by multiplying the *elitist ratio* with the population size. An individual from the current population is selected for mating randomly with a fitness-proportionate probability. The selection process from the current population is with replacement. A chromosome selected for mating is either subjected to *crossover operation* with a probability called the *crossover probability* P_c or is reproduced with a probability (1-P_c). If crossover is going to be applied, then a second individual is selected from the current population randomly with a fitness-proportionate probability. *Mutation operation* is applied to the chromosome resulting from this process with a probability called the *mutation probability* P_{mu}. The stopping criterion for the creation of new generations is the *number of generations*. The application of the procedure for a given number of generations is called a replication. The stopping criterion for the replications is the *number of replications*.

3.1. The Outer Loop

The objective function of the outer loop is as stated in Eq. (1). When calculating the objective function value of a chromosome, the number of significant digits is limited here to two. It is likely that for a given objective function value, there may exist several solutions corresponding to different NPV_{client} and NPV_{cont} pairs some of which are dominant solutions. Obviously, dominant solutions would be preferred by the decision makers. It is assumed that the NPV_{client} is computed by using the same discount rate *r* as in the case of NPV_{cont} .

3.1.1. The encoding

The chromosome representation of the outer loop captures information regarding the percentage of progress payments that will be paid at specific event nodes. Each gene holds an integer number representing an event node on the project network. The number of genes is

directly related to the size of the unit share of the budget allocated to a gene. For example, if the chromosome is composed of 20 genes, then the unit share of a gene is 5% of the budget. Each time an event node number appears on the chromosome, the corresponding event node receives a unit share of the budget. Hence, the number of times an event node number appears on a chromosome determines the progress payment to be received at that event node in question as a percentage of the budget.

A sample chromosome for a network with 6 event nodes and the unit share of the budget for each gene being 10% would be, for example, (3/5/1/6/5/6/1/6/3/5). Looking at the sample chromosome, it can be deduced that node 1 having a count of two on the chromosome, will receive 20% of the total payment; node 6 having a count of three will receive 30% of the total payment, and so on. If a finer distribution is desired, then the number of genes can be increased accordingly. This way, a more precise distribution of the budget over the events might be obtained, but at a cost of increased computational effort.

3.1.2. The crossover operator

A small scale experimentation including one-point, two-point and uniform crossover operators has revealed that the *two-point crossover operator* is the better one. Since the only information important is the number of occurrence of the event numbers, two-point crossover is applicable here without causing any infeasibility. To apply the crossover operation, two loci are chosen to divide the parent chromosomes into three parts each. The offspring child is formed by taking the first and last parts from the mother and the middle part from the father. An example of a crossover operation is given in Figure 3. The genes transferred to the child are shown in bold. The crossover operation results in a single offspring.

Figure 3 about here

3.1.3. The mutation operator

Two types of mutation operators have been considered. One is *the global mutation operator*. A chromosome is selected for mutation with the global mutation probability and each individual locus on that chromosome is considered for mutation using global mutation

probability and if chosen for mutation, the chosen locus is assigned an event number randomly. Here, on a selected chromosome more than one locus can be selected for mutation. The other is *the local mutation operator*. A chromosome is selected for mutation with the local mutation probability and a locus is selected for mutation and the chosen locus is assigned an event number randomly. Hence, only one gene is subjected to mutation on a selected chromosome. The choice between the two types of mutation operators is made as a result of parameter finetuning experiments.

3.1.4. The fitness value

To determine the fitness value of a chromosome X of the outer loop the objective function value for chromosome X is subtracted from the maximum objective function value for the population. In order to calculate the fitness value, one needs to know NPV_{client} and NPV_{cont} . Each chromosome X generated in the outer loop is sent to the inner loop so as to determine the maximum NPV_{cont} it can locate and the corresponding timing of the payments. This information is passed on to the outer loop to be used together with the payment structure for the calculation of NPV_{client}. So, the fitness value for chromosome X is obtained.

3.2. The Inner Loop

In the inner loop, the objective is to maximize NPV_{cont} under a specific payment distribution. The payment distribution is imposed by an individual of the outer loop whose fitness function value is to be computed. The outcome of the inner loop is an activity schedule found by maximizing the NPV_{cont}. NPV_{cont} is calculated using a discount rate *r*. During the time periods when the cumulative payments made by the client do not meet the cumulative expenses incurred by the contractor, the contractor borrows money at a borrowing rate larger than the discount rate *r*. Upon receiving the activity schedule, NPV_{client} can be calculated readily since the occurrence times of the nodes and hence the payments are ready.

3.2.1. The encoding

The chromosome representation captures both the information regarding the activities' ordering and their chosen operational mode. Thus, each gene holds the activity number and its corresponding mode.

It is obvious that any activity placed in a gene must not violate the precedence relationships. Moreover, the operators that cause a change in an individual's gene contents must preserve the feasibility. A sample individual with 5 activities is written as follows: (1-1, 3-2, 5-1, 2-1, 4-3). Here, the first digit in each entry represents the activity number and the second digit the mode associated with that activity.

3.2.2. The crossover operator

The crossover operator used in the inner loop is *the multi-component uniform order-based crossover* (MCUOX) developed by Sivrikaya-Şerifoğlu (1997). This crossover operator operates in the following manner (Figures 4 and 5).

Figures 4 and 5 about here

Starting from the first genes on the parent chromosomes, iteratively, one of the parents is chosen randomly and its next unconsidered activity becomes the next activity on the child. If the mode for the selected activity is different on the mother and the father, then the mode assigned to the activity on the child chromosome is randomly selected from the mode selection of the mother or the father. The crossover operation results in a single offspring. MCUOX has the advantage of preserving the precedence feasibility of the schedule.

3.2.3. The mutation operators

There are two types of mutation operators implemented in the inner loop. The first one is *the repositioning mutation* that is employed for the activities. Repositioning mutation operator chooses two positions on the chromosome randomly and transfers the contents of the former to the latter. Since the repositioning mutation can cause precedence violations, a simple

repair function complements repositioning mutation. Suppose that the gene at locus *pos1* is repositioned into another locus *pos2*. The algorithm implements a repair function for the two cases *pos1*<*pos2* and *pos1*>*pos2*. When *pos1*<pos2, the activity now in *pos2* may have jumped over some successor(s). If this is the case, the gene transferred to locus *pos2* is repositioned in the chromosome at the location right before the first successor between *pos1* and *pos2-1*. In the latter case, when *pos1*>*pos2*, the activity put on locus *pos2* may violate some precedence relationships. So, the activity is placed right after last predecessor between *pos2+1* and *pos1*.

Bit mutation, on the other hand, is used to change the mode selection for an activity. An activity, that means a locus on the chromosome, is chosen randomly and is assigned a random mode.

3.2.4. The scheduler

To obtain the NPV of an individual in the population, it must first be scheduled. The scheduler employed is a *forward* scheduler which schedules the activities as early as possible without violating the resource constraints and the activity ordering on the chromosome. Suppose there are two consecutive activities, *act1* and *act2*, on the chromosome in that order and suppose that they are both eligible, i.e. their predecessors have been scheduled. If the resource availability is not sufficient for *act1*, then *act2* will not be scheduled even if the resource availability is sufficient for scheduling it. Consequently, the scheduler will preserve the ordering of the activities on the chromosome, which means that any activity in a locus will not have a starting time smaller than an activity in a locus located prior on the chromosome. Furthermore, trying to improve the schedule by not translating the order of the activities on the chromosome to the starting times of the activities in the schedule would imply to interfere with the natural evolutionary process of the genetic algorithm. The schedule modified this way would correspond to a different chromosome which is expected to be generated by the algorithm itself.

A *backward* scheduler has also been developed for employing together with the forward scheduler. Backward scheduler would delay an activity whenever a delay is possible without violating the order of activities on the chromosome and/or a payment hinges upon the

completion of that activity. But backward scheduling has not resulted in a significant increase in the NPVs so as to justify the increase in the computation times resulting from its use. Due to this restriction, backward scheduler is applied only to the final chromosome for improving the final solution, if possible, by delaying certain activities without causing any increase in the project duration.

3.2.5. The fitness value

Once the chromosome is scheduled, the activities' finish times and the progress payments' occurrence times are at hand to compute NPV_{cont}. To compute the objective function value of the inner loop, a discount rate, a borrowing rate and the timing and amount of the progress payments are needed. In order to obtain the fitness value of a chromosome in a population, the minimum NPV value of the population is subtracted from that chromosome's NPV value.

3.3. Obtaining the Ideal Values for the Contractor and the Client

Recall that the most desired payment schedule for the contractor is a lump-sum payment made at the beginning of the project and for the client it is a lump-sum payment made at the completion of the project. Both ideal points are affected by the deadline imposed on the problem.

To capture the ideal solutions for the contractor and for the client, the genetic algorithm defined in Section 3.1 is employed. Here, the payment structure consists only of a lump-sum payment made at the beginning of the project and at the end of the project, respectively.

3.4. Parameter Finetuning

There are some parameters to be set to get more efficient results from the genetic algorithm. For finetuning the parameters of the double loop GA two meta-GAs are designed for the inner loop and the outer loop, respectively. The first attempt on meta-GA is provided by Grefenstette (1986). The idea with the meta-GA is to employ GA itself in the search for

good performing parameter values. The encodings of the chromosomes of the meta-GAs are based on the parameters to be set. For the inner loop, the chromosome representation is as follows: {*number of replications, number of generations, population size, crossover probability, bit mutation probability, repositioning mutation probability, elitist ratio*}. For the outer loop, the representation becomes the following: {*number of replications, number of generations, population size, crossover probability, mutation probability, elitist ratio*}. The range of the parameters employed for the meta-GAs are given in Table 1 and Table 2 for the inner loop and outer loop, respectively. The meta-GA for the inner loop is performed first and the parameters obtained are employed in the meta-GA for the outer loop which then follows.

The objective and the fitness functions are the same for the meta-GAs as the original GAs for which the parameters are being finetuned. The problem selected for solution is one with 53 activities, 30 nodes, and 3 renewable resources. The population size is 50 and the meta-GA continues for 60 generations. The initial population is generated randomly. The chromosome with the largest fitness value of the previous generation is transferred to the next generation. For generating new chromosomes, two-point crossover and bit mutation are applied consecutively with probability of crossover being 0.60 and probability of bit mutation being 0.125. The selection of parents for crossover is made randomly from the current population with replacement and with the probability of selection being proportional to the fitness of the chromosomes.

The parameter values recommended as a result of parameter finetuning are reported in Table 1 and Table 2 for use in the inner and outer loop, respectively. Both the global and the local mutation operators have been tested during the parameter finetuning experiments. The global mutation operator turned out to produce slightly better results and has thus been adopted for the computational studies following.

Table 1 about here

Table 2 about here

---

4. NUMERICAL STUDY

4.1. Example Problem

The double loop genetic algorithm is run on an example problem. The *AoA* representation of the example problem is given in Figure 6 with the activity numbers indicated on the arcs. The durations of the activities and the resource usages for both modes are given in Table 3. The resource limits for Resource 1 and Resource 2 are 5 and 2 units respectively. The budget is taken as the sum of the maximum cost of each of the activities in the project by using the most costly mode alternative and multiplying the sum by a profit margin. The discount rate and the borrowing rate are taken as 0.5% and 0.9% per period respectively.

Figure 6 about here

Table 3 about here

The deadline for the project is obtained using the following expression:

$$D = C_{min} + (C_{max} - C_{min}) * m_d \qquad \text{(Eq. 2)}$$

where C_{max} and C_{min} are the maximum and the minimum makespan values respectively, that the project can accomplish, and m_d is a multiplier, $0 \leq m_d \leq 1$.

For generating C_{max} and C_{min}, the genetic algorithm of the inner loop is used but this time with the objective of maximizing and minimizing the makespan, respectively. The maximum and the minimum makespan values of the sample project are determined to be 130 and 72, respectively. The scheduler employed avoids the maximum makespan to become infinite since the scheduler forces the activities to be scheduled at their earliest possible time their precedence and resource constraints permit.

4.1.1. Analysis of the results

The results obtained are given in Table 4. The deviational ratios of the client's and contractor's NPVs from their respective ideal solutions are roughly equal; 0.467 and 0.465,

respectively. The makespan of the project is 82. Backward scheduler is applied to the final solution resulting in a delay for the activities 5, 9, and 15.

Table 4 about here

The resource profile of the sample problem is given in Figure 7 with the activity numbers indicated on the figure. The cash flow profile is provided in Figure 8.

Figure 7 about here

Figure 8 about here

The change of the NPVs of the contractor and the client over the generations of the outer loop are provided in Figure 9. In this particular example, NPV_{client} starts at a low value and increases over the generations and stabilizes after generation 44. NPV_{cont} , on the other hand, starts at a high value and decreases over the generations to stabilize after generation 33. As can be seen from Figure 9, the sequence of solutions found over the generations, the decrease and the increase in the NPV_{client} and NPV_{cont} values are not necessarily monotonic.

Figure 9 about here

4.1.2. Some sensitivity experiments

To have an idea about the sensitivity of the double loop genetic algorithm and to test its responsiveness, a series of runs are taken by changing the data of the activities one at a time. Backward scheduler is not employed in any of the solutions reported in this section.

In the final solution reported in Figure 7, activity 12 appears to have been scheduled fairly early. Thus, as a first experiment, it is tested whether activity 12 will be delayed once its cost is increased. For that purpose, the cost of activity 12 is multiplied by 10. Indeed, the algorithm delays the start time of activity 12 from 29 to 60. A similar argument goes for activity 4. To check whether activity 4 will be delayed once its cost is increased, its cost is

multiplied by 10. With this new data, the algorithm delays the start time of activity 4 from 9 to 36. In both cases, with increasing activity cost the project makespan has increased to 90 and 86, respectively.

In the third experiment, the duration for the second mode of activity 7 is decreased with the resource usage remaining constant. In the fourth experiment, the resource usage for activity 8 is reduced while keeping the duration constant. The changes made are given in Table 5. Note that in both cases the changes are designed to make the use of the second mode for both activities more attractive. Indeed, the second mode has been selected in both cases without causing any considerable delays in project durations. Project durations become 84 and 82, respectively.

Table 5 about here

Another test is based on the sensitivity of the problem to the discount rate. In this test, the borrowing rate is taken as the discount rate multiplied by 1.8 for all of the discount rates tested. The results obtained are given in Table 6. As the discount rate r increases, the payments are observed to occur earlier and the costs later compared to the base case. Furthermore, the project duration increases considerably with increasing discount rate.

Table 6 about here

Another run is based on keeping the discount rate constant and evaluating the example problem with different borrowing rate multipliers. The results are given in Table 7. As the borrowing rate increases, in other words, the borrowing rate multiplier increases, a more evenly distribution of payments over the event nodes is observed so as to prevent the need for borrowing. The project duration doesn't seem to have been affected although the order of scheduling changes.

Table 7 about here

4.2. Computational Experience

To test the genetic algorithm further and gain computational experience, a set of 93 problems taken from the literature have been solved (Ulusoy and Özdamar, 1995). The distributions of the test problems according to the number of activities and the number of renewable resources are given in Table 8a and 8b respectively. The number of event nodes varies between 8 and 30. The number of modes per activity takes on values between 1 and 3 with the average number of modes per activity being 1.47.

Table 8a,b about here

The 93 problems are run for two levels of the deadline multiplier: m_d= 1.0 and m_d= 0.6, where a lower value of m_d corresponds to a tighter deadline. When m_d = 1.0, for 69 problems out of 93, $Z < 0.010$. Only for 2 problems, $Z \geq 0.200$; in other words, a satisfactory equitable solution could not be found for these two problems. For a large percentage of the problems, NPV_{client} and NPV_{cont} vary from their ideal values in the range 0.400 to 0.600. When m_d = 0.6, for 88 problems out of 93, Z = 0.0. For four of the remaining problems, Z ranges between 0.020 and 0.060. One problem has Z = 0.190. NPV_{client} and NPV_{cont} vary from their ideal values in the range 0.400 to 0.600. Thus, when m_d = 0.6, the results obtained are more satisfactory. The ideal contractor values do not decrease substantially for most of the problems where the ideal client values reduce considerably. Furthermore, a tighter deadline leaves less degrees of freedom for the temporal arrangement of the activities and hence also for the events. All these lead to improved objective function values, when m_d is taken as 0.6.

The objective function values Z converge rather early in the process. The number of generations for achieving the best value for Z in a problem is on the average 12. For a given best value for Z, a dominant solution might exist which has the same value for Z but has at least as high values for both NPV_{client} and NPV_{cont}. Most of the search is spent for finding such dominant solutions. In 13 problems out of 93, no dominant solution could be found for the best solution obtained. The number of dominant solutions obtained after reaching the first solution has been on the average 2.1. The dominant solutions have resulted on the average in an improvement of 2.2% for NPV_{cont} and 2.4% for NPV_{client}.

The computational times over a problem of typical size are as follows:

i. Inner loop takes on the average 0.45 sec for 1000 chromosomes in a problem with 3 renewable resources, 29 activities, and 13 nodes. The chromosome consists of 29 genes with two parameters each. These results are comparable to the ones reported by Hartmann (1998).

ii. Outer loop takes on the average 2542 sec for 1000 chromosomes in a problem with 3 renewable resources, 29 activities, and 13 nodes. The size of the chromosome is 20 genes.

These computation times have been obtained on a PC with 133 MHz clock-pulse and 32 MByte RAM using C^{++} compiler.

5. SUMMARY AND SUGGESTIONS FOR FURTHER STUDY

A double loop GA has been proposed to reach an equitable solution for the payment scheduling problem. The algorithm is demonstrated at the hand of an example problem and tested using 93 problems from the literature. The procedure is shown to reach equitable solutions in reasonable computation times.

A different cash flow structure than the payment scheduling problem results with the use of *progress payments*. Progress payments are made at the end of time intervals throughout the project to cover the cost incurred in that interval subjected usually to a retainage or a profit margin. Kazaz and Sepil (1996) considered the unconstrained max-NPV problem for the case of progress payments made at equal time intervals such as monthly payments, formulating it as an integer programming problem and making use of the activity profit curves. Sepil and Ortac (1997) extend the work to the resource-constrained max-NPV problem and suggest three different heuristic priority rules. The GA procedure presented here for the case of payment scheduling problem can also be applied to the case of progress payments. Candidate payment time nodes are located at equal time intervals throughout the project. The GA procedure then determines the payment time nodes and their corresponding amounts of payments.

The GA approach presented here can easily be extended to include resource-resource trade-off. The only extension needed is to define the modes.

The GA approach presented here can also take into account nonrenewable resources but in the test problems nonrenewable resources are not included. The inner loop has recently been the subject of another study where nonrenewable resources are included to the same set of 93 test problems employed here (Ulusoy *et al*., 1999). In that study, chromosomes violating the nonrenewable resource constraints are eliminated rather than repaired. Repair is avoided since such a repair would be expensive and would intervene with the evolutionary process of GA. The results have indicated that the addition of nonrenewable resources cause a 5.6% increase of the number of chromosomes generated.

Several extensions are due. One such extension would be imposing a bonus and a penalty structure around the deadline.

Computational gain can be achieved by not searching for dominant solutions. This can reduce the computational times considerably. It would also be interesting to look into the possibility of some computational gain as a result of a heuristic solution procedure replacing the GA in the inner loop. In both cases, the possible computational gain has to be weighed against the loss in the quality of the solutions.

ACKNOWLEDGEMENT

The authors would like to thank two anonymous referees for their constructive comments which improved the presentation of this paper.

REFERENCES

Bey, R.B., R.H. Doersch, J.H. Patterson, "The net present value criterion: its impact on project scheduling", *Project Management Quarterly*, **12**, pp. 223-233, 1981

Dayanand, N., R. Padman, "Payments in projects: A contractor's model", Working Paper 93-71, The Heinz School, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1993.

Dayanand, N., R. Padman, "A simulated annealing approach for scheduling payments in projects", Working Paper 94-40, The Heinz School, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1994.

Dayanand, N., R. Padman, "Project contracts and payment schedules: The client's problem", Working Paper 95-23, The Heinz School, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1995.

Dayanand, N., R. Padman, "On modelling payments in projects", *Journal of the Operational Research Society*, **48**, pp. 906-918, 1997.

Drexl, A., J. Gruenewald, "Non-preemptive multi-mode resource constrained project scheduling", *IIE Transactions*, **25**, 74-81, 1993.

Gravel, M., A.L. Nsakanda, W. Price, "Efficient solutions to the cell-formation problem with multiple routings via a double-loop genetic algorithm", *European Journal of Operational Research*, **109**, pp. 286-298, 1998.

Grefenstette, J.J., "Optimization of control parameters for genetic algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-16**, 122-128, 1986.

Hartmann, S., "Project scheduling with multiple modes: A genetic algorithm", Technical Report 435, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universitaet Kiel, Kiel, Germany, 1997.

Hartmann, S., "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics*, **45**, pp. 733-750, 1998.

Herroelen, W.S., P. Van Dommelen, E.L. Demeulemeester, "Project network models with discounted cash flows: A guided tour through recent developments", *European Journal of Operational Research*, **100**, pp. 97-121, 1997.

Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Michigan, 1975.

Kazaz, B., C. Sepil, "Project scheduling with discounted cash flows and progress payments", *Journal of the Operational Research Society*, **47**, pp.1262-1272, 1996.

Kolisch, R., R. Padman, "An integrated survey of project scheduling", Manuskript No. 463, Institut für Betriebswirtschaftslehre, Christian - Albrechts - Universitaet zu Kiel, 1998.

Lee, J.-K., Y.-D. Kim, "Search heuristics for resource constrained project scheduling", *Journal of the Operational Research Society*, **47**, pp. 678 - 689, 1996.

Mori, M. and C.C. Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem", *European Journal of Operational Research*, **100**, 134-141, 1997.

Özdamar, L., "A genetic algorithm approach to a general category project scheduling problem", *IEEE Trans. Systems, Man and Cybernetics*, Part C: Applications and Reviews, **29**, 1, pp. 44-59, 1999.

Özdamar, L., G. Ulusoy, "A survey on the resource constrained project scheduling problem", *IIE Transactions*, **27**, pp. 574 - 586, 1995.

Russell, A.H., "Cash flows in networks", *Management Science*, **16**, pp. 357-373, 1970.

Sepil, C., N. Ortac, "Performance of the heuristic procedures for constrained projects with progress payments", *Journal of the Operational Research Society*, **48**, pp. 1123-1130, 1997.

Sivrikaya-Şerifoğlu, F., *A New Uniform Order-based Crossover Operator for Genetic Algorithm Applications to Multi-Component Combinatorial Optimization Problems*, Unpublished Dissertation, Boğaziçi University, Istanbul, 1997.

Ulusoy, G., S. Şahin, F. Sivrikaya-Şerifoğlu, "A genetic algorithm approach for resource constrained project scheduling with discounted cash flows", Working Paper, Department of Industrial Engineering, Boğaziçi University, Istanbul, 1999.

Ulusoy, G., L. Özdamar, "A heuristic scheduling algorithm for improving the duration and net present value of a project", *International Journal of Operations & Production Management*, **15**, pp. 89-98, 1995.

TABLE 1. Inner loop parameter range and selected values

Parameter	Values	Value Selected
Number of replications	4, 5, 6, 7, 8, 9, 10, 11, 12	6
Number of generations	25, 30, 35,..., 95, 100	60
Population size	25, 30, 35,..., 95, 100	45
Crossover probability	0.2, 0.25, 0.3,..., 0.75, 0.8	0.65
Bit mutation probability	0.05, 0.1, 0.15, 0.2	0.15
Rep. mutation probability	0.05, 0.1, 0.15, 0.2	0.15
Elitist ratio	0.01, 0.02, 0.03,..., 0.14, 0.15	0.06

TABLE 2. Outer loop parameter range and selected values

Parameter	Values	Value Selected
Number of replications	2, 3, 4, 5, 6, 7, 8	2
Number of generations	25, 30, 35,..., 95, 100	60
Population size	25, 30, 35,..., 95, 100	65
Crossover probability	0.2, 0.25, 0.3,..., 0.75, 0.8	0.70
Global mutation prob.	0.05, 0.1, 0.15, 0.2, 0.25	0.25
Elitist ratio	0.01, 0.02, 0.03,..., 0.14, 0.15	0.07

TABLE 3. Data for the sample problem

Activity	**Mode 1**			**Mode 2**		
	Duration	*Resource 1 Usage*	*Resource 2 Usage*	*Duration*	*Resource 1 Usage*	*Resource 2 Usage*
1	0	0	0			
2	3	5	2			
3	5	3	1			
4	5	3	0			
5	5	3	0			
6	1	3	1	2	1	1
7	6	5	2	10	2	1
8	6	5	2	10	4	2
9	5	3	1	7	1	1
10	4	5	2			
11	9	3	0	10	2	0
12	7	5	2	10	2	1
13	8	2	2			
14	3	3	0	6	1	0
15	3	3	0	6	1	0
16	5	3	2			
17	11	1	0			
18	5	3	2	7	2	1
19	4	0	1			
20	3	1	1			
21	15	0	1			
22	0	0	0			

TABLE 4. Results obtained for the sample problem

NPV_{cont} = 763.014		**NPV_{client} =** 570.55
Deviation = 0.464		**Deviation** = 0.467
Event node	**Percent payment**	**Occurence time**
1	5 %	0
2	5 %	3
3	0 %	9
4	0 %	29
5	10 %	35
6	15 %	50
7	15 %	56
8	50 %	82

TABLE 5. Change of resource profile for activity 7 and 8

Activity 7 Mode 2			**Activity 8 Mode 2**		
Duration	*Resource 1 Usage*	*Resource 2 Usage*	*Duration*	*Resource 1 Usage*	*Resource 2 Usage*
7	2	1	10	1	1

Table 6. Discount rate analysis for the sample problem

Discount Rate	**NPV_{client}**	**Deviation of NPV_{client} from its ideal value**	**NPV_{cont}**	**Deviation of NPV_{cont} from its ideal value**	**Makespan**
0.005	570.551	0.467	761.213	0.465	82
0.01	764.219	0.531	752.386	0.536	89
0.015	861.09	0.552	781.724	0.554	90
0.02	943.774	0.545	839.063	0.548	104
0.025	971.512	0.549	886.197	0.540	99
0.03	1014.3	0.538	895.598	0.547	102

Table 7. Borrowing rate analysis for the sample problem

Borrowing Rate Multiplier	**NPV_{client}**	**Deviation of NPV_{client} from its ideal value**	**NPV_{cont}**	**Deviation of NPV_{cont} from its ideal value**	**Makespan**
1.2	575.142	0.463	759.734	0.466	86
1.4	578.685	0.460	765.112	0.462	89
1.6	574.591	0.463	754.297	0.470	87
1.8	570.551	0.467	761.213	0.465	82
2.0	567.168	0.470	766.433	0.462	85

Table 8a. Distribution of test problems according to the number of activities

Number of activities	Number of problems
15-19	20
20-24	51
25-29	16
30-39	4
40-44	1
45-55	1

Table 8b. Distribution of test problems according to the number of renewable resources

Number of renewable resources	Number of problems
1	29
2	47
3	14
4	2
5	1

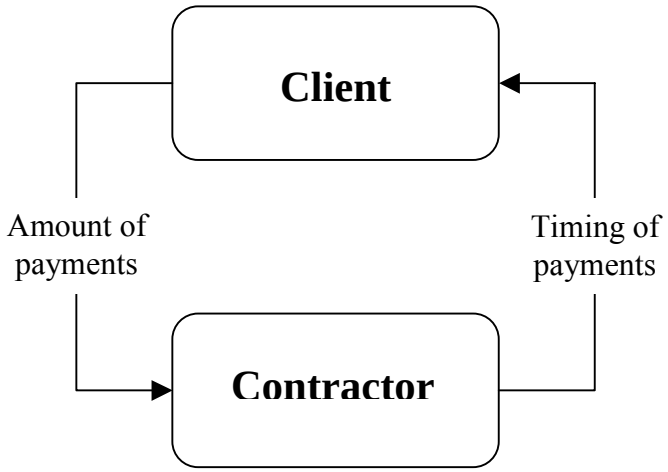

FIGURE 1. Interaction between the parties involved in the contract

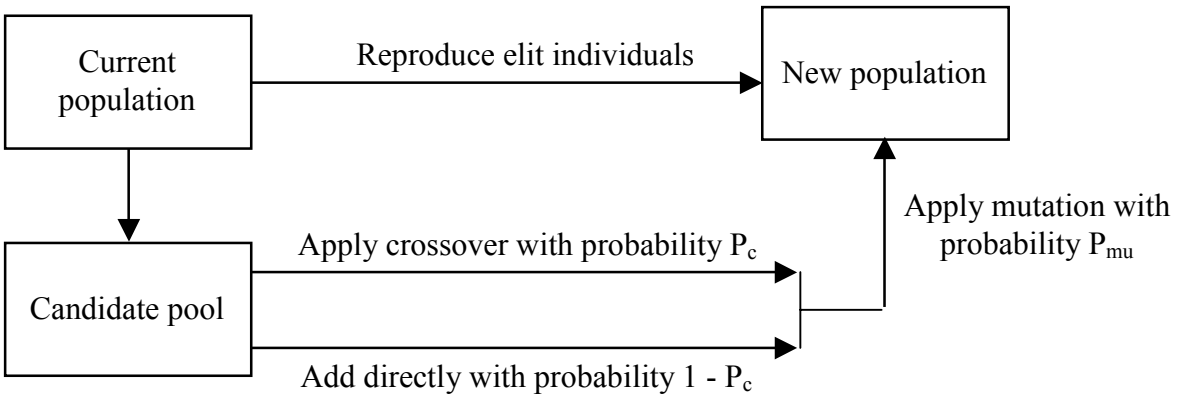

FIGURE 2. Management of GA

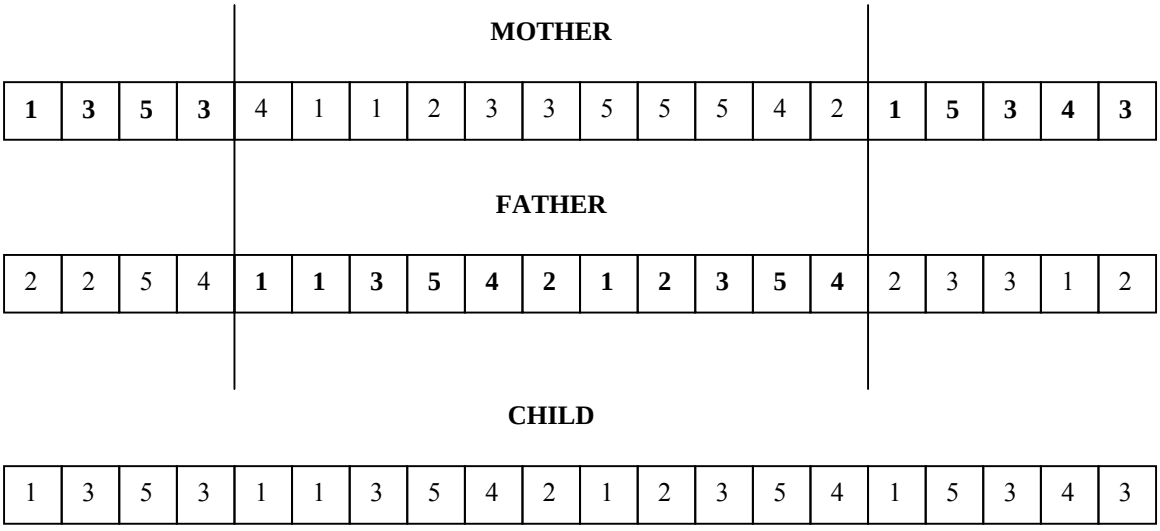

FIGURE 3. Crossover operator for the outer loop

START

- With the first position on the parents and the child.

LOOP

- Choose one of the parents randomly.
- Find the first activity on the chosen parent which is not assigned to the child yet. That activity becomes the next activity on the child.
- If the mode assignment made for that activity are the same on both parents, then make this same mode assignment on the child, too. If not, choose one of the mode assignments of the parents randomly and record it on the child.
- If all the genes of the child chromosome are completed, STOP;
- else proceed to the next gene on the child and repeat LOOP.

FIGURE 4. Algorithm of MCUOX

MOTHER **FATHER**

[1-1, 3-2, 2-1, 5-1, 4-3, 6-2] [1-1, 4-2, 2-2, 3-1, 6-2, 5-2]

Iteration	Result of parent selection		Child coding
	Activity	Mode	
0			[-, -, -, -, -, -]
1	Mother	No need	[1-1, -, -, -, -, -]
2	Father	Mother	[1-1, 4-3, -, -, -, -]
3	Mother	Father	[1-1, 4-3, 3-1, -, -, -]
4	Mother	Father	[1-1, 4-3, 3-1, 2-2, -, -]
5	Father	No need	[1-1, 4-3, 3-1, 2-2, 6-2, -]
6	Mother	Mother	[1-1, 4-3, 3-1, 2-2, 6-2, 5-1]

CHILD

[1-1, 4-3, 3-1, 2-2, 6-2, 5-1]

FIGURE 5. An example of MCUOX crossover operation

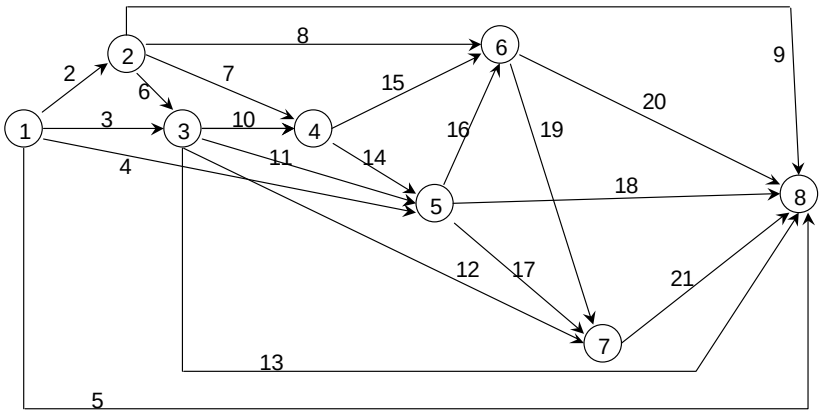

FIGURE 6. Activity-on-arc representation of the sample problem

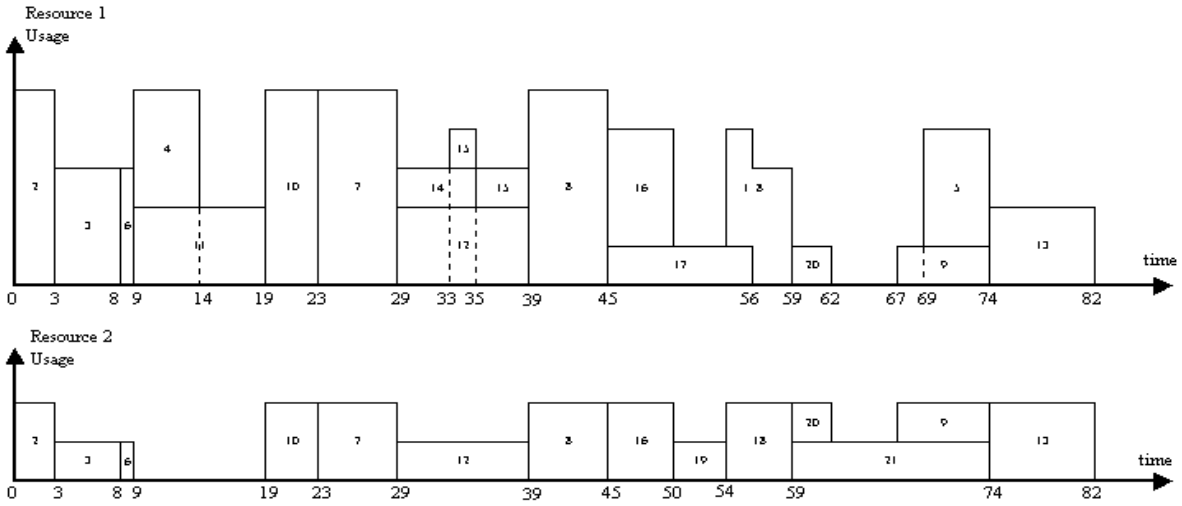

FIGURE 7. Resource profile of the sample problem

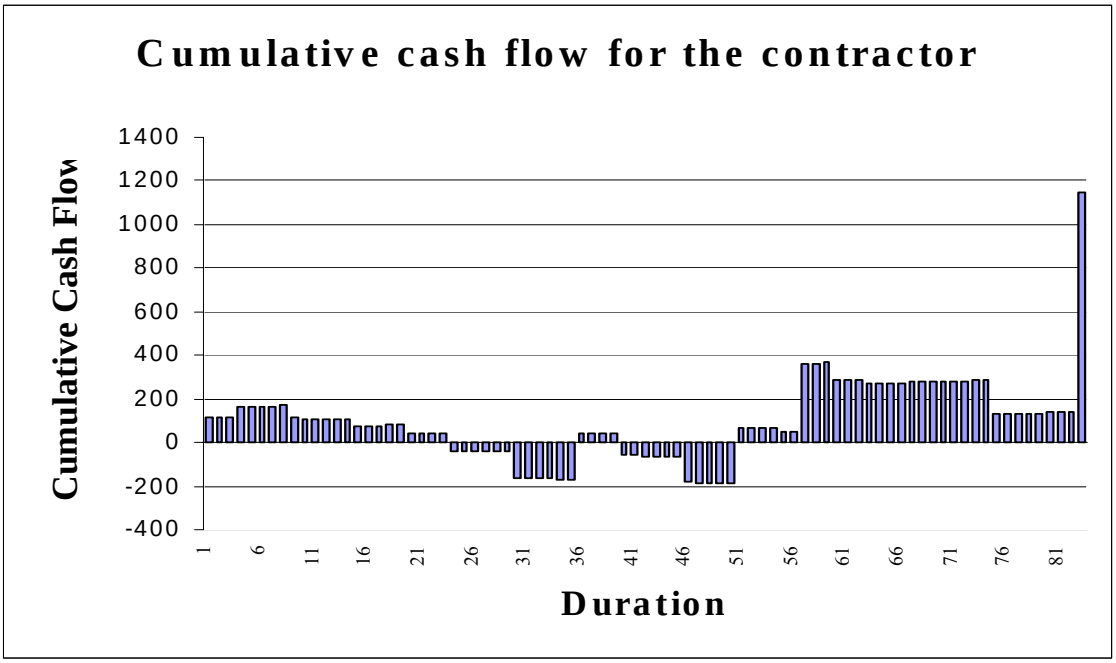

FIGURE 8. Cumulative cash flow for the contractor for the sample problem

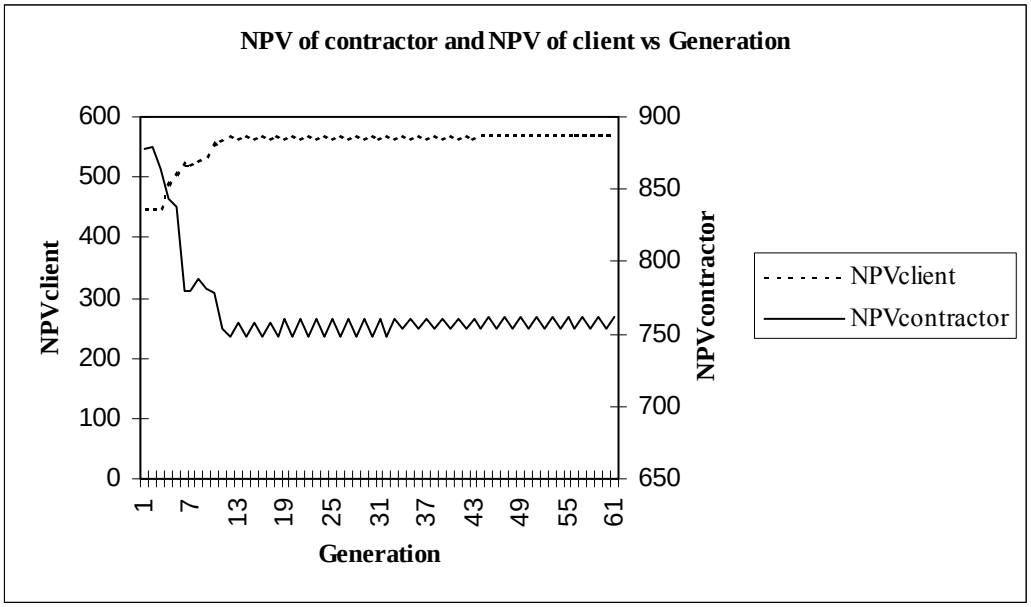

Figure 9. NPV_{cont} and NPV_{client} vs. generations