Constraint handling in Multi-Objective Evolutionary Algorithms- Constrained Multi-Objective Optimization Problem (CMOP)

A. Kumar , A. Ahmadi

INSA, Toulouse

1. **Abstract**

Multi-objective optimization (also known as multi-objective programming or Pareto optimization) is an area of multiple criteria decision making that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. In real life there are problems where we have to optimize the more than one function with some constraints that have to be satisfied to accept the solution. Many methods have been developed to solve these constrained multi objective optimization problems but there is no single method which works well for all the problems. In this study we will discuss different methods to solve these types of problems. All the experiments and plotting are done in PYTHON.

Keywords -: Constrained Optimization, Constraint Handling, Constraint Dominancy, Dynamic Penalty, APM, Self-Adaptive Penalty, Stochastic Ranking, Genetic Algorithm, AMOEA-MAP, NSGA-II

2. Introduction

Typically a constraint problem can be represented as follows:

$$\text{Minimize /maximize: } f_m(x), \quad m=1, 2, 3\ldots M$$

$$\text{Subject to: } g_j(x) \le 0 \quad j=1, 2, 3\ldots J$$

$$h_k(x) = 0 \quad k=1,2,3\ldots K$$

$$x_i(L) <= x_i <= x_i(U) \quad i =1,2,3.....n$$

Constraints divide the search space in two divisions: one is feasible and another one is infeasible search space. Constraints can be categorized based on many qualities-

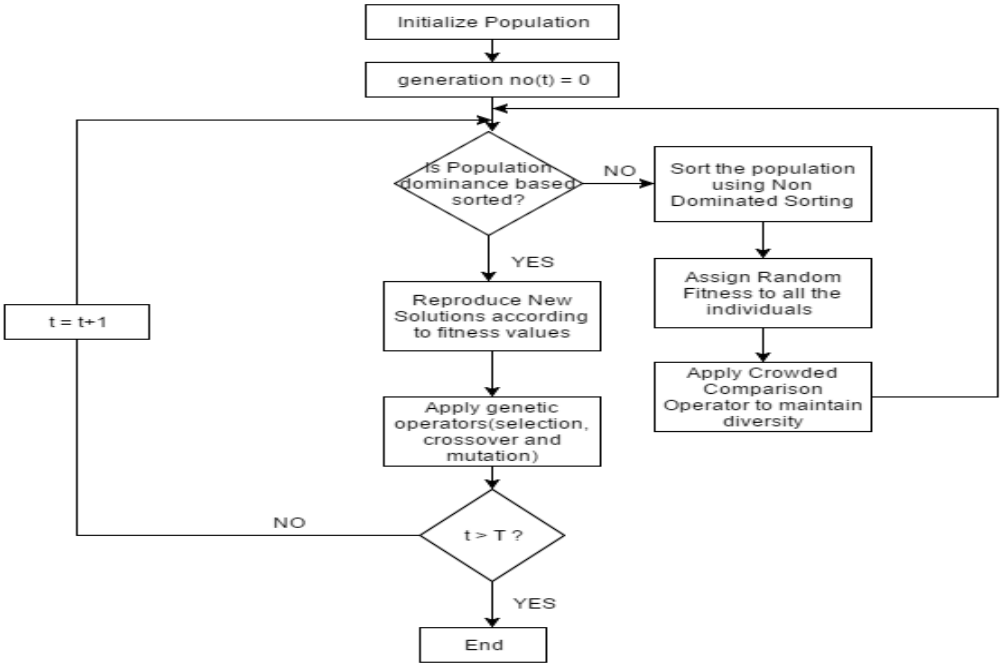
1. Equality and Inequality Constraint- Constraint depicted by h(x) in the formulation are equality constraint and those depicted by g(x) are inequality constraint.

2. Hard and Soft Constraint- A hard constraint must be satisfied by a solution of the problem and a soft constraint is that which can be relaxed to some extent to accept any solution.

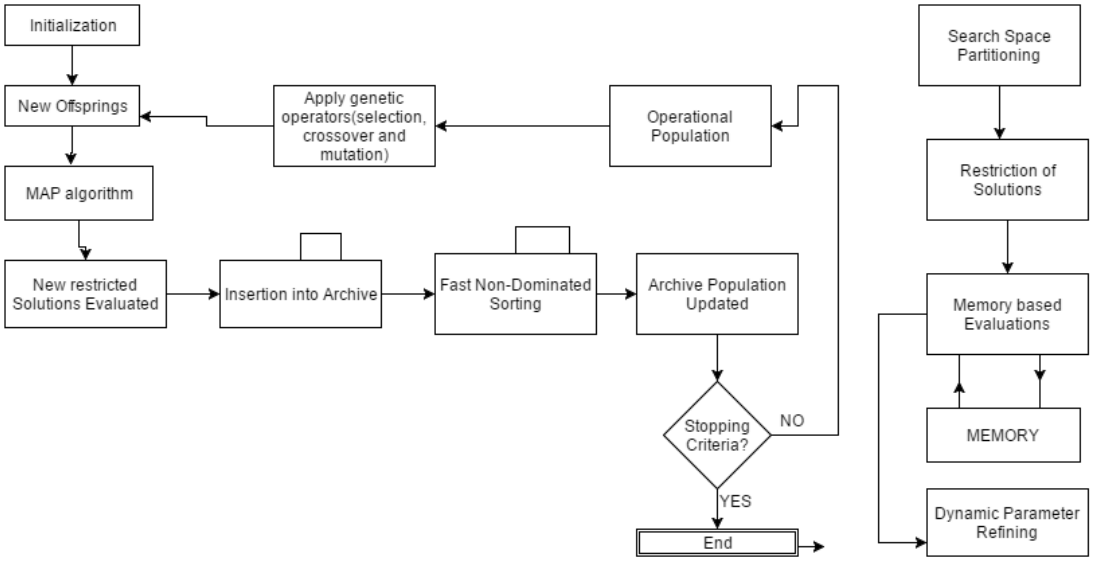
First two approaches that were used to get all the results are described:

1. Non Dominated Sorting Genetic Algorithms (NSGA-II) [12]

In a genetic algorithm, population of solution is encoded as binary string of 0s and 1s and this population is evolved towards a better solutions. To start the optimization process, a randomly generated population is used. Usually the initial population is spread in entire search space. Only those are seeded which can lead to a more good solution. The basic flow chart of working of NSGA-II is as follows:



2. **AMOEA-MAP** [5][6]



Archive Based Multi-objective evolutionary algorithm(AMOEA) **Memory Based Adaptive Partitioning(MAP)**

Memory based adaptive partitioning is an algorithm that is primarily based on partitioning of the search space so as to reduce the effort for finding optimal solutions in infinite search space to a partitioned search space. A memory is used to store previously evaluated solutions and to avoid unnecessary evaluation of new solutions that are close to them.

Archive based Multi-objective evolutionary algorithm uses two populations. One is the large one called Archive population and second is smaller called operational population. AMOEA starts by randomly initializing operational population. Tournament Selection, Simulated Binary crossover (SBX) and Importance Based adaptive mutation operator (IAMO) are used for selection and mating. MAP algorithm is now applied on Offspring population and resulting solutions are added to archive population until its size is maximum.

Despite having so large no of methods to handle CMOP, there is still no best algorithm which can work in all cases. So here we are going to see the nice 5 algorithms which are promising to work better than then the rest and also their advantages and drawbacks. We will be discussing these methods to solve CMOP.

1. **Dynamic Penalty** – It is a penalty based method which applies penalty to infeasible solution. This penalty use to increase with every generation number.

2. **Adaptive Penalty** -It is a penalty based method which adapts the information about feasibility of the solutions from the current generation and according to that it either increases or decreases the penalty applied to the solutions.

3. **Self-Adaptive Penalty Formulation**-It is another penalty based method which gathers more information from the current generation about best and worst individual and feasibility of the individuals and according to all these information it change the penalty applied to the individuals.

4. **Constraint Dominancy** -This method takes care of constraint violation values of the individual and the objective function values and the solution with low amount of constraint violation is preferred i.e. constraint violation is dominated factor to choose the good individual.

5. **Stochastic Ranking-** This method introduces an approach to balance the dominance of objective and penalty function stochastically. It ranks the solution stochastically without implementing any penalty function.

In the next section these methods are presented with details.

---

3. **Constraint Handling Methods in Multi-Objective Optimization**

Every method is presented here in full details:-

1. Dynamic Penalty Method

The basic idea of a penalty based method is to transform a constrained problem into an unconstrained problem by adding a penalty term into the objective function. Dynamic penalty is a method which applies a penalty that increases with each generation.

Jones and Houck [1] gave following dynamic formulae to evaluate individual solution at each generation (t):

$$fitness(x) = f(\overline{x}) + \left(C\ t\right)^{\alpha}\ SVC\ (\beta, \overline{x})$$

where C, α, β are user defined parameters. Author used C=0.5, α=1 or 2, $\beta = 1\ or\ 2$.

Here SVC is defined as follows:

$$\text{SVC}= \sum_{i=1}^{J} {D_i}^{\beta}(\overline{x}) + \sum_{j=q+1}^{K} D_j(\overline{x})$$

where

$$D_i(\overline{x}) = \begin{cases} 0 & if \quad g_i(\overline{x}) \le 0,\ 1 \le i \le J \\ |g_i(\overline{x})| & otherwise \end{cases}$$

$$D_j(\overline{x}) = \begin{cases} 0 & if \quad -\epsilon \le h_j(\overline{x}) \le \epsilon,\ J+1 \le j \le K \\ |h_j(\overline{x})| & otherwise \end{cases}$$

In our study, in the place of generation number **t** we have chosen number of function evaluations because with every increasing generation, number of function evaluations will increase and so is the penalty applied.

2. Adaptive Penalty Method[2]

The adaptive penalty method (APM) does not require any user defined parameter and it gathers information from the population. Fitness function is proposed this way:

$$fitness(x) = \begin{cases} f(x) & if\ x\ is\ feasible \\ \overline{f(x)} + \sum_{j=1}^{m} k_j\ v_j(x) & otherwise \end{cases}$$

where

$$\overline{f(x)} = \begin{cases} f(x) & f(x) > \langle f(x) \rangle \\ \langle f(x) \rangle & otherwise \end{cases}$$

where $\langle f(x) \rangle$ is the average of the objective function value in the current population.

The penalty parameter is defined at each generation as follows:

$$k_j = \langle f(x) \rangle \ \frac{\langle v_j(x) \rangle}{\sum_{j=1}^{m} [v_j(x)]^2}$$

3. Self-Adaptive Penalty[3]

The self-adaptive method is also an adaptive penalty method which calculates the penalty in following steps:

1. This method defines a new term infeasibility value which is sum of normalized constrained violation values

$$t(X) = \frac{\sum_{j=1}^{m} \frac{c_j(X)}{c_{max.j}}}{m}$$

where $c_j(X)$ is the constraint violation for the individual X for constraint j and $c_{max.j}$ is the maximum constraint violation for constraint j in current generation.

2. This method requires to calculate these 3 boundary solution in each generation which are calculated in following ways-

2.1. Best Individual (X^v) – If there is at least one feasible solution, then the feasible solution having the lowest objective value is taken as best. If there is no feasible solution, then the infeasible solution having the lowest amount of infeasibility value is taken as the best individual (regardless of the objective function value).

2.2. Worst Individual ($X^{\wedge}$)- For selecting the worst individual, all infeasible individuals are compared against best individual. Two cases may arise for the comparison. First if one of more infeasible individuals have smaller objective function value than the best individual, then the infeasible individual with highest amount of infeasibility value and objective value lower than the objective value of best individual. Second case is if all of the infeasible individuals have an objective value greater than the objective value of the best individual, then the infeasible individual with highest amount of infeasibility value and objective value greater than the objective value of best individual.

2.3. Highest Objective value Individual ($X^{\hat{}}$) - The individual which have highest amount of Objective function value is taken as highest objective value individual.

3. First Stage penalty

The first stage penalty is only applied if there is any infeasible individual, that have a lower objective function value than the best individual. If this condition is true then the first stage penalty is applied in the following way:

$$t^{\sim}(X) = \frac{t(X) - t(X^{\vee})}{t(X^{\wedge}) - t(X^{\vee})}$$

and fitness function is calculated in following way:

$$fitness_1(X) = f(X) + t^{\sim}(X)\ (f(X^{\vee}) - f(X^{\wedge}))$$

and if no penalty is applied then the fitness function remains same as objective function

$$fitness_1(X) = f(X)$$

4. Second Stage Penalty

The second stage penalty is applied in following way:

$$fitness_2(X) = fitness_1(X) + \gamma\ |fitness_1|\ \left(\frac{e^{(2.0 * t^{\sim}(X))} - 1}{e^{2.0} - 1}\right)$$

where ϒ is defined in the following way:

$$\Upsilon = \begin{cases} \frac{f(X^{\hat{}}) - f(X^{\vee})}{f(X^{\vee})} & if\ f(X^{\wedge}) \leq f(X^{\vee}) \\ 0.0 & if\ f(X^{\wedge}) = f(X^{\hat{}}) \\ \frac{f(X^{\hat{}}) - f(X^{\wedge})}{f(X^{\wedge})} & if\ f(X^{\wedge}) > f(X^{\vee}) \end{cases}$$

4. Constraint Dominancy [7]

This method constrained dominancy does not imply any penalty to the individuals. It considers the objective and constraint function separately. In this method individuals are selected using the following rules in the dominance based tournament selection procedure:

- If two solutions are feasible then solution with lowest fitness value is chosen
- Between a feasible and infeasible solution , feasible solution is chosen
- If both of them are infeasible, solution with lowest sum of constraint violation is chosen.

5. Stochastic Ranking[4]

This method stochastic ranking is a state of the art method. In previously discussed methods there is no balance between dominance of objective function and constraint function. This method introduces a probability factor Pf for using only objective function value for ranking the solution in infeasible search space. It means given any two individuals if both of them are feasible then the probability of comparing them according to objective function value if 1 otherwise it is Pf. This method ranks the population according to this probability factor in the bubble sort manner.

Stochastic Sort ()

```
For i =1 to N do
    For j=1 to N-1 do
        Generate a random no u ∈ (0, 1)
        If ( ( φ(j)= φ(j+1)=0) or u<Pf) then
            If ( f(j) > f(j+1)) then
            Swap ( j , j+1)
            End
        Else
            If ( φ(j) > φ(j+1)) then
            Swap (j , j+1)
            End
        End
        If (there is no swap) then
        Break the for loop
End
```

This way after applying this sorting function, stochastically best solution will be in the start of the population.

4. **Results**

The results and general settings for experimental environment for each of the problems are presented here:

4.1 In the Context of NSGA-II

General Settings and Operators:

For each of the problem Simulated Binary Crossover with probability 1 , tournament selection and Polynomial mutation was used.

Mutation Probability = 1 / Number of variables

1. Beam Design

Population Size: 30, Maximum Function Calls: 5000

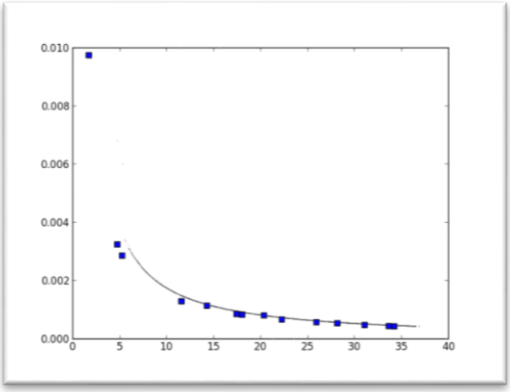


Figure 1.1(n) Dynamic Penalty

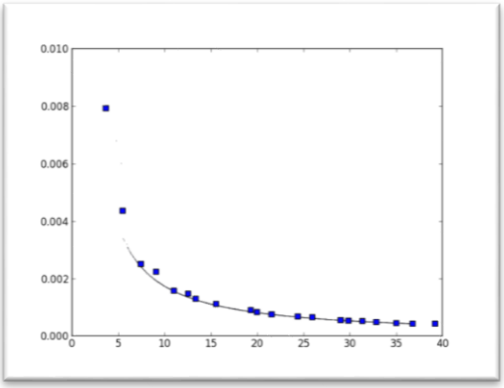


Figure 1.2(n) APM

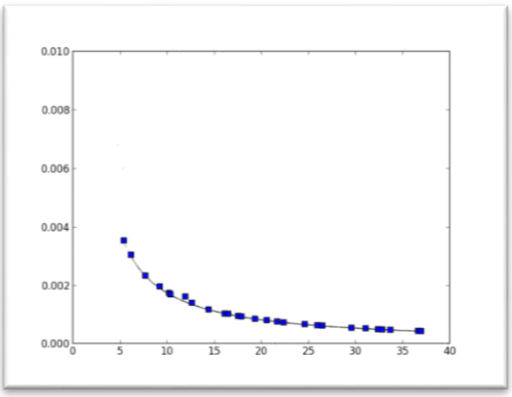


Figure 1.3(n) Constraint Dominancy

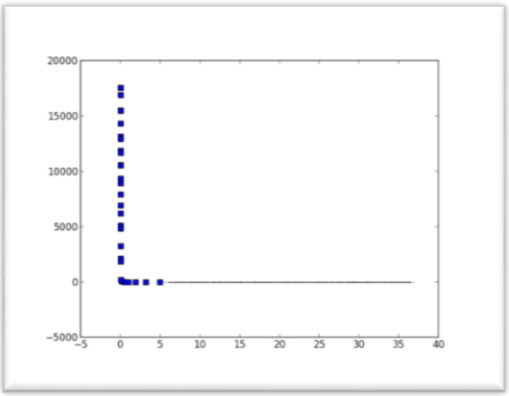


Figure 1.4(n) Self-Adaptive

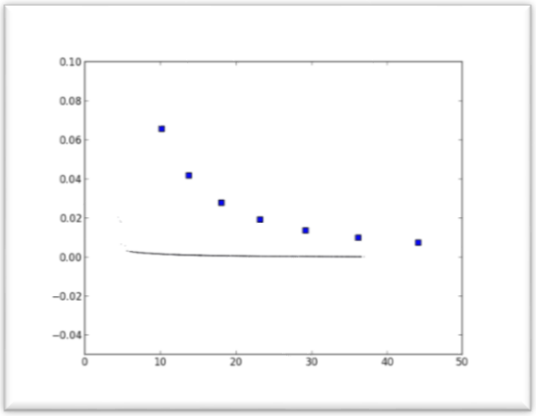


Figure 1.5(n) Stochastic Ranking

2. BICOP_2D

Population Size: 50, Maximum Function Calls: 10000

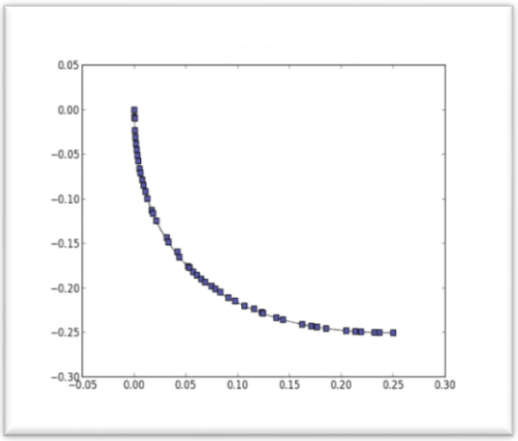


Figure 2.1(n) Dynamic Penalty

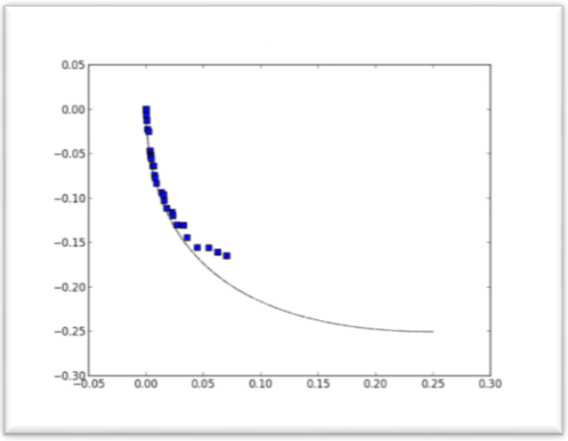


Figure 2.2(n) APM

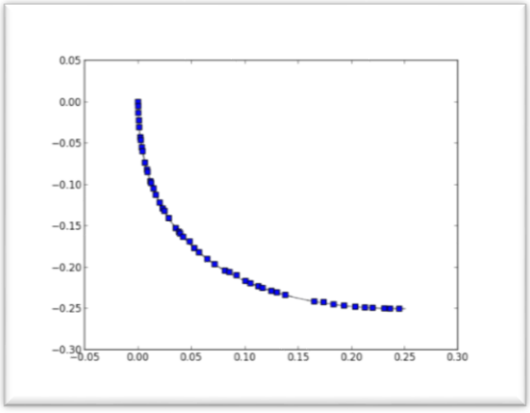


Figure 2.3(n) Constraint Dominancy

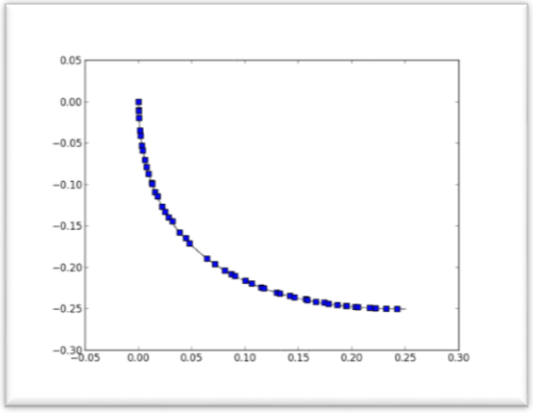


Figure 2.4(n) Self Adaptive

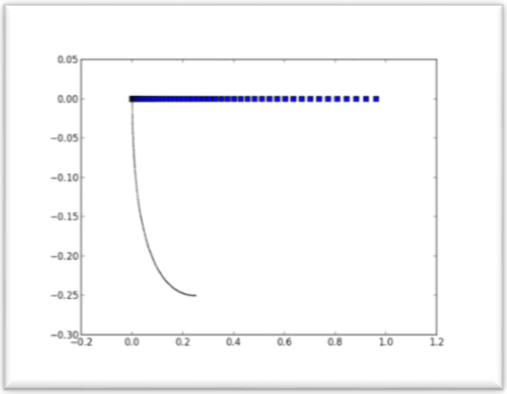


Figure 2.5(n) Stochastic Ranking

3. G7_2D

Population Size: 50, Maximum Function Calls: 10000

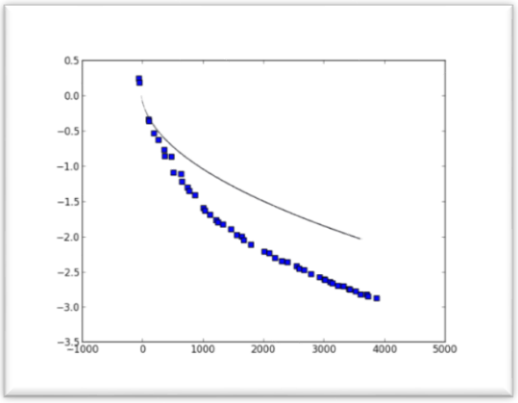


Figure 3.1(n) Dynamic Penalty

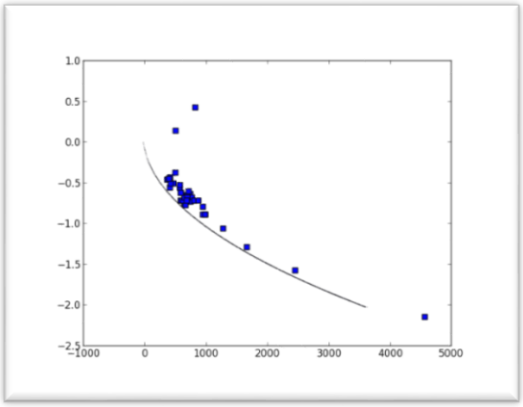


Figure 3.2(n) APM

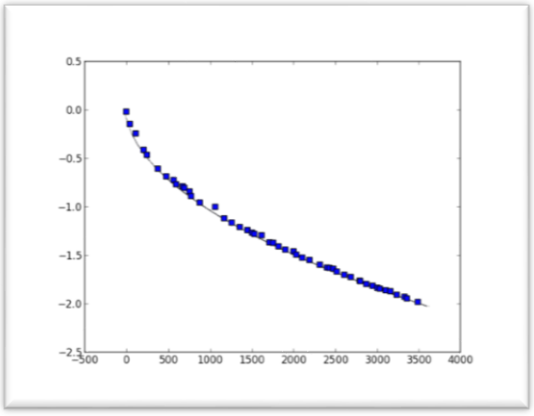


Figure 3.3(n) Constraint Dominancy

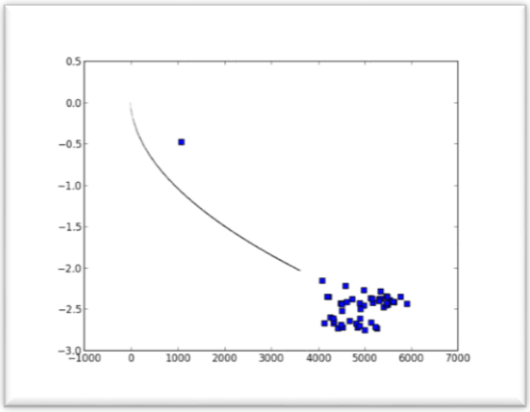


Figure 3.4(n) Self Adaptive

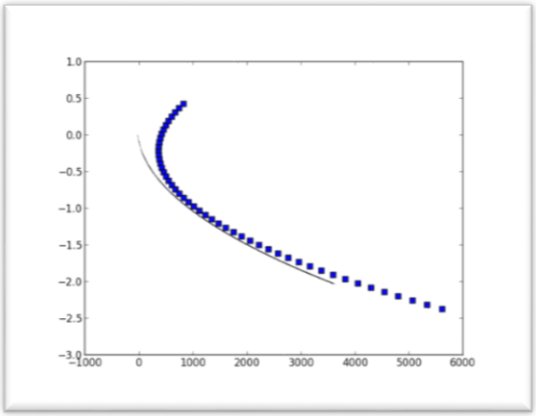


Figure 3.5(n) Stochastic Ranking

4. OSY_2D

Population Size: 30, Maximum Function Calls: 10000

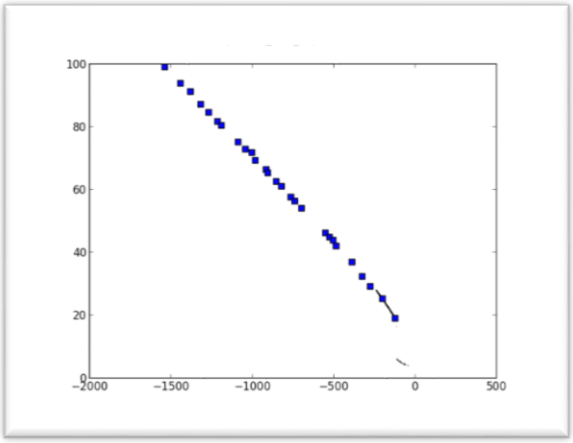


Figure 4.1(n) Dynamic Penalty

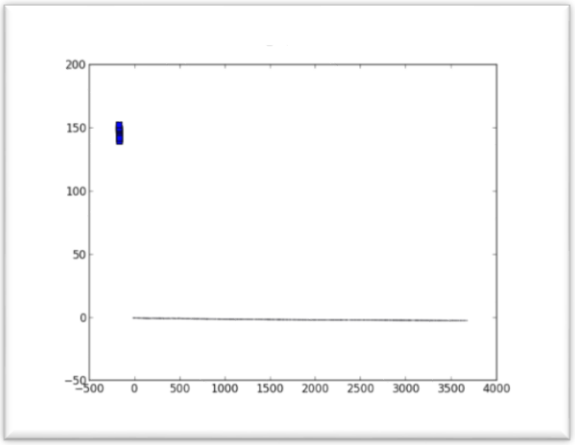


Figure 4.2(n) APM

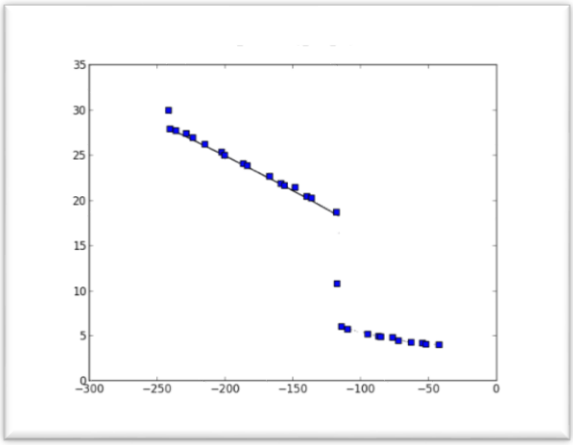


Figure 4.3(n) Constraint Dominancy

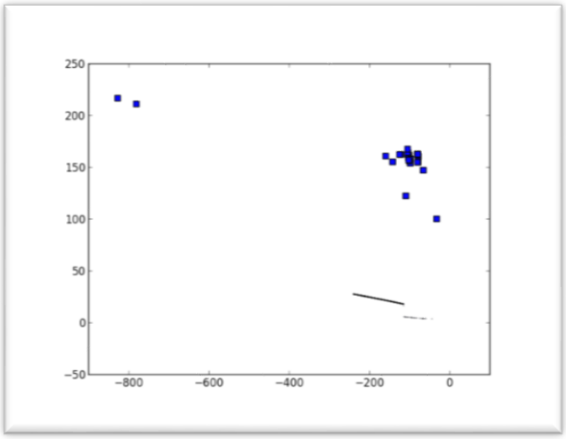


Figure 4.4(n) Self-Adaptive

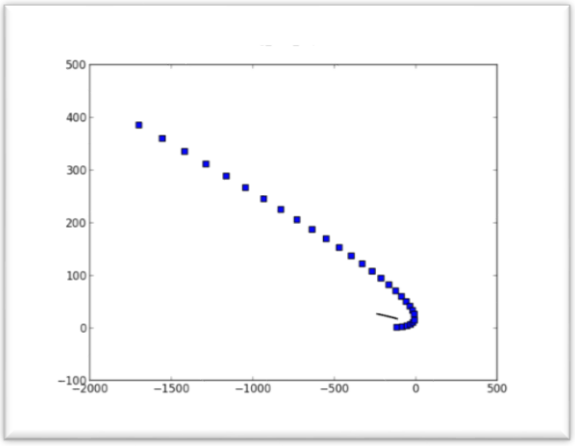


Figure 4.5(n) Stochastic Ranking

5. G19_3D

Population Size: 60, Maximum Function Calls: 25000

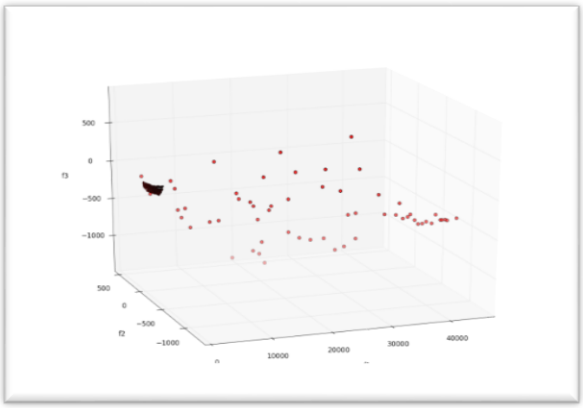


Figure 5.1(n) Dynamic Penalty

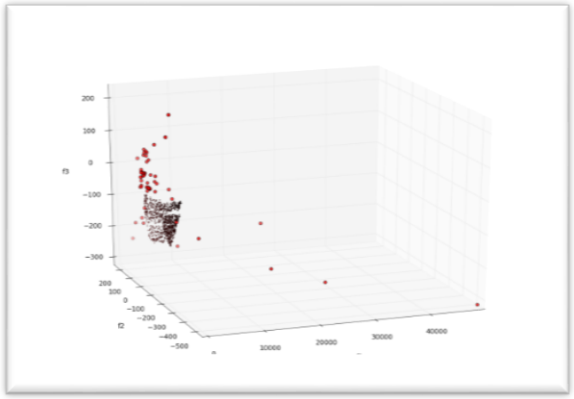


Figure 5.2(n) APM

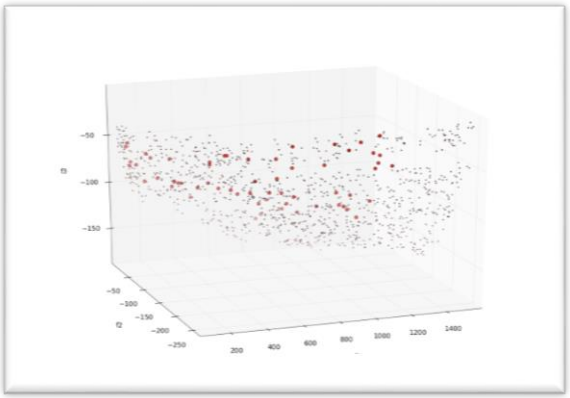


Figure 5.3(n) Constraint Dominancy

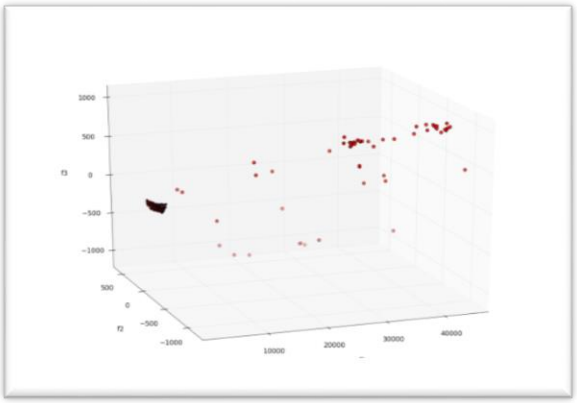


Figure 5.4(n) Self Adaptive

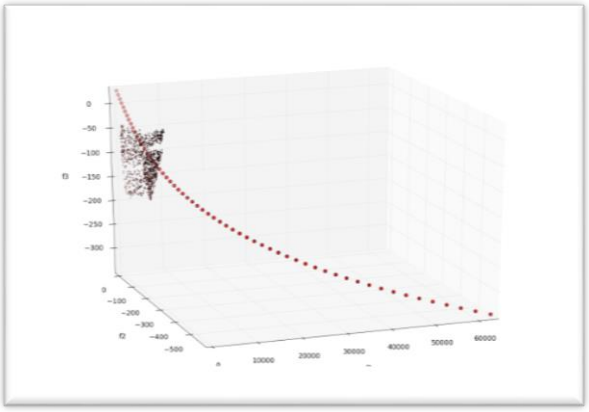


Figure 5.5(n) Stochastic Ranking

6. G7_3D

Population Size: 50, Maximum Function Calls: 15000

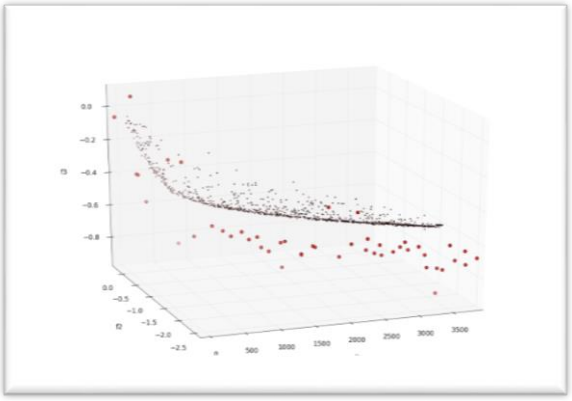


Figure 6.1(n) Dynamic Penalty

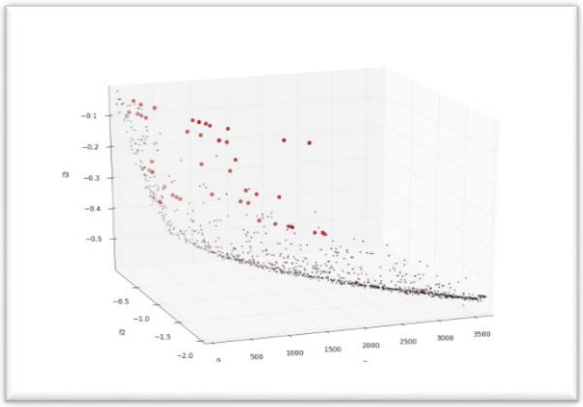


Figure 6.2(n) APM

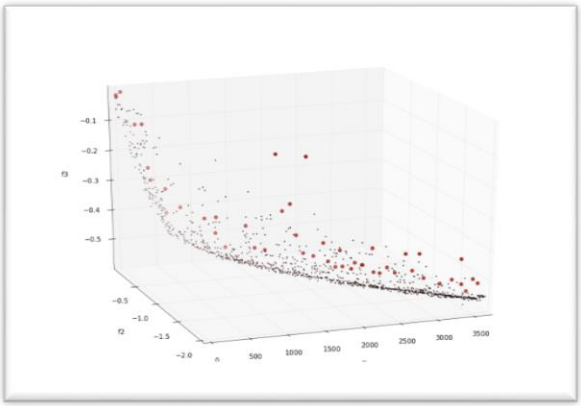


Figure 6.3(n) Constraint Dominancy

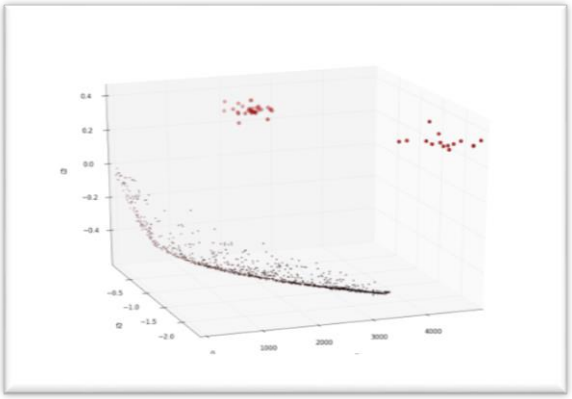


Figure 6.4(n) Self Adaptive

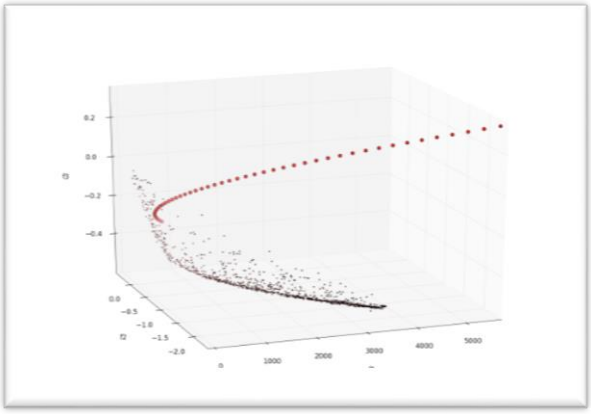


Figure 6.5(n) Stochastic Ranking

4.2**. In the context of expensive optimization via AMOEA-MAP [5][6]**

General Settings and Operators:

For all the problems Simulated Binary Crossover with probability 1, tournament selection and Importance-based Adaptive Mutation Operator (IAMO) was used with probability 0.05.

1. Beam Design

Population Size: 8, Maximum Function Calls: 200

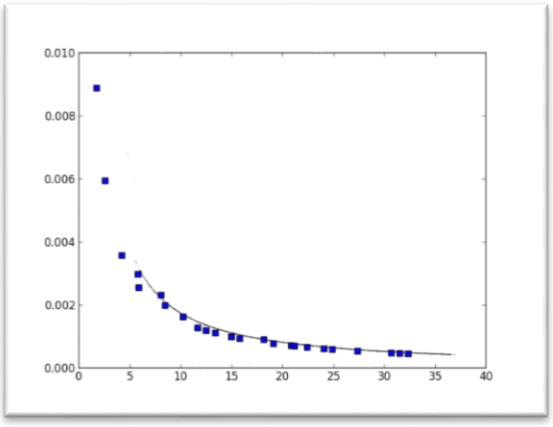


Figure 1.1(E) Dynamic Penalty

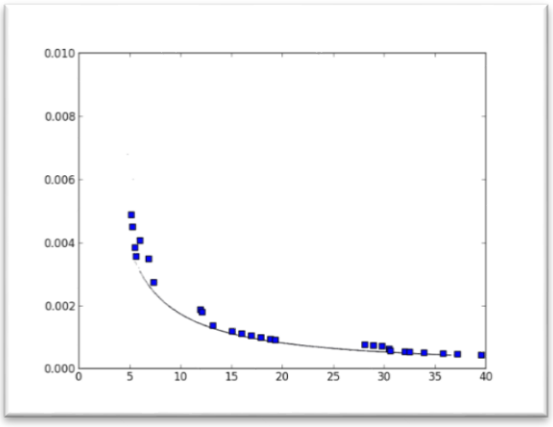


Figure 1.2(E) APM

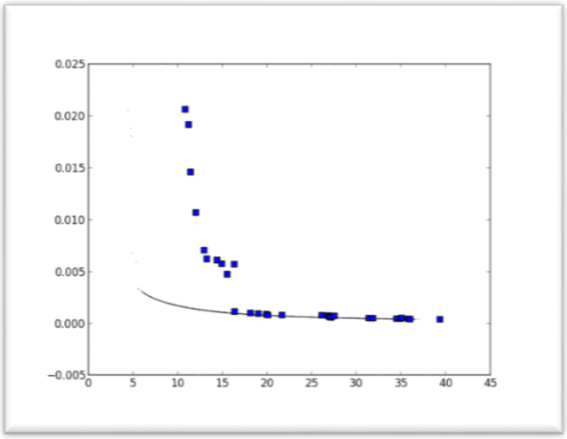


Figure 1.3(E) Constraint Dominancy

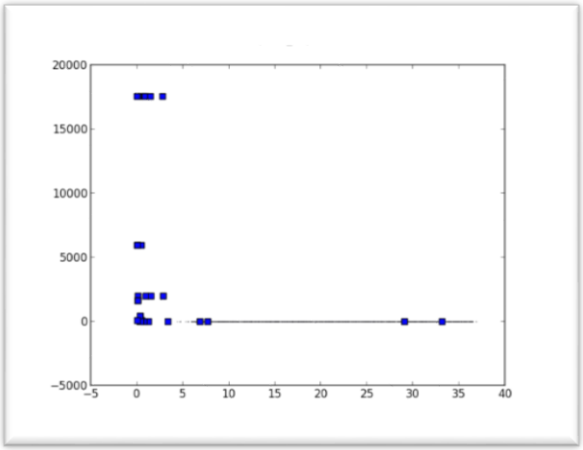


Figure 1.4(E) Self Adaptive

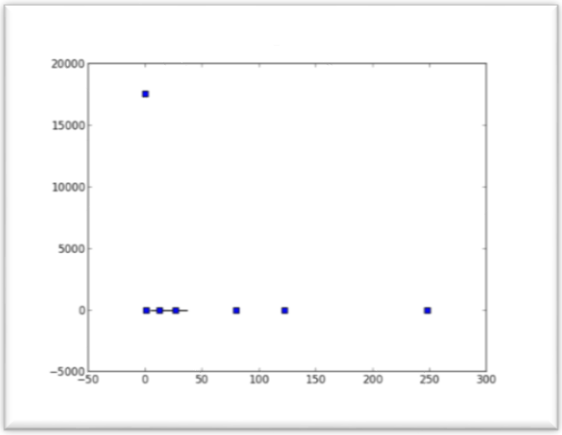


Figure 1.5(E) Stochastic Ranking

2. BICOP1_2D

Population Size: 8, Maximum Function Calls: 200

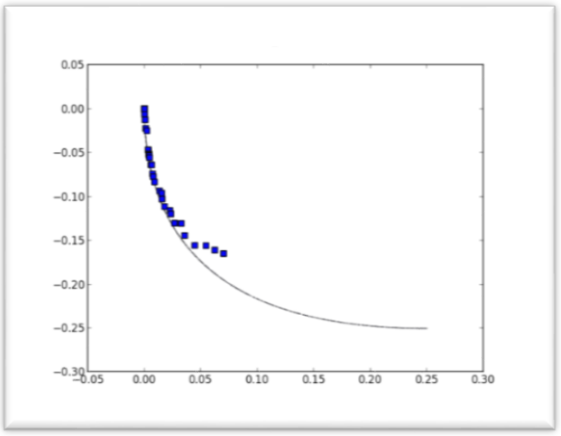


Figure 2.1(E) Dynamic Penalty

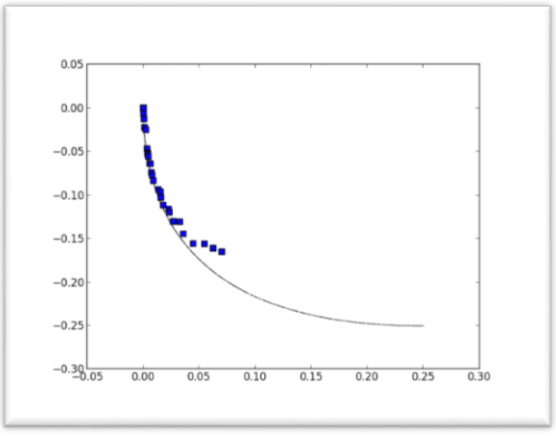


Figure 2.2(E) APM

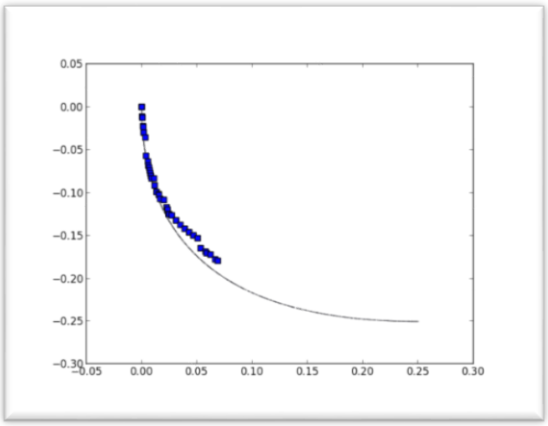


Figure 2.3(E) Constraint Dominancy

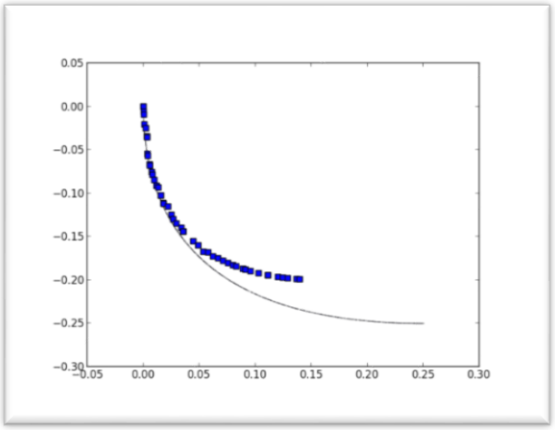


Figure 2.4(E) Self Adaptive

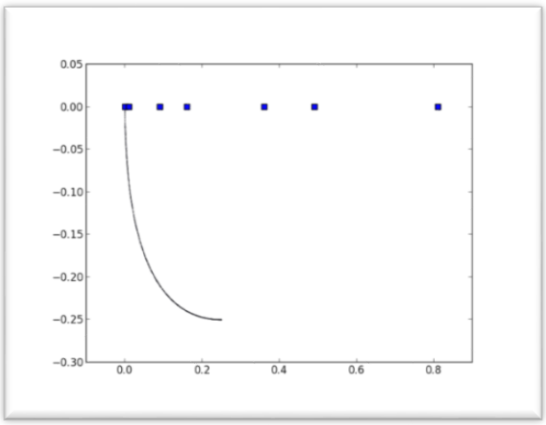


Figure 2.5(E) Stochastic Ranking

3. G7_2D

Population Size: 8, Maximum Function Calls: 400

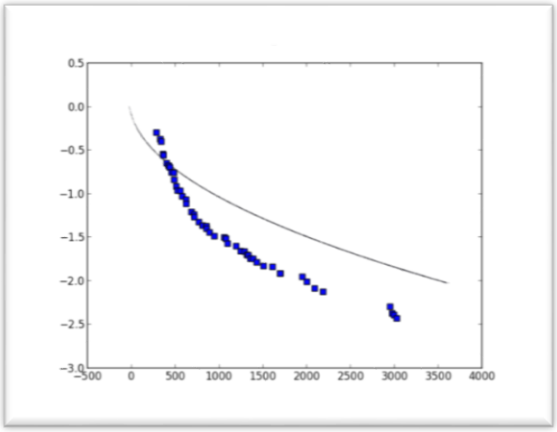


Figure 3.1(E) Dynamic Penalty

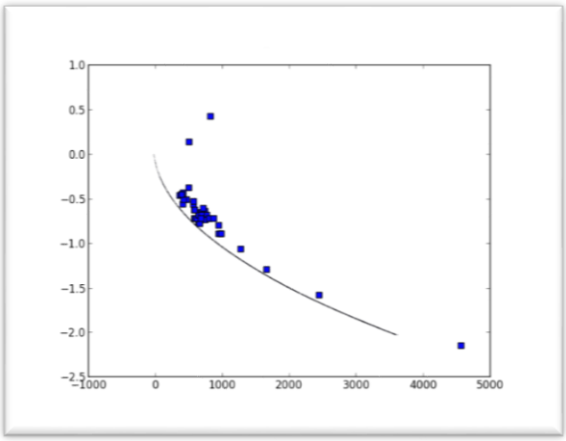


Figure 3.2(E) APM

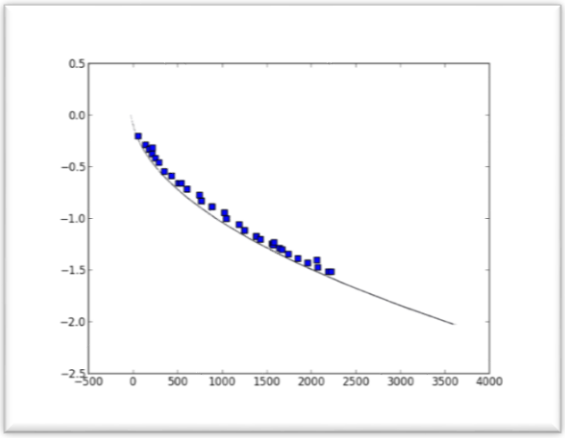


Figure 3.3(E) Constraint Dominancy

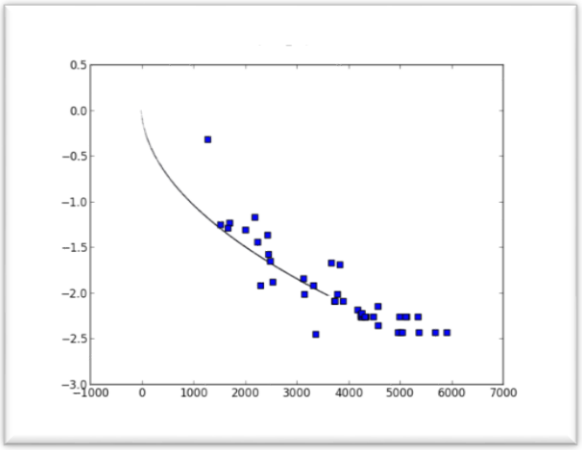


Figure 3.4(E) Self Adaptive

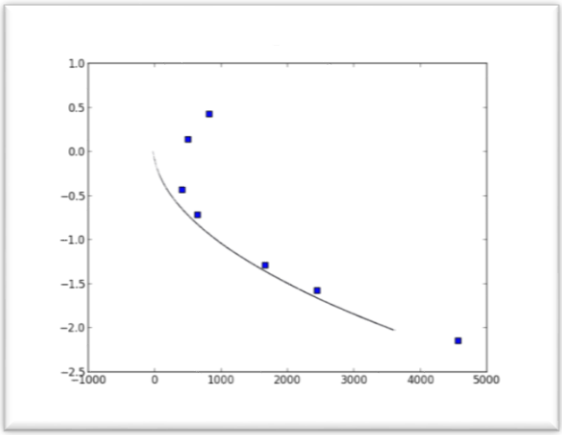


Figure 3.5(E) Stochastic Ranking

4. OSY_2D

Population Size: 8, Maximum Function Calls: 600

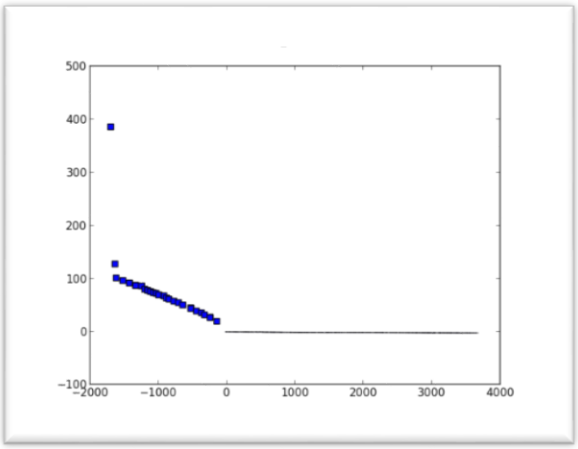


Figure 4.1(E) Dynamic Penalty

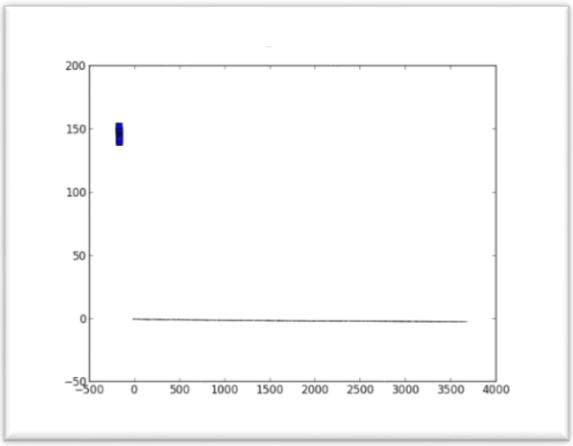


Figure 4.2(E) APM

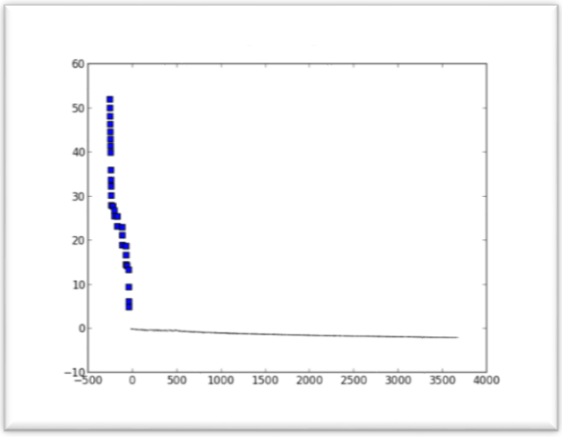


Figure 4.3(E) Constraint Dominancy

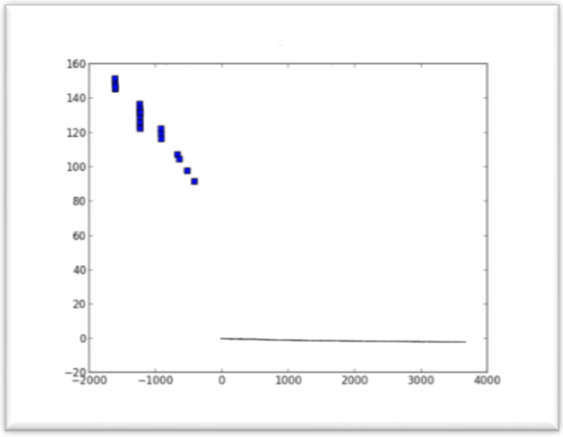


Figure 4.4(E) Self Adaptive

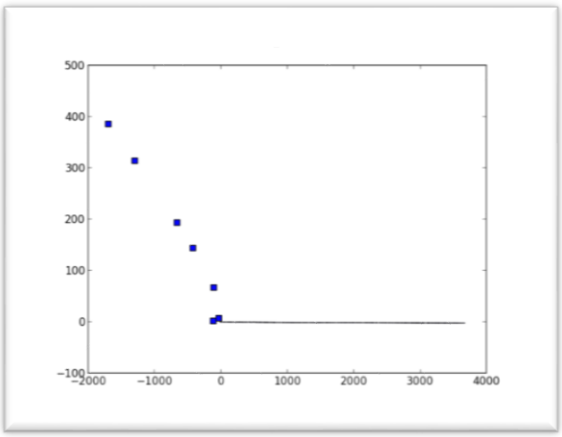


Figure 4.5(E) Stochastic Ranking

5. G19_3D

Population Size: 8, Maximum Function Calls: 200

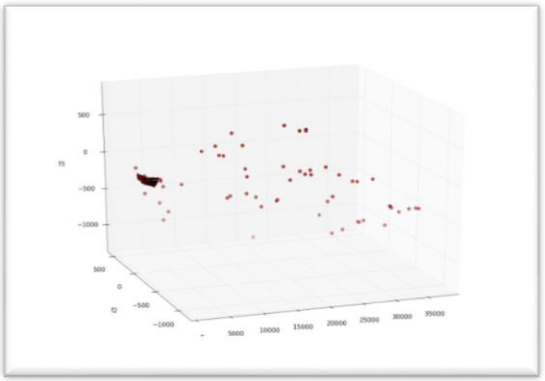


Figure 5.1(E) Dynamic Penalty

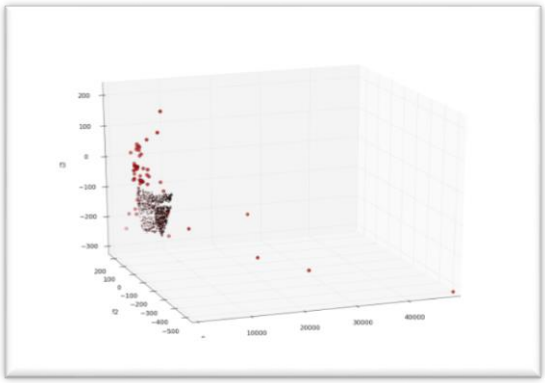


Figure 5.2(E) APM

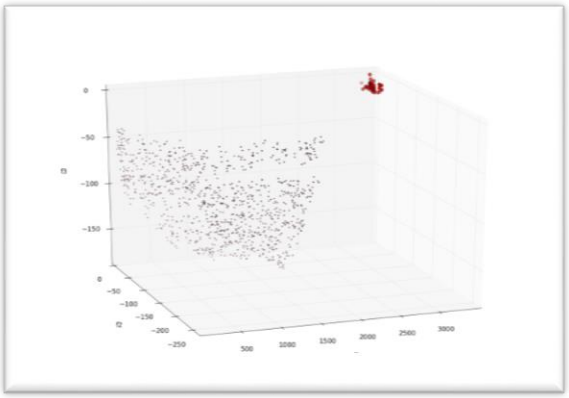


Figure 5.3(E) Constraint Dominancy

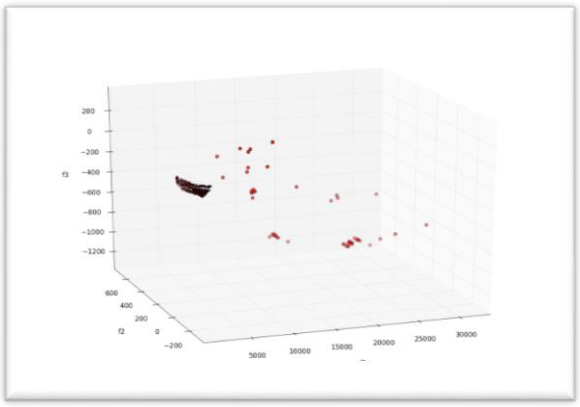


Figure 5.4(E) Self Adaptive

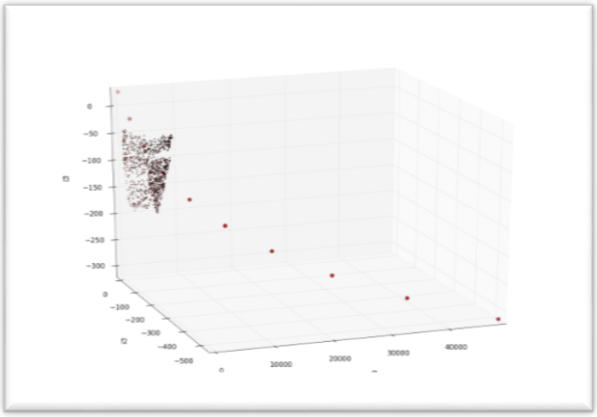


Figure 5.5(E) Stochastic Ranking

6. G7_3D

Population Size: 8, Maximum Function Calls: 400

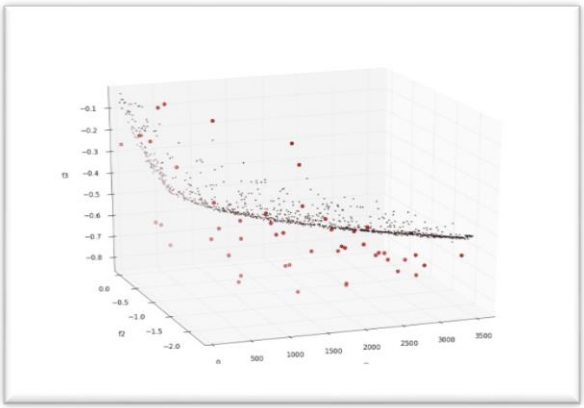


Figure 6.1(E) Dynamic Penalty

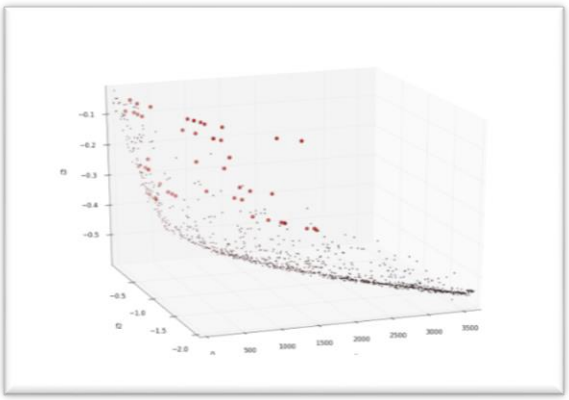


Figure 6.2(E) APM

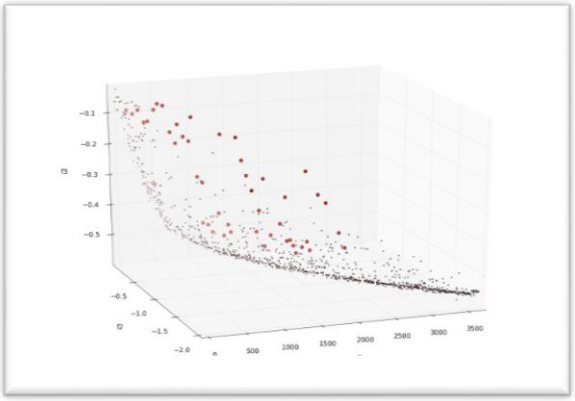


Figure 6.3(E) Constraint Dominancy

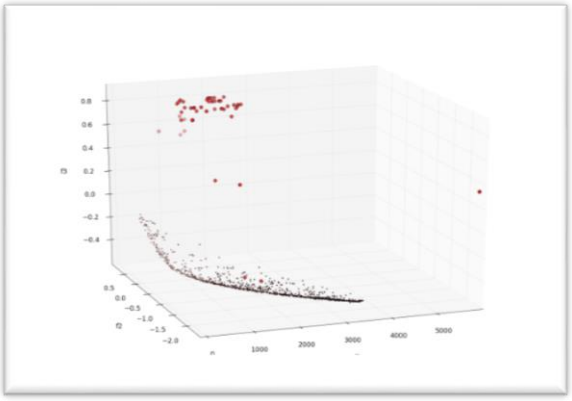


Figure 6.4(E) Self Adaptive

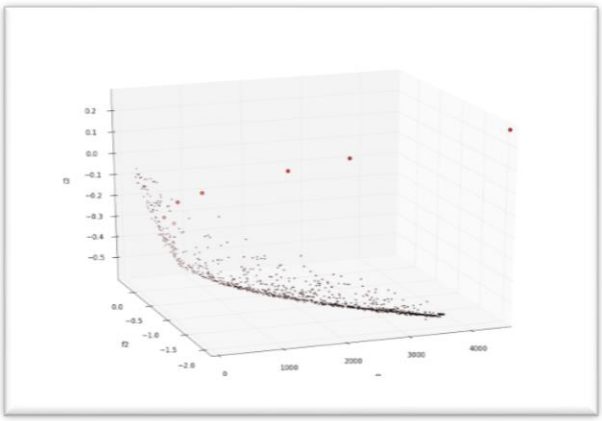


Figure 6.5(E) Stochastic Ranking

(*)In the name of figure **n** corresponds to Non expensive optimization and **E** for Expensive optimization.

5. Discussion

From the obtained Pareto fronts, it can be seen that method Constraint Dominancy works well for all the tested problems. Stochastic Ranking and Self Adaptive method performed poor in some of the problems. For the expensive optimization also Constraint dominancy method performed better than all the methods. For some of the expensive optimization, obtained Pareto front does not overlap with real Pareto front because of less function evaluations. If we increase computational budget then obtained Pareto front will overlap more with real Pareto front. From the obtained Pareto front it can also be seen that to obtain a good Pareto front Self-adaptive and APM method requires more number of function evaluations than Constraint dominancy and Dynamic penalty method. Moreover if we take a look at the number of feasible solution then constraint dominancy and dynamic penalty method wins as in dynamic penalty method, penalty weight increases with each generation so there are more number of feasible solution. For constraint dominancy method due to given priority to feasible solution over infeasible solution, there are more number of feasible solutions in the population. From a overall point of view, it can be concluded that constraint dominancy method and dynamic penalty method performs better than other method in terms of optimal pareto front and number of feasible solution in population whether it is non-expensive optimization or expensive AMOEA-MAP optimization.

6. Acknowledgments

All the 2D/3D graphical representations were produced using the MatplotLib open-source graphics environment under Python. All the programming of the methods is done under PYTHON (version 2.7).

7. **References**

1. Joines, Jeffrey A., and Christopher R. Houck. "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's." *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence. Proceedings of the First IEEE Conference on*. IEEE, 1994.

2. Barbosa, Helio JC, and Afonso CC Lemonge. *An adaptive penalty method for genetic algorithms in constrained optimization problems*. INTECH Open Access Publisher, 2008.

3. Farmani, Raziyeh, and Jonathan A. Wright. "Self-adaptive fitness formulation for constrained optimization." *IEEE Transactions on Evolutionary Computation* 7.5 (2003): 445-455.

4. Runarsson, Thomas P., and Xin Yao. "Stochastic ranking for constrained evolutionary optimization." *IEEE Transactions on evolutionary computation* 4.3 (2000): 284-294.

5. Ahmadi, Aras, et al. "An archive-based multi-objective evolutionary algorithm with adaptive search space partitioning to deal with expensive optimization problems: Application to process eco-design." *Computers & Chemical Engineering* 87 (2016): 95-110.

6. Ahmadi, Aras. "Memory-based adaptive partitioning (MAP) of search space for the enhancement of convergence in Pareto-based multi-objective evolutionary algorithms." *Applied Soft Computing* 41 (2016): 400-417.

7. Coello, Carlos A. Coello, and Efrén Mezura Montes. "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection." *Advanced Engineering Informatics* 16.3 (2002): 193-203.

8. Kusakci, Ali Osman, and Mehmet Can. "Constrained optimization with evolutionary algorithms: a comprehensive review." (2012).

9. Coello Coello, Carlos Artemio. "Constraint-handling techniques used with evolutionary algorithms." *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. ACM, 2012.

10. Runarsson, Thomas Philip. "Approximate evolution strategy using stochastic ranking." *2006 IEEE International Conference on Evolutionary Computation*. IEEE, 2006.

11. Deb, Kalyanmoy. "An efficient constraint handling method for genetic algorithms." *Computer methods in applied mechanics and engineering* 186.2 (2000): 311-338.

12. Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation* 6.2 (2002): 182-197.