A Fast Algorithm for Finding the Non Dominated Set in Multi objective Optimization

K.K.Mishra and Sandeep Harit
MNNIT,Allahabad,India

INTRODUCTION:

The working of single objective optimization algorithm and multi objective optimization algorithm is quite different. This difference is due to number of optimal solution approached by both the algorithms. In single objective optimization problem there will be a single optimal solution, even though in multi model optimization there may be more than one solution but we are interested in only one optimal solution, where as in multi objective optimization problem, there will be many set of optimal solutions. These sets are called different non dominated front, and every non dominated front will contain a set of non dominated solutions thus there are two tasks of an ideal multi objective optimization algorithm (i) To find multiple non dominated fronts (or to identify different set of non dominated solutions). (ii) To seek for Pareto optimal solutions with a good diversity in objective and decision variable values.

In this paper, we explain a new algorithm for finding non dominated set of a multi objective optimization problem. In literature many algorithms are used for this task like naïve and slow method [2], fast and efficient method [3] and Kung et al method [7]. Recently two new algorithms are proposed by Ding [4] and Jun Du[1].The worst case time complexity of all algorithm(including recently proposed algorithm) is $(OM(N)^2)$,Previously Kung's algorithm was best in its average and best case time complexity but Jun du in 2007 proved that his algorithm is best in comparison of kung's algorithm. While we were unable to reduce worst case time complexity, The best case time complexity of proposed algorithm is O(NLog(N))(for any number of objective functions)which is a improvement as compared to othere algorithms. Also it follows a simple approach, no hectic summation and production method.

The paper is organized into four sections. Section 2 presents some background detail of Different Preexisting Algorithms and specifies the necessary definition related to non dominated set. Section 3 describes the proposed approach and stepwise algorithm also difference between Kung's algorithm and proposed algorithm is presented; In Section 4, an experimental analysis and complexity of the proposed algorithm are presented. Finally, Section 5 concludes the paper.

2.BACKGROUND

2.1Dominance and Pareto-Optimality

Most multi-objective optimization algorithms use the concept of dominance in their search. Here, we define the concept of dominance and related terms and present a number of techniques for identifying dominated solutions in a finite population of solutions. Definition of dominated points and non-dominated set are given below

Definition 1: A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$, if both conditions 1 and 2 are true:

1. The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives, or $f_j(x^{(1)})$ $f_j(x^{(2)})$ for all j = 1,2…M.
2. The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective, or $f_j(x^{(1)})$ $f_j(x^{(2)})$ for at least one $j \in \{1,2\ldots M.\}$.

Definition 2: (Non-dominated set): Among a set of solutions P, the non-dominated set of solutions P' are those that are not dominated by any member of the set P

2.2. Review of some standard algorithms

Before discussing the proposed algorithm, let us review some preexisting algorithms.

2.2.1Kung's Algorithm:

Kung algorithm is the most efficient and widely used one. In this approach we have to first sort the population in decending order in accordance to first objective function. Thereafter, the population is recursively halved as top(T) and bottom(B) subpopulations. As Top half (T) is better in objective in comparison to Bottom half (B) in first objective ,so we check the bottom half for domination with top half .The solution of B which are not dominated by solutions of T are merged with members of T to form merged population M.The complete algorithm is given below:

Step 1: Sort the population according the descending order of importance in the first objective function and rename the population as P of size N.

Step 2 : Front(P): if |P| =1, return P as the output of Front(P).Otherwise, T= Front($P^{(1)}$ - $P^{(\lfloor |P|/2 \rfloor)}$) and B= Front($P^{(\lfloor |P|/2+1 \rfloor)}$ - $P^{(P)}$). IF i^{th} non –dominated solution B is not dominated by any non-dominated solution of T, create a merged set M=T U i. Return M as output of Front(P).

2.2.2Sorting based algorithm (Jun Du Algorithm):

This algorithm is given in two steps as follows:

Step 1: the population of solutions is sorted according to the descending order to every objective functions. Thereafter, several solutions sequences could be presented and each of them corresponds to one objective. Every solution could be scored according to the position it takes, higher score it gains. Then, each solution could get various

scores corresponding to various objectives. Take the scores summation of every solution, and sort it to get one new solution summation sequence. The new summation sequence could be used for finding non dominated set by deleting all the dominated solution inside.

Step 2: The bottom of the summation sequence is used as the start of the compared solution, while the top of the summation sequence is used as the start of the comparing one. Once the compared solution is observed by any one anterior to it, it is deleted. When the compared solution becomes the top one of the sequence, the dominated set is drawn out.

2.2.3Ding's algorithm: Based on the definitions of indices, rank set and propositions given in paper 4, the following procedure is used by Ding which identifies the non dominated solutions. Interested reader may refer paper[4] for further study.

Algorithm *Identifying the Non-dominated Set: MLNFC.*
Input: X.
*Output:The non-dominated set —X*0 *of X*
*(R*0 *store the non-dominate set of rank set R).*
*Using quick-sort method to create indices I*1; *I*2; :::; *I*M *and*
rank set R;
*Initiate R*0 *with I*1(1)0s*, I*2(1)0s*, ..., I*M(1)0s*, eliminate duplicate ones;*
for(i = 1; i·N; i++) Set r(i) *not checked;*
stop = N;// Set initial termination position N*;*
*for(i=2;i·stop;i++)*f
//Search at the ith entries of the indices;
*for(j=1;j·M;j++)*f*//Search at the ith entry of index I*j *;*
*if(I*j(i)0s *not checked)*f
Compare it with the non-dominated ones among
*I*j(1)0s; *I*j(2)0s; :::; *I*j(i ¡ 1)0s *and set it checked;*
*if(I*j(i)0s *is not dominated by any of them)*f
*put it in R*0*;*
i0 *=max of its ranks(or components);*
*stop=min*f*stop,* i0g*;//Update termination position;*
}
}
}
}

3. PROPOSED ALGORITHM:

The proposed approach is very different from existing algorithms. In existing algorithms, we are able to classify solutions only after finding all dominated solutions. So to find non dominated solutions, we need to search entire set repeatedly. In proposed algorithm, we store only non dominated solutions. We perform a few comparisons to classify a point in to either dominated set or non dominated set. We first sort population according to the descending order of importance to the first objective value. In this way the solutions which are good in first objective will come first, in the list those having bad value will come in last. We initialize a set S1, for keeping non dominated solutions only. We start with the first solution and add this solution to non dominated set S1. Since first point is best in terms of first objective so no point can dominate this point in first objective, so it will be non dominated. Now we compare every other solution of the list with this set S1 and update this set when we find another non dominated solution and skip on those solutions which are dominated by any element of the set. For example if solutions in the list are unique in first objective function value, then for second point we need only one comparison to decide whether this is dominated or non dominated. The reason can be explained as follows, this solution can be dominated by only first solution (which is best in first objective).it can not be dominated by other solutions because its value for first objective function is greater than all solutions except first. Similarly for the third solution we need at most two comparisons from fist and second point. And for the last point of list we need to compare this solution to all non dominated solutions. If solutions in list are not unique in first objective function value, then we have to make certain modification in proposed algorithm. Like we have to check every solution to its immediate successors, if any immediate solution dominates this solution then we have to remove this point from the non dominated set S1.Finally we display the non dominated solutions.

3.1Step by step procedure for the proposed algorithm

The proposed algorithm can be executed using the following steps.

1. Sort all the solutions ($P_1 \ldots P_N$) in decreasing order of their first objective function (F1) and create a sorted list (O)
2. Initialize a set S_1 and add first element of list O to S_1
3. For every solution O_i (other than first solution) of list O, compare solution Oi from the solutions of S_1
   i. If any element of set S_1 dominate O_i, Delete O_i from the list
   ii. If O_i dominate any solution of the set S_1, Delete that solution from S_1
   iii. If O_i is non dominated to set S_1, Then update set $S_1 = S_1 \cup O_i$
   iv. If set S1 becomes empty add immediate solution at immediate solution to S1
4. Print non dominated set S_1

3.2 How Proposed Algorithm is Different from Kung's Algorithm:

In First look the working of proposed algorithm resembles with Classic Kung's algorithm but it is quite different from Kung's algorithm. This difference is due to deleting procedure of both the algorithm. In Kung's algorithm a recursive approach was used to delete the dominated points from the set, this approach gives no weight age to the knowledge that is earned by sorting the solutions according to their first objective. Let us make it clear, when we sort the solutions according to their first objective. The solutions which have a good probability to dominate other solutions will come on the beginning of the list. This can be understand by the definition of dominated point, If a point dominate other point it has to be good in all objectives so it

will be defiantly good in first objective. That is what we are doing when we apply sorting to the list. So it would be better if we start deleting points from the front end of the sorted list. In this way, we will not waste our time in making unnecessary comparisons as was done in Kung's algorithm. In our algorithm we give proper weight age to the knowledge earned by sorting and our deleting procedure start with the front end of the list. Let as take a example that will differentiate between our algorithm and Kung's algorithm. Let we have some solutions in which only one solution dominate all other solutions. Since this solution is better than all solutions when we apply sorting it will come at the starting of the list. When we apply both the algorithms Kung's and Proposed algorithm on the sorted list of this problem, then for this case time complexity of our algorithm will be O(nlongn) where as The complexity of Kung's Algorithm will be $O(n^2)$.So our algorithm will always work faster than Kung's algorithm.

3.3Detail of Algorithm:

To make the algorithm clear, consider the example taken from Jun Du's paper, we illustrate the working of above steps on this example. First, let's take out an MOO example with 10 solutions and 4 objectives. The following Table 1 presents the 4 objective function values (O1-O4) for 10 solutions (P1-P10).

Table 1: Objective Function Values

Obj. Func.	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
O1	0.94	0.35	0.76	0.88	0.39	0.86	0.27	0.91	0.73	0.53
O2	2934	3599	2780	1998	3476	3331	2597	2318	3273	4055
O3	5.3	6.6	5.4	8.0	8.7	7.9	9.1	2.1	4.9	7.7
O4	289	45	23	598	444	99	188	239	177	328

According to the step 1 of the algorithm, we first sort solutions in descending order of their first objective function value therefore sorted list will contain {P1, P8, P4, P6, P3, P9, P10, P5, P2, P7}. We take first solution of this list and add this solution to set S1.Now, we initialize a set S1 and put the first element P1 of the list to the set S1. Next we compare second solution P8 from set S1, as P1 dominate P8 so discard this solution and go for third solution P4. As the set S1 contains only one solution P1, so we need only one comparison to decide the category of P4. As P1 is non dominated to P4, so we update the initial set and now set S1 contains two solutions (P1, P4). We have to make two comparisons for P6, this solution is non dominated to both the solutions so we update the set S1 to (P1, P4, P6). For P3, we check its dominance with the set (P1, P4, P6), it is dominated by P6, so we discard this point. Similarly for other solutions we can repeat the same process

4.EXPERIMENTAL RESULTS AND COMPLEXITY ANALYSIS:

To find a non-dominated set of a multi objective optimization problem, Comparison of Jun du's algorithm and proposed algorithm are performed on a computer with Intel **core™2** 1.60G Hz CPU and 1GB memory. Running time of the algorithms is taken as the criterion to evaluate the efficiency. Various objectives number and solutions number are set for test.

The objective function values are generated randomly. To remove experimental error, comparisons are performed more than once. Tables 1 and 2 contain the experiments results, where M is objective number, N is population size and Q is the size of non dominated set. From the table, it is clear that our algorithm is more efficient than Jun du,s algorithm.

Table1: Running time analysis of Jun du and proposed algorithm for Four Objective Functions

M (No. of Objective Functions)	N (Population Size)	Running Time		Q (No. of Non dominated Solutions)
		Jun du's Algorithm	Proposed Algorithm	
4	5000	885.4940	204.3920	737
		871.2670	183.0970	675
		443.1880	97.6710	737
		1463.5	317.5350	956

4	10000	1313.2	305.8380	916
		1353.1	308.8800	918
4	20000	3543.3	1010.5	1304
		3674.7	1017.7	1317
		3724.5	1026.2	1354
4	50000	9029.1	5583.5	1872

Table 2: **Running time analysis of Jun du and proposed algorithm for Seven Objective Functions**

M (No. of Objective Functions)	N (Population Size)	Running Time		Q (No. of Non dominated Solutions)
		Jun du's Algorithm	Proposed Algorithm	
7	5000	1035.5	223.5940	4263
		1011.4	209.0090	675
		1027.6	209.9300	697
7	10000	3427.7	1001.6	918
		3546.3	1010.7	917
		3619.1	1026.2	926

Table3**: Running time analysis of Jun du and proposed algorithm for Ten Objective Functions**

M (No. of Objective Functions)	N (Population Size)	Running Time		Q (No. of Non dominated Solutions)
		Jun du's Algorithm	Proposed Algorithm	
10	5000	813.9270	185.3600	1118
		792.1040	198.5720	1106
		811.4530	187.4750	1126
10	10000	9167.1	2667.9	1567
		9475.7	2654.1	1585
		20497.0	4664.3	2131

4.1Complexity analysis

(i) The best case complexity of the proposed algorithm will be O(NLogN) this can be calculated as follows.

In step 1 the time taken by quick sort will be NLogN per List. For the best case the number of solutions in set S1 will be one, so in total N(one for each solution) comparisons will be required to find the non dominated set. This condition occurs when all non dominated fronts contain only one solution.

(ii) The worst case complexity of the proposed algorithm will be O($M(N)^2$**) this can be calculated as follows.**

In step 1 the time taken by quick sort will be NLogN per List. For this case there will be no dominated solution in the list. So to check non dominance of a solution, all solutions of S1 is to be checked.

5. CONCLUSION:

The algorithm proposed in this paper could find non-dominated set efficiently by two steps: sorting step and deleting step. The time complexity analysis shows that this algorithm is better than any other algorithm in its best case analysis. Also in average case its complexity is same as of Kung's algorithm which has better complexity in comparison of other traditional algorithm.The idea can be extended to provide non dominated sorting of the population.

REFERENCES:

[1]. Jun Du, Zhihua Cai and Yunliang Chen," A Sorting Based Algorithm for Finding Non-Dominated Set in Multi-Objective Optimization" Third International Conference on Natural Computation (ICNC 2007)

[2] K. Deb, "Multi-objective Optimization Using Evolutionary Algorithms," JOHN WILEY&SONS, LTD, 2001 pp.33-43, 2000.

[3]. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. Afast elitist non-dominated sorting genetic algorithm for multi-objectv e optimization: NSGA-II. In *Proceeding of the 6th International Conference on Parallel Problem Solving from Nature,* Pages 849-858, 2000.

[4]. L. Ding. S.Zeng. and L. Kang, "A fast algorithm on finding the non-dominated set in multi-objectve optimization.", *In Proceedings of International Conference on Evolutionary Computation*, pages 2565-2571, 2003

[5] A. Freitas," A critical review of multi-objective optimization in data mining: a position paper. ",SIGKDD Explorations, 6(2): 77-86, 2004.

[6] J. Knowles and D. Corne,"Reducing local optima in single-objective problems by multiobjectivization", In *Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization*, pages 269-283, 2001.

[7] H. Kung. F. Luccio, and F. Preparata,"On finding the maxima of a set of vectors.", *Journal of the Association Computing Machinery,* 22(4) : 469-476, 1975.

[8] E. Zitzler, M. Laumanns, and L. Thiele,"Spea2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization.", In *Proceedings of the Evolutionary Methods for Design, Optimization, and Control,* 19-26, Barcelona, Spain, 2002.

[9]. C. Fonseca, M. and P. J. Fleming " Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," In S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, pp. 416C423. Morgan Kaufmann. , 2003

[10]. J. Horn, and N. Nafpliotis " Multiobjective optimization using the niched Pareto genetic algorithm," IlliGAL Report 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, Champaign.,1993

[11]. J. Horn, N. Nafpliotis, and D. E. Goldberg" A niched Pareto genetic algorithm for multiobjective optimization," In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation, Volume 1, Piscataway, NJ, pp. 82C87. IEEE. ,1994

[12]. R. C. Purshouse, P. J. Fleming " The Multi-objective Genetic Algorithm Applied to Benchmark Problems–an Analysis," Research Report No. 796. Department of Automatic Control and Systems Engineering University of Sheffield, Sheffield, S1 3JD, UK. 2001

[13]. J. D. Schaffer "Multiple objective optimization with vector evaluated genetic algorithms," In J. J. Grefenstette (Ed.),Proceedings of an International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA, pp. 93C100. sponsored by Texas Instruments and U.S. Navy Center for Applied Research in Artificial Intelligence (NCARAI), 1985

[14]. N. Srinivas and K. Deb , " Multiobjective optimization using non-dominated sorting in genetic algorithms," Evolutionary Computation 2(3), 221C248, 1985

[15]. A. Tiwari and R. Roy(2002) " Generalised Regression GA for Handling Inseparable Function Interaction: Algorithm and Applications," Proceedings of the seventh international conference on parallel problem solving from nature. (PPSN VII). Granada, Spain

[16]. E. Zitzler" Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph. D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Aachen, Germany, 1999

[17]. E. Zitzler, M. Laumanns and L. Thiele," SPEA2: Improving the Strength Pareto Evolutionary Algorithm," TIK-Report 103. ETH Zentrum, Gloriastrasse 35, CH-8092 Zurich, Switzerland. 1999.