Preprints of the
9th International Symposium on Advanced Control of Chemical Processes
The International Federation of Automatic Control
June 7-10, 2015, Whistler, British Columbia, Canada

Fast Mesh-Sorting in Multi-objective Optimization ⋆

Narendra Patel * **Nitin Padhiyar** **

* *Chemical Engineering Department, Vishwakarma Government Engineering College, Chandkheda, Ahmedabad-382424, Gujarat, India. (e-mail: narendra@iitgn.ac.in)*
** *Department of Chemical Engineering, Indian Institute of Technology Gandhinagar, Ahmedabad-382424, Gujarat, India. (e-mail: nitin@iitgn.ac.in)*

Abstract: A single parameter based fast mesh-sorting is proposed in this work. The single parameter algorithm as compared to non-dominated sorting eliminates the classification of the population into non-dominated fronts and calculating crowding distance. The proposed one parameter approach also provides flexibility of choosing any probability based selection operator. On the other hand, non-dominated sorting approach can only use tournament selection directly. We have considered Zitzler-Deb-Thieles (ZDT) test functions to test computational and convergence capabilities of proposed algorithm. The performance of the proposed algorithm is compared with conventional Non-dominated Sorting Genetic Algorithm-II (NSGA-II) and NSGA-II with a recent fast corner-sort algorithm. We have also considered optimal control of fed-batch reactor as Multi-objective optimization application.

Keywords: Multi-objective Optimization, Genetic Algorithm, NSGA, mesh-sort

1. INTRODUCTION

Multi-objective optimization (MOO) is a class of optimization, which deals with multiple and conflicting objectives simultaneously. Some examples of sets of conflicting objectives are: capital cost and operating cost, selectivity and conversion, quality and conversion, profit and environmental impact, and profit and safety cost. MOO problems with conflicting objectives will have a set of solutions, which are called pareto optimal solutions (Steuer, 1989). Biologically inspired evolutionary algorithms have gained significant attention for past two decades for solving MOO problems.

The pioneering work on the evolutionary approach to solve MOO was carried out in mid-1990s (Schaffer, 1985). Though, his Evolutionary Algorithm (EA) was biased towards few points on the pareto front, which was taken care by Goldberg et al. (1989) and Deb (1995). Since then, Non-dominated Sorting Genetic Algorithm (NSGA) proposed by Srinivas and Deb (1995) has been widely used. NSGA was criticized for its large computational time, non-elitism approach, and not having a sharing parameter. Deb et al. (2002) addressed these issues and presented the modified NSGA, which they called NSGA-II. Since then, NSGA-II has been widely accepted and applied Genetic Algorithm (GA) for MOO in its original and modified version in various fields.

Non-dominated sorting is an important component of pareto based multi-objective optimization. In non-dominated sorting, all the population members are classified into different ranks of pareto fronts, with 1st rank pareto front closest to the true pareto front. All the members of the first p ranks of the pareto fronts are selected for the next generation, if the sum total number of members of the p pareto fronts are less than or equal to the population size. Thus, these members have the equal fitness values for survival selection. The remaining population members are selected from the next higher rank pareto front depending upon the crowding distance criteria.

The dominance based ranking of populations requires multiple comparisons of members for sorting and hence are computational expensive. Wagner et al. (2007) demonstrated why the performance of well-established Evolutionary Multi-objective Optimization Algorithms (EMOA), NSGA-II and SPEA2 rapidly degrades with increasing dimensions. They also analysed that newer EMOA like ϵ-MOEA, MSOPS, IBEA and SMS-EMOA cope very well with high-dimensional objective spaces. Wang and Yao (2014) present computationally more efficient corner-sort algorithm for non-dominated sorting. A sorting approach based on summation of normalized objective values and diversified selection is proposed by Qu and Suganthan (2010). They observed this sorting to be faster and giving better convergence for both multi-objective evolutionary programming (MOEP) and multi-objective differential evolution (MODE). Lu and Yen (2003) proposed a rank-density-based genetic algorithm (RDGA)with ranking method with automatic accumulated ranking strategy, and a "forbidden region" concept. They used a revised adaptive cell density evaluation scheme and a rank-density-based fitness assignment technique.

We in this work follow a different sorting mechanism approach than the above mentioned sorting mechanisms for better computational efficiency. Instead of classifying

⋆ IIT Gandhinagar, Ahmedabad, Gujarat, India

the population members in various ranked pareto fronts, we divide the entire population into a multidimensional mesh in objective space. Here, the location of each population member in the mesh determines the quality of the population member. A fitness value for each population member is assigned depending upon the location of it in the m-dimensional mesh. Further, various fitness criteria such as large pareto front uniformity and the total span of the m objective functions, are also accounted for in the fitness value of the population member. Later, this fitness value of each member having m objectives is used in both, the fitness selection and survival selection steps in the algorithm.

Note that this sorting mechanism for selection purpose can be used under any population based multi objective approach. In this work, the proposed sorting mechanism has been compared with popular non-dominated sorting (Deb et al., 2002)and recently proposed sorting method, namely corner-sort (Wang and Yao, 2014) under GA framework. Zitzler-Deb-Thieles (ZDT) test problems (Zitzler et al., 2000) covering different complexities of pareto fronts have been used as test problems in this work. The evaluation of the proposed algorithm is carried out using a convergence metric, namely generational distance; the distance of the pareto front from the true pareto front. Finally, the proposed mesh-sort based GA has been applied for an optimal control of a fed-batch reactor producing glucanase (Shene et al., 1999).

The proposed algorithm is presented in Section 2 along with illustration. MOO mathematical functions and the performance measure, namely generational distance are discussed in Section 3. Results for test functions are discussed in Section 4. Industrial application along with results have been discussed in Section 5 and conclusions of this work in Section 6.

2. PROPOSED MESH-SORT BASED GA FOR MOO

A stepwise implementation of the proposed mesh-sort based GA for minimization of all objectives has been summarized as follows,

(1) Initialize population of size Np.
(2) Calculate the objective function value for each member of population.
(3) Classify the parents based on their total weight as per the mesh-sort algorithm discussed in section 2.1.
(4) Use any selection operator for selecting the population members for crossover.
(5) Carry out crossover and mutation of selected members to produce Np children.
(6) Calculate the objective function value for each member of the child population.
(7) Elitism selection: Sort the 2Np members (Np parents and Np children) using mesh-sort and select the best Np members.
(8) This completes one generation. Stop if appropriate convergence criteria has been met. Else go to 3.

The proposed algorithm uses mesh-weight assignment to sort the population for the selection. A fitness value for each population member is assigned depending upon the location of it in the Np^m-dimensional mesh. Note that the member located in the bottom left corner will be the most preferred one while that at top right corner will be the least preferred. An additional weightage is assigned to a member to account for the uniformity and distribution. Also, an additional weight is provided to a member having the best value along each dimension in the objective space. The stepwise algorithm for mesh weight assignment to the population members for mesh-sorting is discussed in the next sub-section.

2.1 Mesh-weight assignment

- *Step 1: Construct an equi-distance mesh of the dimension Np^m in objective space. Thus, for two objectives, the mesh will contain Np strips along each of the two objectives.*
- *Step 2: Assign rank in descending order from Np (best) to 1 (worst) to all the grids along each objective dimension. Note that each population member carries rank along each of the m dimensions.*
- *Step 3: The mesh weight (mw) for a member located in the Np^m in objective space is the sum total of all the ranks.*
- *Step 4: Best weight (bw): Assign a weight of Np to the best member along each objective.*
- *Step 5: Strip weight (sw): Select an objective dimension (from j=1 to m) and select one best member for each of the Np rank-strips. The best member for a given strip is computed based on the $j+1^{th}$ objective function for j=1 to m-1. On the other hand for the last objective dimension, the best member is chosen depending upon the 1st objective function. Assign strip weight (sw) equal to Np to the best member if it is improving in next dimension.*
- *Step 6: Neighbour weight (nw): Every member getting strip weight is also assigned a weight proportional to dimensionless distance in terms of number of strips. The dimensionless distance is multiplied by a factor. This multiplying factor is Np for first Np generations and the factor is generation number (nGen) after Np generations.*
- *Step 7: The total weight for each population member is the sum total of all the individual weights, namely mesh weight (mw), best weight (bw), strip weight (sw), and neighbour weight (nw). Note that the member having the maximum total weight will be the most preferred one.*

2.2 Illustration of mesh-weight assignment

Assignment of the fitness value to each population member is explained as follows using a fictitious example with two dimensional space. The two objective function values of each of the ten population members (Np) are shown in table (1) and marked by letters A to J in Fig. 1. The weight assignment is as follows:

- The minimum and maximum values of f_1 are 20 and 30, respectively, while that of f_2 are 10 and 40.
- Divide the f_1 space of [20, 30] in 10 (Np) equal sized compartments, each of span one.
- Divide the f_2 space of [10, 40] in 10 (Np) equal sized compartments, each of span three.

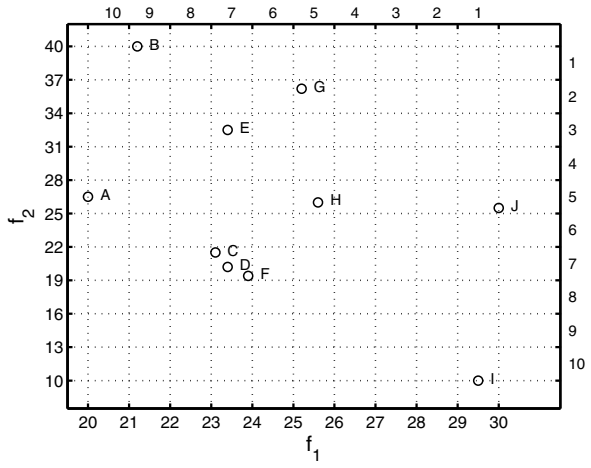

Fig. 1. Population in two dimensional space for Illustration of mesh weight assignment

- Number the rows in increasing order starting from 1 at the top and 10 at the bottom.
- Number the columns in increasing order starting from 1 at the right 10 at the left.
- Mesh weight (mw): Column number represents the mesh weight along f_1 and row number represents mesh weight along f_2. These entries are shown in columns 4 and 5 in table (1).
- Best weight (bw): Best member along f_1, point A is assigned a weight of 10 equivalent to population size, and similarly best member along f_2, point I is assigned a weight of 10. The corresponding entries are shown in the last column in table (1).
- In case where multiple members belong to one row or column a strip weight (sw) is assigned to the maximum of one of them. Members A, C and I are assigned $sw = 10$ (Np) along f_1 and I,F and A are assigned $sw = 10$ (Np) along f_2. The corresponding entries are shown in the second and third columns of table (2).
- For getting uniformity in distribution another neighbour weight(nw) is assigned along each dimension. Normalized distance (distance divided by mesh span) between A to C is multiplied with 10 (Np) and assigned to C as neighbour weight. The corresponding entries are shown in the fourth and fifth columns of table (2).
- All the weights are summed as total weigh and shown in last column in table (2). Population is sorted based on the total weight. Here, C is the best member and J is the worst member.

3. MOO TEST FUNCTIONS AND PERFORMANCE EVALUATION METRIC

We have used five Zitzler-Deb-Thieles (ZDT) test problems [ZDT 1, ZDT2, ZDT3, ZDT4, and ZDT6] suggested by Zitzler et al. (2000) as test functions for evaluating the proposed sorting algorithm. Since they are well known test functions we skip the detail of the selected MOO test functions here. All the problems have two objective functions, which are to be minimized. Each test function presents certain difficulties for multi-objective optimisation.

Performance metrics are important performance assessment measure, which also allow us to compare algorithms. Deb (2001) classifies them in three categories, metrics evaluating closeness to the pareto optimal front, metrics evaluating diversity amongst non-dominated solutions and metrics evaluating closeness and diversity. We in this work choose Generational Distance (GD) metric (γ) to represent convergence to true pareto front defined as follows,

$$\gamma = \frac{\left(\sum_{i=1}^{|Q|} d_i^p\right)^{1/p}}{|Q|} \tag{1}$$

where Q represents solution set having $|Q|$ members. we use p=2 and d_i is minimum distance between the member in solution set and nearest member is true pareto set, which is defined as.

$$d_i = min\sqrt{\sum_{m=1}^{M}\left(f_m^{(i)} - f_m^{*(k)}\right)^2} \tag{2}$$

where M represents number of objectives, i and k represent member index in solution set and true pareto set respectively.

4. RESULTS AND DISCUSSION FOR TEST FUNCTIONS

Real coded GA program for MOO developed in MATLAB 2011 is used in this work. It uses the proposed mesh-sort for fitness calculation, tournament selection, simulated binary crossover (SBX) and non-uniform mutation with elitism survival selection operators. The NSGA-II code uses same simulated binary crossover, SBX (with $\eta_c = 20$,

Table 1. Mesh weight calculation for population in two dimensional space

Label	f_1	f_2	mw-f_1	mw-f_2	bw
1	2	3	4	5	6
A	20.0	26.5	10	5	10
B	21.2	40.0	9	1	0
C	23.1	21.5	7	7	0
D	23.4	20.2	7	7	0
E	23.4	32.5	7	3	0
F	23.9	19.4	7	7	0
G	25.2	36.2	5	2	0
H	25.6	26.0	5	5	0
I	29.5	10.0	1	10	10
J	30.0	25.5	1	5	0

Table 2. Strip weight and neighbour weight calculation for population in two dimensional space

Label	sw-f_1	sw-f_2	nw-f_1	nw-f_2	total wt
1	2	3	4	5	6
A	10	0	0.00	35.20	70.20
B	0	0	0.00	0.00	10.00
C	10	0	74.60	0.00	98.60
D	0	0	0.00	0.00	14.00
E	0	0	0.00	0.00	10.00
F	0	10	0.00	45.62	69.62
G	0	0	0.00	0.00	7.00
H	0	0	0.00	0.00	10.00
I	0	10	64.17	0.00	95.17
J	0	0	0.00	0.00	6.00

crossover probability 0.90),non-uniform mutation (with b = 4, mutation probability 0.2) and elitism survival selection operators along with non-dominated sorting, crowding distance calculation and binary tournament selection operators as recommended in the NSGA-II (Deb et al., 2002). Corner-sort MOGA uses all same operators replacing non-dominated sorting with corner-sort (Wang and Yao, 2014). Population size is kept 100 for all runs. As GA is a stochastic optimization technique, it does not converge to the same solution every time even with the same initial population. Hence, we carry out ten simulation runs for every combination of the test application with different initial population. The results reported are average of ten simulation runs.

4.1 Performance comparisons based on computational time

Since the selected test problems have known true pareto fronts, it is possible to evaluate convergence as the performance metric, γ. Convergence metric γ for ZDT1 function is presented in Fig. 2. The generation wise convergence for all three algorithms are observed very close to each other with a marginal improvement by the mesh-sort GA after 50 generations. Though, the major contribution of the proposed mesh-sort algorithm is visible when the convergence plot is made as a function of CPU time. While corner-sort algorithm provides faster convergence, the proposed mesh-sort is the fastest in convergence. The convergence (γ=0.01) achieved using NSGA-II at 5 s has been achived at 4 s using the corner sort (20% faster) while it is 2 s (60% faster) using the proposed mesh-sort algorithm.

Similar convergence results for ZDT2 test function are presented in Fig 3. Approximately 50% saving in computational time has been achieved using corner-sort compared to NSGA-II, while it is 75% using the mesh-sort approach. CPU time for all test cases at the end of 100 generations are summarised in table (3) along with % CPU time saving with corner-sort and mesh-sort algorithms as compared to NSGA-II. It clearly reflects that the mesh-sort MOGA needs less time for 100 generations consistently. To analyse this results in more details we present effect of population size on computational efforts for proposed algorithm and compare it with corner-sort and NSGA-II.

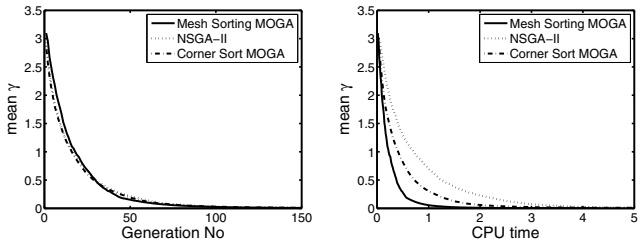

Fig. 2. Generation wise convergence(left) and convergence as a function of CPU time(right) for ZDT1

4.2 Effect of population size on computational time

The population size is very important parameter which plays role on computations required for sorting. Population size effect on average CPU time required for one

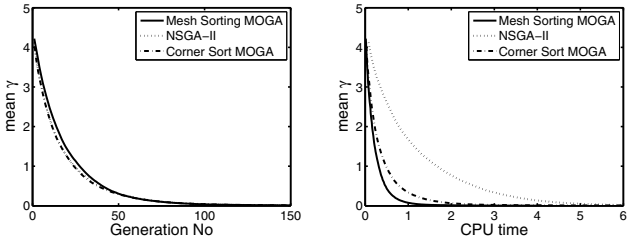

Fig. 3. Generation wise convergence (left) and convergence as a function of CPU time (right) for ZDT2 function

generation for mesh-sort MOGA, NSGA-II and corner-sort MOGA algorithms are presented for ZDT1 and ZDT2 functions in Fig. 4. As can be observed from left figure for ZDT1 function, the difference in computational effort becomes more prominent with larger population size. For a population size of 500 while NSGA-II takes 0.69 s, corner-sort takes 0.39 s and mesh-sort takes only 0.07 s. Similar observations can be made for right side plot of ZDT2 function. It should be noted that the difference is expected to increase even more drastically beyond 500 population size.

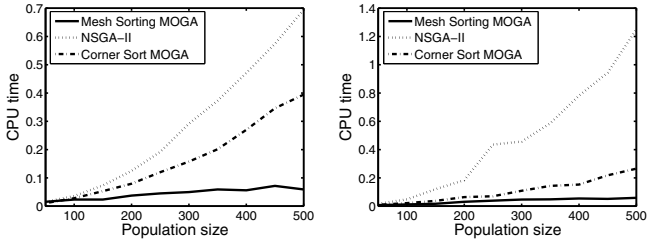

Fig. 4. Effect of population size on one generation CPU time for ZDT1 (Left)and ZDT2 (right) functions

4.3 Convergence to true pareto front

The pareto front obtained for one of the run for ZDT1 function at the end of 100 and 250 generations are presented along with true pareto fronts in Fig. 5. Similar pareto front plots for ZDT2 function are presented in Fig. 6. Visually it is very difficult to observe convergence from those pareto front plots and hence we use generational distance(GD) metric (γ) to compare convergence. The γ metric average values at the end of 100 generations for all test cases are summarised at table (4). Except function ZDT4 all others show better convergence metric values for mesh-short MOGA. The ZDT4 function is having closely located (closer than the mesh size) multiple pareto fronts which makes it difficult to show better or equivalent converge for mesh-sort as compared to NSGA-II at same number of generations. While we compare performance of

Table 3. Average CPU time for 100 generations using different sorting algorithms

function	NSGA-II	corner-sort	% saving	mesh-sort	% saving
ZDT1	3.85	2.67	30.84	1.39	64.02
ZDT2	5.25	2.37	54.86	1.27	75.88
ZDT3	3.93	2.75	29.92	1.42	63.97
ZDT4	6.89	1.66	75.88	0.80	88.41
ZDT6	4.88	2.31	52.72	1.00	79.62

algorithm based on CPU time, mesh-sort gives the best result compared to other two algorithms for all functions. Performance of the proposed algorithm is tested for a fed-batch reactor control application in next section.

Table 4. Average convergence at the end of 100 generation as Generational Distance (GD)

function	NSGA-II	Mesh Sort
ZDT1	0.0303	0.0248
ZDT2	0.0398	0.0370
ZDT3	0.0152	0.0147
ZDT4	9.0985	12.7532
ZDT6	0.0726	0.0265

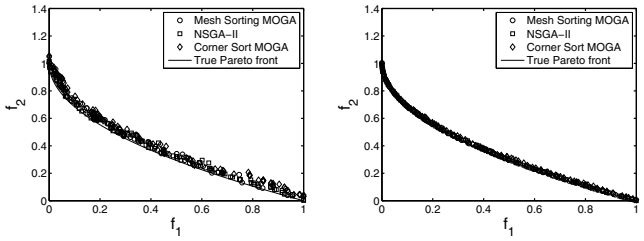

Fig. 5. Pareto front for ZDT1 function at the end of 100 generation (left) and 250 generation(right)

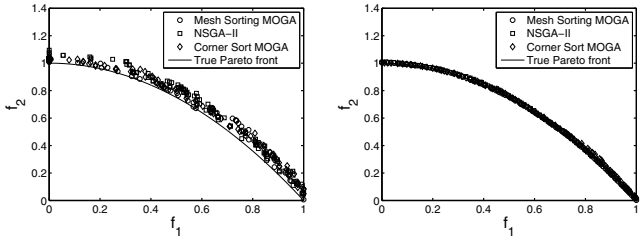

Fig. 6. Pareto front for ZDT2 function at the end of 100 generation (left) and 250 generation(right)

5. OPTIMAL CONTROL OF FEB-BATCH REACTOR

Shene et al. (1999) studied the kinetics for Bacillussubtilis for the synthesis of a recombinant β-1,3-glucanase enzyme synthesis. They studied effect of different nutrient feeding strategies on the synthesis of β-1,3-glucanase and protease enzymes. We skip the detailed mathematical model description here for the sake of brevity. The multi-objective optimal control problem in the fed-batch reactor has been formulated in section 5.1 and the results are presented in section 5.2.

5.1 MOO for fed-batch reactor

We here are considering multi objective optimal control problem for fed-batch fermenter, where productivity and yield are to be maximised simultaneously for recombinant β-1,3-glucanase enzyme synthesis. The productivity is defined as the amount of product formed per unit time while the yield is defined as the product per unit amount of substrate fed. The mathematical definitions of the productivity and the yield are shown in (3) and (4), respectively. The time history of substrate feed and fed-batch time are considered to be manipulated variables for maximizing the two control objectives.

$$J_P = \frac{P(t_f)}{t_f} \tag{3}$$

$$J_Y = \frac{P(t_f)}{\int_0^{t_f} F(t) S_F dt} \tag{4}$$

Constrains imposed on the fermenter volume and feed rate are $V(t) \leq 2.5$ L and $0 \leq F(t) \leq 0.5$ L/h, respectively. The lower and upper bounds on the fed-batch time are 4 and 6 hours, respectively. The initial conditions are X=0.16 g, G=0 g, P=0 g, S=5 g, V=0.85 L. X, G, P and S are amount of biomass, β-1,3-glucanase, protease and substrate, respectively. V is fermenter volume; F is substrate volumetric feed rate of concentration S_F.

5.2 Result and Discussion for fed-batch reactors

Since the selected problems does not have known true pareto front, an expected pareto front is generated by running GA for large number of generations. Mesh-sort MOGA and NSGA-II are run for 1000 generations with each 500 population size. The final populations of both the runs are mixed and the resultant pareto of the combined population is considered as the expected pareto front for this optimal control problem. The generation wise convergence profile and the pareto fronts by all the three MOGAs at the end of 100 generations are shown in Fig. 7. The pareto front shown is for one of the ten runs and convergence is average of ten simulation runs. The overlapping generation wise convergence plot reflects generation wise closeness of convergence.

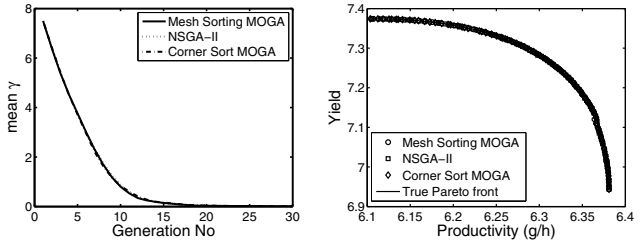

Fig. 7. Generation wise convergence(left) and pareto front at 100 generation(right) for Glucanese application

There is no significant distinction among the three MOGAs in the generation wise convergence graphs shown in Fig. 7. Though the convergence profile as a function of CPU time shown in Fig. 8 reflects that there is faster convergence rate using mesh-sorting approach compared to the other two. Note that the corner-sort MOGA convergence faster till 30 s and merges with the NSGA-II afterwards. The figure also shows approximately 20% reduction in computational effort for mesh-sort compared to NSGA-II and corner-sort. When function evaluation for application uses dynamic simulation, such reduction in computational time is remarkable. The time spent on function evaluation is much more higher than the time for sorting and hence for similar convergence, the 20% computational time saving is appreciable. Mesh-sort MOGA, corner-sort MOGA and NSGA-II converge to identical

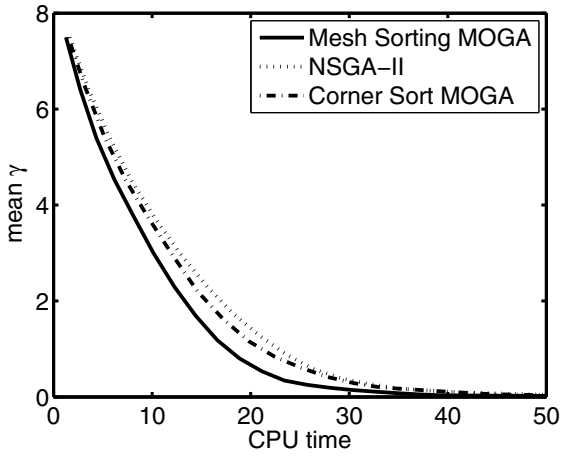

Fig. 8. Average convergence metirc GD (γ) as a function of CPU time for Glucanase application

values of metric γ=0.0004 at the end of 100 generations. The application with large computational requirement for objective function evaluation also gives faster convergence is a clear contribution by mesh-sort algorithm.

6. CONCLUSION

Mesh-sort algorithm has been proposed for population based multi-objective optimization problems. This algorithm eliminates the requirement of classifying the whole population into non-dominated fronts with different ranks. In non dominated sorting approach every population member's fitness is identified in terms of the two parameters, namely front number and crowding distance. In the proposed mesh-sort approach each population member obtains mesh weight contribution by the relative position of the member in the mesh. Additional strip weight and neighbour weight account for convergence rate and uniform spread of the members in pareto front. Thus the single fitness value can be used for the survival and fitness selection steps in MOGA. Though, this sorting can be applied for any population based MOGA, we show the results under GA framework in this study. Selection based on total mesh-weight indirectly selects better members which are non-dominated.

The proposed mesh sorting based selection has been compared with the widely popular NSGA-II and recently proposed corner-sort based MOGA (Wang and Yao, 2014). While the generation wise convergence rate by the mesh-sort approach has been observed to be very close to the other two sorting approaches, there is significant gain in the computational time for sorting the population. As a result, the convergence rate in terms of CPU time has been found to be considerably higher with the mesh-sort approach. Further, the single parameter based selection provides flexibility in using any selection operator as against tournament selection for NSGA-II. Proposed mesh-sort MOGA is tested with ZDT test problems and observed to be computationally more efficient. Finally the proposed MOGA has been applied to an optimal control of a fed-batch reactor, where productivity and yield are optimized by optimizing the substrate feed recipe.

ACKNOWLEDGEMENTS

We acknowledge the Directorate of Technical Education, Gujarat State and IIT Gandhinagar for supporting this research work.

REFERENCES

Deb, K. (1995). *Optimization methods for engineering design*. Prentice-Hall, New Delhi, India.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2), 182–197.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*, volume 16. Wiley.

Goldberg, D.E., Korb, B., and Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3(5), 493–530.

Lu, H. and Yen, G. (2003). Rank-density-based multiobjective genetic algorithm and benchmark test function study. *Evolutionary Computation, IEEE Transactions on*, 7(4), 325–343.

Qu, B. and Suganthan, P. (2010). Multi-objective evolutionary algorithms based on the summation of normalized objectives and diversified selection. *Information Sciences*, 180(17), 3170 – 3181. Including Special Section on Virtual Agent and Organization Modeling: Theory and Applications.

Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, 93–100. L. Erlbaum Associates Inc.

Shene, C., Andrews, B., and Asenjo, J. (1999). Fedbatch fermentations of bacillus subtilis toc46 (ppff1) for the synthesis of a recombinant -1,3-glucanase: experimental study and modelling. *Enzyme and Microbial Technology*, 24(56), 247 – 254.

Srinivas, N. and Deb, K. (1995). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3), 221–248.

Steuer, R.E. (1989). *Multiple Criteria Optimization: Theory, Computation, and Application*. Krieger Publishing Company.

Wagner, T., Beume, N., and Naujoks, B. (2007). Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata (eds.), *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, 742–756. Springer Berlin Heidelberg.

Wang, H. and Yao, X. (2014). Corner sort for pareto-based many-objective optimization. *Cybernetics, IEEE Transactions on*, 44(1), 92–102.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2), 173–195.