

# Running Time Analysis of a Multiobjective Evolutionary Algorithm on Simple and Hard Problems

Rajeev Kumar and Nilanjan Banerjee

Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur  
Kharagpur, WB 721 302, India

rkumar@cse.iitkgp.ernet.in, nilanb@cs.umass.edu

**Abstract.** In this paper, we suggest a multiobjective evolutionary algorithm based on a restricted mating pool (REMO) with a separate archive for storing the remaining population. Such archive based algorithms have been used for solving real-world applications, however, no theoretical results are available. In this paper, we present a rigorous running time complexity analysis for the algorithm on two simple discrete pseudo boolean functions and on the multiobjective knapsack problem which is known to be NP-complete. We use two well known simple functions LOTZ (Leading Zeros: Trailing Ones) and a quadratic function. For the knapsack problem we formalize a  $(1 + \epsilon)$ -approximation set under a constraint on the weights of the items. We then generalize the idea by eliminating the constraints based on a principle of partitioning the items into blocks and analyze REMO on it. We use a simple strategy based on partitioning of the decision space into fitness layers for the analysis.

## 1 Introduction

Evolutionary algorithms are emerging as a powerful tool to solve NP-hard combinatorial optimization problems. EAs use a randomized search technique with a *population* of individuals. The genetic operators used by EAs do not apply, in general, any problem-specific knowledge, however, special genetic operators may be designed by incorporating domain knowledge to expedite the search for some applications. In the multiobjective scenario, EAs often find effectively a set of diverse and mutually competitive solutions. Some results for solving computationally hard problems [1, 2] using multiobjective EAs are available in the literature – e.g.,  $m$ -dimensional knapsack [3], minimum spanning tree [4, 5], partitioning of high-dimensional patterns spaces [6], code-book design [7], communication network topology design [8], and network design [9].

The EA operators like mutation and crossover imitate the process of natural evolution. The underlying principles of the operators are simple, but nevertheless, EAs exhibit complex behavior which is difficult to analyze theoretically. Hence, though there are numerous empirical reports on the application of EAs, work on their theoretical analysis is rare. However, besides empirical findings, theoretical analysis is essential to understand the performance and behavior of such heuristics. Some work, in this direction, has recently started, e.g., [10–12].

In case of single objective optimization, many results have been obtained on the analysis of evolutionary algorithms. Results on the time bounds of algorithms in the

discrete search space [13] as well as continuous search space [14] are available. Some analysis on special functions using  $(1 + 1)$  EA has been done - linear functions [15], ONE-MAX function [16], unimodal function [13, 17], and pseudo-boolean quadratic functions [18]. Most of the work above analyzed evolutionary algorithms with mutation as the only genetic operator. However, a proof that crossover is essential is presented in [19].

The analysis of the multiobjective case, however, is more difficult than its single objective counterpart since it involves issues like the size of Pareto-set, diversity of the obtained solutions and convergence to the Pareto-front [20, 21]. Consequently, results on theoretical analysis of multiobjective evolutionary algorithms are few. Rudolph [22, 23] and Rudolph and Agapie [24] have studied multiobjective optimizers with respect to their limit behavior. Laumanns et al. pioneered in deriving sharp asymptotic bounds for two-objective toy functions [10, 11]. Recently, Giel [25] and Thierens [26] derived bounds for another simple function.

Most of the work done earlier deals with analysis of simple problems. However, analysis of a multiobjective evolutionary algorithm on a simple variant of the 0-1 knapsack problem was started by Laumanns et al. [12]. They analyzed the expected running time of two multiobjective evolutionary algorithms ‘Simple Evolutionary Multiobjective Optimizer (SEMO)’ and ‘Fair Evolutionary Multiobjective Optimizer (FEMO)’ for a *simple* instance of the multiobjective 0-1 knapsack problem. The considered problem instance has two profit values per item and cannot be solved by one-bit mutations. In the analysis, the authors make use of two general upper bound techniques, the decision space partition method and the graph search method. The paper demonstrates how these methods, which have previously only been applied to algorithms with one-bit mutations, are equally applicable for mutation operators where each bit is flipped independently with a certain probability. However, the work takes care of only a very special instance of the knapsack and cannot be extended to any knapsack problem.

None of the work involves analysis of real-world combinatorial optimization problems using multiobjective EAs. In this work, we continue such an analysis for the well-known bi-objective 0–1 knapsack problem [27–30]. In the most general case, the Pareto-optimal set for the knapsack can be exponentially large in the input size. Therefore, we first, formulate a  $(1 + \epsilon)$ -approximate set for the 0 - 1 knapsack, followed by a rigorous analysis of the expected time to find the solution-set. We also carry out the expected running time analysis on two simple pseudo-boolean functions, namely the Leading Zeros: Trailing Ones (LOTZ) [10] and Quadratic Function (QF) [25].

We suggest a simple multiobjective optimizer based on an archiving strategy which is well adapted to work efficiently for problems where the Pareto-optimal points are Hamming neighbors (i.e., having a Hamming distance of 1). We call our algorithm Restricted Evolutionary Multiobjective Optimizer (REMO). The algorithm uses a special archive of two individuals which are selected based on a special fitness function which selects two individuals with the largest Hamming distance. Such a mechanism assures that the individuals selected for mutation are more likely to produce new individuals. However, this assumption holds for functions where the optimal set consists of individuals which are Hamming neighbors of each other and the distribution of the optimal points is fairly uniform.

The rest of the paper is organized as follows. Section 2 describes the related work in the field of theoretical analysis of evolutionary algorithms. Section 3 includes a few definitions pertaining to multiobjective optimization. Section 4 describes our algorithm REMO. Section 5 presents the analysis of the REMO on the LOTZ and the Quadratic function. Section 6 formulates the linear functions, the knapsack problem and its  $(1+\epsilon)$ -approximate set. The analysis of the algorithm on the knapsack problem is given in section 7. Finally, conclusions are drawn in section 8.

## 2 Related Work

### 2.1 Problems Analyzed

The work of Beyer et al. [31] revolves around how long a particular algorithm takes to find the optimal solutions for a given class of functions. The motivation to start such an analysis was to improve the knowledge of the randomized search heuristics on a given class of functions. Rudolph [22, 23] and Rudolph and Agapie [24] studied multi-objective optimizers with respect to their limit behavior. They aimed to find whether a particular algorithm converges if the number of iterations goes to infinity.

In the single objective case, the working of EAs have been analyzed for many analytical functions - linear functions [15]; unimodal functions [13]; and quadratic functions [18]. Some recent work has been done on sorting and shortest-path problems by recasting them as combinatorial problems [32]. A study is done to evaluate the black-box complexity of problems too [33].

All the above work used a base-line simple  $(1+1)$  EA. The analysis of  $(1+1)$  EA was done using the method of partitioning of the decision space in accordance with the complexity of the problem into fitness layers as one of the methods [15]. For all such work, the only genetic operator used was mutation. However, Jansen and Wegener [19] analyzed the effectiveness of crossover operator, and showed that crossover does help for a class of problems.

For multiobjective optimization, the analysis of the asymptotic expected optimization time has been started by Laumanns et al. [10]. They presented the analysis of population-based EAs (SEMO and FEMO) on a bi-objective problem (LOTZ) with conflicting objectives. They extended this work by introducing another pseudo-boolean problem (Count Ones Count Zeros: COCZ), another algorithm Greedy Evolutionary Multiobjective Optimizer (GEMO), and scaling the LOTZ and COCZ problems to larger number of decision variables and of objectives [11]. A similar analysis was performed by Giel [25] and Thierens [26] on another bi-objective problem (Multiobjective Count Ones: MOCO) and the quadratic function that we use for our algorithm. These authors designed simple toy functions to understand the behavior of simple EAs for multiobjective problems. In [12] the authors solve a special instance of the multiobjective knapsack problem.

### 2.2 Algorithms Analyzed

The single objective optimizer basically yields a single optimal solution so  $(1+1)$  EAs have been successfully used and analyzed for different functions. However, in the multiobjective case, an optimizer should return a set of *incomparable or equivalent*

solutions. Hence a population-based EA is preferred. For this purpose, Laumanns et al. proposed a base-line population based EA called Simple Evolutionary Multiobjective Optimizer (SEMO). Another strategy used is a multi-start variant of  $(1 + 1)$  EA [11].

These algorithms have an unbounded population. Individuals are added or removed from the population based on some selection criterion. Laumanns et al. introduced two other variants of SEMO called FEMO (Fair EMO) and GEMO (Greedy EMO) which differ in their selection schemes. The algorithms do not have any defined stopping criterion and are run till the desired representative approximate set of the Pareto-front is in the population [11].

There is another group of algorithms which use an explicit or implicit archiving strategy to store the best individuals obtained so far. This approach has proven to be very effective in finding the optimal Pareto-front at much reduced computational cost, e.g., NSGA-II [34], PAES [35], PCGA [36] and SPEA2 [37]. Also, many real-world problems have effectively been solved using such a strategy. However, there exists no analysis of such algorithms. In this work, we propose and use an archive-based EA.

Another issue in archive-based EAs is the size of the archive and the mating pool. If we restrict the number of individuals used for mating in the population to a constant, the expected waiting time till the desired individual is selected for mutation, is considerably reduced. But, for such an algorithm an efficient selection strategy to choose the proper individuals from the archive to the population needs to be devised. This is further discussed while formulating and analyzing the REMO algorithm

### 3 Basic Definitions

In the multiobjective optimization scenario there are  $m$  incommensurable and often conflicting objectives that need to be optimized simultaneously. We formally define some terms below which are important from the perspective of MOEAs. We follow [3, 23–25, 36, 38] for some of the definitions.

**Definition 1. Multiobjective Optimization Problem (MOP):** *A general Multiobjective Optimization problem includes a set of  $n$  decision variables  $x = (x_1, x_2, \dots, x_n)$ , a set of  $m$  objective functions  $F = \{f_1, f_2, \dots, f_m\}$  and a set of  $k$  constraints  $C = \{c_1, c_2, \dots, c_k\}$ . The objectives and the constraints are functions of the decision variables. The goal is to:*

$$\begin{aligned} &\text{Maximize/Minimize: } F(x) = \{f_1(x), f_2(x), \dots, f_m(x)\} \\ &\text{subject to satisfaction of the constraints:} \\ &C(x) = \{c_1(x), c_2(x), \dots, c_k(x)\} \leq 0 \text{ for} \\ &X = (x_1, x_2, \dots, x_n) \in Y \\ &F = (f_1, f_2, \dots, f_m) \in G \end{aligned}$$

where  $X$  is the decision vector,  $F$  is the objective vector,  $Y$  denotes the decision space and  $G$  is a function space.

The  $m$  objectives may be mutually conflicting in nature. In some formalizations the  $k$  constraints defined above are treated as objective functions, thus, making the problem constraint-free, and vice-versa the objectives may be treated as constraints to reduce the dimensionality of the objective-space.

**Definition 2. Quasi Order, Partial Order:** A binary relation  $\preceq$  on a set  $\mathbf{F}$  is called a quasi order if it is both reflexive and transitive. The pair  $(F, \preceq)$  is called a partially ordered set if it is an antisymmetric quasi order.

The Pareto dominance relationship in multiobjective evolutionary algorithms are partial orders (posets). The reason for the Pareto dominance relation to be a partial order is that there might be a number of individuals in the population which are mutually *incomparable or equivalent* to each other. An ordering is not defined for them.

**Definition 3.  $\preceq_q$ :** Let  $Y$  be the decision space and let  $(G, \preceq)$  be a poset of objective values. Let  $f : Y \rightarrow G$  be a mapping. Then  $f$  induces a quasi-order  $\preceq_q$  on  $Y$  (a set of binary vectors) by the following definitions for a minimization problem:

$$\begin{aligned} x \prec_q y &\text{ iff } f(x) \prec f(y) \\ x =_q y &\text{ iff } f(x) = f(y) \\ x \preceq_q y &\text{ iff } x \prec_q y \vee x =_q y . \end{aligned}$$

The above definition introduces the concept of Pareto-dominance and Pareto optimality. Pareto-dominance can be defined as follows:

In a maximization problem of  $m$  objectives  $o_1, o_2, \dots, o_m$ , an individual objective vector  $F_i$  is partially less than another individual objective vector  $F_j$  (symbolically represented by  $F_i \prec F_j$ ) iff  $(\forall_{o_i})(f_i^{o_i} \leq f_j^{o_i}) \wedge (\exists_{o_j})(f_i^{o_j} < f_j^{o_j})$ , where  $f_i^{o_i}$  and  $f_i^{o_j}$  are components of  $F_i$  and  $F_j$  respectively.

Then  $F_j$  is said to dominate  $F_i$ . If an individual is not dominated by any other individual, it is said to be non-dominated. We use the notion of Pareto-optimality if  $F = (f_1, \dots, f_m)$  is a vector-valued objective function. Pareto dominance is formally defined in the next definition.

**Definition 4. Pareto Dominance:** A vector  $f_m = \{f_1^m, \dots, f_k^m\}$  is said to dominate a vector  $f_n = \{f_1^n, \dots, f_k^n\}$  (denoted by  $f_m \prec f_n$ ) iff  $f_n$  is partially less than  $f_m$ , i.e.,  $\forall i \in \{1, \dots, k\}, f_m^i \leq f_n^i \wedge \exists i \in \{1, \dots, k\} : f_m^i < f_n^i$ .

**Definition 5. Pareto Optimal Set:** A set  $A \subseteq Y$  (where  $Y$  denotes the entire decision space) is called a Pareto optimal set iff

$$\forall a \in A: \text{ there does not exist } x \in Y: a \prec x .$$

In most practical cases it is not possible to generate the entire Pareto-optimal set. This might be the case when the size of the set is exponentially large in the input size. Thus, we confine our goal to attain an approximate set. This approximate set is usually polynomial in size. Since in most cases the objective functions are not bijective, there are a number of individuals in the decision space which are mapped to the same objective function. Hence, one might define an approximate set by selecting only one individual corresponding to an objective function. This is usually done in the case of single objective optimization problem.

**Definition 6. Approximate Set:** A set  $A_p \subseteq A$  (Pareto-optimal Set) is called an approximate set if there is no individual in  $A_p$  which is weakly dominated by any other member of  $A_p$ .

Another strategy that might be used to attain an approximate set is to try and obtain an inferior Pareto front. Such a front may be inferior with respect to the distance from the actual front in the decision space or the objective space. If the front differs from the actual optimal front by a distance of  $\epsilon$  in the objective space, then, the dominance relation is called a  $(1 + \epsilon)$ -dominance.

**Definition 7.  $(1 + \epsilon)$ -Domination:** For decision vectors  $X_1, X_2 \in X$  where  $X_1 = (x_{11}, x_{12}, \dots, x_{1i})$  and  $X_2 = (x_{21}, x_{22}, \dots, x_{2i})$ , we say that  $X_2$   $(1 + \epsilon)$ -dominates  $X_1$ , denoted by  $X_1 \preceq^{1+\epsilon} X_2$ , if  $f(x_{1i}) \leq (1 + \epsilon) \cdot f(x_{2i})$  for all objectives to be maximized, and  $f(x_{2i}) \leq (1 + \epsilon) \cdot f(x_{1i})$  for all objectives to be minimized.

The  $(1 + \epsilon)$ -dominance is transitive. The optimal set created by applying the above dominance relation is called a  $(1 + \epsilon)$ -approximate set.

**Definition 8.  $(1 + \epsilon)$ -Approximate Set:** A set  $A_{1+\epsilon}$  is called a  $(1 + \epsilon)$ -approximate set of the Pareto set if for all elements  $a_p$  in the Pareto-set there exists an element  $a' \in A_{1+\epsilon}$  such that  $a_p \preceq^{1+\epsilon} a'$ .

**Definition 9.  $\delta$ -Sampling of the Pareto-Set:** If  $P$  denotes the Pareto-optimal set, then a  $\delta$ -sampling of  $P$  is a set  $P' \subset P$ , such that no two individuals in  $P'$  is within a distance of  $\delta$  units in the objective space (assuming some metric in the objective space).

One might also attain an approximate set by taking a proper subset of the Pareto-optimal set. A strategy used to get such a subset is called sampling. A  $\delta$ -sampling is a special form of sampling in which no two individuals are within a distance of  $\delta$  in the objective space. The reason is to reduce the output length of the algorithm and reduce the problem to  $P$ -space.

**Definition 10. Running Time of an EA:** The running time of an EA searching for an approximate set is defined as the number of iterations of the EA loop until the population is an approximate set for the considered problem.

## 4 Algorithm

The algorithm suggested in this paper uses a restricted mating pool or population  $P$  of only two individuals and a separate archive  $A$  for storing all other points that are likely to be produced during a run of the algorithm. The two individuals to be chosen are selected based on a special function called **Handler**. With a probability of  $\frac{1}{2}$  a function  $Fit(x, P \cup A) = H(x)$  is evaluated where  $H(x)$  is the number of hamming neighbors of  $x$  in  $P \cup A$ . The two individuals with the smallest  $Fit$ -values are selected into the population  $P$  and the rest are transferred to the archive  $A$ . Such a selection strategy assures that we select an individual for mating that has a higher probability of producing a new offspring in the next run of the algorithm. Such a selection strategy has been found to improve the expected running time for simple functions like LOTZ. The intuition behind such a selection scheme is that for simple functions whose optimal set is uniformly distributed over the front and whose individuals are Hamming neighbors of each other. However, for the other half of the cases the handler function selects the two individuals at random. This is similar to the selection mechanism in SEMO [10]. This

is done because in certain functions it is probable that if the two individuals selected are always those with the largest hamming distance, the algorithm might not be able to explore all the bit vectors in the optimal set. This happens for harder problems like the 0 - 1 knapsack. The algorithm takes  $\epsilon$  ( $\epsilon \geq 0$ ) as an input parameter and produces as its output a  $(1 + \epsilon)$ -approximate set of the Pareto-optimal set. The algorithms in its main loop creates an individual uniformly at random and adds the individual into the population if it is not weakly dominated by any other individual in the population and it is not dominated by any other individual. All those individuals which are dominated by the new individual and if the new individual does not dominate the individual in the  $P$  and  $A$  are removed from  $P$  and  $A$ . Such a strategy is adopted so that if any individual from the  $(1 + \epsilon)$ -set is created it is never removed from the population. Note that if  $\epsilon = 0$  and we aim to find the entire Pareto set then we only need to check whether there is some individual which is dominated by the newly created individual and remove it from  $P$  and  $A$ .

---

### Restricted Evolutionary Multiobjective Optimizer (REMO)

---

1. Input Parameter:  $\epsilon$ ,  $\epsilon \geq 0$ , if we desire a  $(1 + \epsilon)$ -approximate set of the Pareto-optimal set.
2. Initialize two sets  $P = \phi$  and  $A = \phi$ , where  $P$  is the mating pool and  $A$  is an archive.
3. Choose an individual  $x$  uniformly at random from  $Y = \{0, 1\}^n$ .
4.  $P = \{x\}$ .
5. **loop**
6. Select an individual  $y$  from  $P$  at random.
7. Apply mutation operator on  $y$  by flipping a single randomly chosen bit and create  $y'$ .
8.  $P = P \setminus \{l \in P \mid l \prec y' \wedge l \text{ does not } (1 + \epsilon)\text{-dominate } y'\}$ .
9.  $A = A \setminus \{l \in A \mid l \prec y' \wedge l \text{ does not } (1 + \epsilon)\text{-dominate } y'\}$ .
10. **if** there does not exist  $z \in P \cup A$  such that  $y' \prec z$  or  $f(z) = f(y')$  **then**  $P = P \cup \{y'\}$ .
11. **end if.**
12. **if** cardinality of  $P$  is greater than 2 **then**
13. **Handler Function**
14. **end if.**
15. **end loop.**

---

### Handler Function

---

1. Generate a random number  $R$  in the interval  $(0, 1)$ .
  2. **if**  $R > \frac{1}{2}$  **then** Step 3 **else** Step 5.
  3. For all the members of  $P \cup A$  calculate a fitness function  $Fit(x, P \cup A) = H(x)$  where  $H(x)$  denotes the number of hamming neighbors of  $x$  in  $P \cup A$ .
  4. Select two individuals with the minimum  $Fit(x, P \cup A)$  values into  $P$  and put the rest of the individuals in the archive  $A$ . In case of equal  $Fit(x, P \cup A)$  values the selection is made at random.
  5. Select two individuals at random from  $P \cup A$ .
-

## 5 Analysis of REMO on Simple Functions

### 5.1 Leading Ones Trailing Zeros

The Leading Ones (LO), Trailing Zeros (TZ) and the LOTZ problems can be defined as follows where the aim is to maximize both the objectives:

$$\begin{aligned} \text{LO} : \{0, 1\}^n &\rightarrow N & \text{LO}(x) &= \sum_{i=1}^n \prod_{j=1}^i x_j \\ \text{TZ} : \{0, 1\}^n &\rightarrow N & \text{TZ}(x) &= \sum_{i=1}^n \prod_{j=i}^n (1 - x_j) \\ \text{LOTZ} : \{0, 1\}^n &\rightarrow N^2 & \text{LOTZ}(x) &= (LO(x), TZ(x)) \end{aligned}$$

**Proposition 1.** The Pareto front (optimal set of points in the objective space) for LOTZ can be represented as a set  $S = \{(i, n - i) \mid 0 \leq i \leq n\}$  and the Pareto set consists of all bit vectors belonging to the set  $P = \{1^i 0^{n-i} \mid 0 \leq i \leq n\}$  [10].

**Proof.** The proof is the same as given in [10].

**Analysis:** The analysis of the function above is divided into two distinct phases. Phase 1 ends with the first Pareto-optimal point in the population  $P$ , and Phase 2 ends with the entire Pareto-optimal set in  $P \cup A$ . We assume  $\epsilon = 0$ , thus, we aim to find the entire Pareto-optimal set.

**Theorem 2.** The expected running time of REMO on LOTZ is  $O(n^2)$  with a probability of  $1 - e^{-\Omega(n)}$ .

**Proof.** We partition the decision space into fitness layers defined as  $(i, j)$ , ( $0 \leq i, j \leq n$ ) where  $i$  refers to the number of Leading-ones and  $j$  is the number of Trailing-zeros in a chromosome. The individuals in one particular fitness layer are incomparable to each other. A parent is considered to climb up a fitness layer if it produces a child which dominates it. In Phase 1 a mutation event is considered a success if we climb up a fitness layer. If the probability of a success  $S$ , denoted by  $P(S) \geq p_i$  then the expected waiting time for a success  $E(S) \leq \frac{1}{p_i}$ .

For LOTZ, in Phase 1 the population cannot contain more than one individual for REMO because a single bit flip will create a child that is either dominating or is dominated by its parent and the algorithm does not accept weakly dominated individuals. The decision space is partitioned into fitness layers as defined above. Phase 1 begins with a initial random bit vector in  $P$ . Let us assume that after  $T$  iterations in Phase 1 the individual  $A(i, j)$  is in the population  $P$ . The individual can climb up a fitness layer  $(i, j)$  by a single bit mutation if it produces the child  $(i + 1, j)$  or  $(i, j + 1)$ . The probability of flipping any particular bit in the parent is  $\frac{1}{n}$ , thus the probability associated with such a transition is  $\frac{2}{n}$ . The factor of 2 is multiplied because we could either flip the leftmost 0 or the rightmost 1 for a success. Therefore, the expected waiting time for such a successful bit flip is at most  $\frac{n}{2}$ . If we assume that Phase 1 begins with the worst individual  $(0, 0)$  in the population then algorithm would require at most  $n$  successful mutation steps till the first Pareto-optimal point is found. Thus, it takes  $\sum_{i=1}^{i=n} \frac{n}{2} = \frac{n^2}{2}$  expected number of steps for the completion of Phase 1.

To prove that the above bound holds with an overwhelming probability let us consider that the algorithm is run for  $n^2$  steps. The expected number of successes for these  $n^2$  steps is at least  $2n$ . If  $S$  denotes the number of successes, then by Chernoff's bounds:

$$P[S \leq (1 - \frac{1}{2}) \cdot 2n] = P[S \leq n] \leq e^{-\frac{n}{2}} = e^{-\Omega(n)}$$

Hence, the above bound for Phase 1 holds with a probability of  $1 - e^{-\Omega(n)}$  which is exponentially close to 1.

Phase 2 begins with an individual of the form  $I = (i, n-i)$  in  $P$ . A success in Phase 2 is defined as the production of another Pareto-optimal individual. The first successful mutation in Phase 2 leads to production of the individual  $I_{+1} = (i+1, n-i-1)$  or  $I_{-1} = (i-1, n-i+1)$  in the population  $P$ . The probability of such a step is given by  $\frac{2}{n}$ . Thus, the waiting time till the first success occurs is given by  $\frac{n}{2}$ . If we assume that after the first success  $I$  and  $I_{-1}$  are in  $P$  (without loss of generality), then the Pareto-optimal front can be described as two paths from  $1^{i-1}0^{n-i+1}$  to  $0^n$  and  $1^i0^{n-i}$  to  $1^n$ . At any instance of time  $T$ , let the individuals in  $P$  be represented by  $L = (l, n-l)$  and  $K = (k, n-k)$  where  $0 \leq k < l \leq n$ . As the algorithm would have followed the path from  $(i-1, n-i+1)$  to  $(k, n-k)$  and  $(i, n-i)$  to  $(l, n-l)$  to reach to the points  $L$  and  $K$ , it is clear that at time  $T$  all the individuals of the form  $S = (j, n-j)$  with  $l < j < k$  have already been found and form a part of the archive  $A$ . Moreover, the handler function, assures that  $L$  and  $K$  are farthest apart as far as Hamming distance is concerned. At time  $T$  the probability of choosing any one individual for mutation is  $\frac{1}{2}$ . Let us assume, without loss of generality, that the individual selected is  $(k, n-k)$ . The flipping of the left most 0 produces the individual  $K_{+1} = (k+1, n-k-1)$  and the flipping of the rightmost 1 produces the individual  $K_{-1} = (k-1, n-k+1)$ . Since, the algorithm does not accept weakly dominated individuals and  $K_{+1}$  is already in  $A$ , the production of  $K_{-1}$  can only be considered as a success. Thus the probability of producing another Pareto-optimal individual at time  $T$  is  $\frac{1}{2n}$ . Thus, the expected waiting time of producing another Pareto-optimal individual is at most  $2n$ . Since, no solutions on the Pareto-optimal front is revisited in Phase 2, it takes a maximum of  $n+1$  steps for its completion. The special case hold when the individual  $0^n$  or  $1^n$  appears in the population. Such individuals represent the end points of the Pareto-front and their mutation do not produce any individual that can be accepted. Moreover, such individuals always form a part of the population  $P$  since they have to be a part of the pair of individuals which have the maximum Hamming distance. However, if such an individual is a part of  $P$  the probability bound still holds as the probability of choosing the right individual (the individual which is not  $0^n$  or  $1^n$ ) for mutation is still  $\frac{1}{2}$  and the probability of successful mutation is  $\frac{1}{2n}$  and expected running time bound holds. Therefore, REMO takes  $O(n^2)$  for Phase 2.

Now, we consider Phase 2 with  $4n^2$  steps. By arguments similar to Phase 1, it can be shown by Chernoff's bounds that the probability of the number of successes in Phase 2 being less than  $n$ , is  $e^{-\Omega(n)}$ .

Altogether considering both the Phases, REMO takes  $n^2$  steps to find the entire Pareto-optimal set for LOTZ.

For the bound on the expected time we have not assumed anything about the initial population. Thus, we notice that the above bound on the probability holds for the next  $n^2$  steps. Since the lower bound on the probability that the algorithm will find the entire

Pareto set is more than  $\frac{1}{2}$  (in fact exponentially close to 1) the expected number of times the algorithm has to run is bounded by 2.

Combining the results of both the phases 1 and 2 yields the bounds in the theorem.  $\square$

**Comments:** The above bound only considers the number of iterations that the algorithm needs to find the Pareto set. However, if all the evaluations of the loop is considered, the handler function can take a time  $O(n^2)$  to find the pair of individuals with the least *Fit* value and hence, the entire algorithm may take time of  $O(n^4)$ .

### 5.2 Quadratic Function

We use a continuous quadratic function which has been widely used in *empirical* analysis of multiobjective evolutionary optimizers and adapt it to the discrete boolean decision space exactly as done in [25]. The function in the continuous decision space is  $((x - a)^2, (x - b)^2)$  where the aim is the minimization of both the objectives. The function in the discrete boolean decision space is described in the following manner exactly as in [25]:

$$\begin{aligned}
 & QF : \{0, 1\}^n \rightarrow N^2. \\
 & \text{if } \|x\| = \sum_{i=1}^n x_i \\
 & QF : (\|x\| - a)^2, (\|x\| - b)^2
 \end{aligned}$$

The idea is to just test the efficiency of REMO in solving problems like those of QF and in fact the analysis follows a line similar to that of [25].

**Proposition 2.** *Without loss of generality we assume that  $a > b$ . The Pareto-optimal front of QF can be represented by the set  $F = \{(i^2, (i - (b - a))^2) \mid a \leq i \leq b\}$ . The Pareto-optimal points of QF are those bit vectors where  $a \leq \|x\| \leq b$ .*

**Proof.** The proof is the same as given in [25].

**Theorem 2.** *The running time of REMO on QF is  $O(n \log n)$  for any value of  $a$  and  $b$ . (We assume  $\epsilon = 0$ , thus we aim to find the entire Pareto-optimal set.)*

**Proof.** We partition the analysis into two phases. The analysis turns out to be very similar to that done in [25] but it is much simpler due to the local mutation operator used in REMO. Phase 1 ends with the first Pareto-optimal point in  $P$  and the second phase continues till all the Pareto-optimal bit vectors are in  $P \cup A$ .

It can be proven that in Phase 1 there can be a maximum of 1 (similar to [25] individuals in  $P \cup A$ . Thus, the archive  $A$  is empty. This is because a single bit mutation of a parent with  $\|x\| < a$  or  $\|x\| > b$  will produce an individual which is dominated by or dominates its parent. We partition the decision space into sets with individuals having the same number of ones. Let us consider a bit vector represented as  $I_d$  where  $d$  represents the number of ones in the individual. A single bit mutation of  $I_d$  is considered to be a success if the number of ones increases (decreases) when  $d < a$  ( $d > b$ ).

Therefore, a success  $S$  requires the flipping of any one of the  $d$  1-bits ( $n - d$  0-bits) when  $d < a$  ( $d > b$ ). The probability of a successful mutation  $P(S) = \frac{d}{n}(\text{or } \frac{n-d}{n})$ . The expected waiting time of  $S$ , given by  $E(S) \leq \frac{n}{d}(\text{or } \frac{n}{n-d})$ . Hence, the total expected time till the first Pareto optimal point arrives in the population is at most  $\sum_{d=1}^n \frac{n}{d} = nH_n = n \log n + \Theta(2n) = O(n \log n)$  (where  $H_n$  is the  $n^{\text{th}}$  Harmonic number) by the linearity of expectations.

Phase 2 starts with the assumption that  $b - a > 1$  or else there would be no second phase. The number of individuals in the population is bounded by 2. The selection mechanism ensures that they are the bit vectors that are most capable of producing new individuals. The Pareto-front can be visualized as a path of individuals with number of ones varying from  $a$  to  $b$  or  $b$  to  $a$ . Let us represent any individual with  $a < \|x\| < b$  as  $I_k$  where  $k$  represents the number of ones in the bit vector. Such a bit vector can be produced either by an individual with  $k + 1$  ones or  $k - 1$  ones. The associated probability for such a successful mutation is at least  $\frac{k+1}{2n}$  and  $\frac{n-k+1}{2n}$  respectively. Hence, the expected waiting time till the  $I_k^{\text{th}}$  Pareto optimal point is in the population (assuming that its parent is in the population) is  $E(I_k) \leq \frac{2n}{k+1}$  and  $\frac{2n}{n-k+1}$  for the two cases above. Thus, the total expected time till all the Pareto points are in  $P \cup A$  is at most  $\sum_{k=a}^b E(I_k) \leq \sum_{k=a}^b \frac{2n}{k+1} \leq \sum_{k=0}^{b-a} \frac{2n}{k+1} = 2nH_{b-a}$  where  $H_n$  stands for the  $n^{\text{th}}$  harmonic number.

Therefore, the expected time for Phase 2 is at most  $2ne \log(b - a) + \theta(2ne) = O(n \log(b - a))$ .

Altogether both Phases take a total of  $O(n \log n + n \log(b - a))$  time. Since  $a$  and  $b$  can have a maximum value of  $n$  the running time for REMO on QF is  $O(n \log n)$  which is the same as proven in [25]. □

## 6 Linear Functions and Knapsack Problem

### 6.1 Linear Functions

A bi-objective linear function is:

$$F(x) = (f_1(x) = \sum_{i=1}^n w_i x_i, f_2(x) = \sum_{i=1}^n w'_i x_i)$$

where  $w_i > 0, w'_i > 0$

The aim of a bi-objective problem may be to maximize or minimize both the objectives; in such a case the problem reduces to a single objective one. In our study, therefore, we take the case of simultaneously maximizing  $f_1$  and minimizing  $f_2$ . Thus, the problem is formulated with two mutually conflicting objectives.

In this section, we show that for the function  $F(x)$  the number of Pareto-optimal points can range from  $n + 1$  to  $2^n$ . Thus, for any arbitrary values of the weights  $w$  and  $w'$  the Pareto-optimal set can be exponential in  $n$ . Throughout this section we investigate the case where the bits of the individuals are arranged in their decreasing value of  $\frac{w}{w'}$ . Thus,  $\frac{w_1}{w'_1} \geq \frac{w_2}{w'_2} \geq \dots \geq \frac{w_n}{w'_n}$ .

**Lemma 1.** For any bi-objective linear function  $F(x) = (f_1, f_2)$  the set  $A_1 = \{1^i 0^{n-i}\}$  where  $0 \leq i \leq n$  represents a set of Pareto-optimal points.

**Proof.** Let us consider an individual  $K^*$  in the decision space (which does not belong to  $A_1$ ) and an individual  $K \in A_1$  which is of the form  $(1^k 0^{n-k})$  for  $0 \leq k \leq n$ . If the set of 1-bits of  $K^*$  is a subset of the set of 1-bits in  $K$  it is clear that  $K$  and  $K^*$  are incomparable.

However, if the set of 1-bits of  $K^*$  is not a subset of  $K$ , let  $S$  denote the set of common bit positions that are set to one in both  $K^*$  and  $K$  with  $x_1 = \sum_{i \in S} w_i$ . Let  $S_1$  denote the set of bit positions that are set to one in  $K$  but not in  $K^*$  and  $y_1 = \sum_{i \in S_1} w'_i$ . If the individual  $K^*$  has to dominate  $K$ , in the best case  $f_2(K^*)$  is at most  $f_2(K)$ . Since all the bits are arranged in the decreasing order of  $\frac{w_i}{w'_i}$ ,  $f_1(K^*)$  is at most  $x_1 + \frac{w_{k+1}}{w'_{k+1}} y_1 \leq x_1 + \frac{w_k}{w'_k} y_1$ . Now,  $f_1(K)$  is at least  $x_1 + \frac{w_k}{w'_k} y_1$ . Hence,  $f_1(K) > f_1(K^*)$ . Therefore,  $K^*$  cannot dominate  $K$ .

Now we need to prove that two individuals in  $A_1$  are mutually incomparable to each other. Let us consider another individual  $I = 1^i 0^{n-i}$  in  $A_1$  where  $0 \leq i \leq n$ . If  $i < k$ ,  $f_1(I) < f_1(K)$  and  $f_2(I) < f_2(K)$ , implying that  $I$  and  $K$  are incomparable (by definition 2). A similar argument holds for  $i > k$ , hence proving the lemma.  $\square$

**Example 1.** Let us consider a linear function with three weights. The  $w$  of the components are  $W = \{20, 15, 19\}$  and the weights  $W'$  are  $W' = \{8, 7, 12\}$ . Clearly  $\frac{W_1}{W'_1} > \frac{W_2}{W'_2} > \frac{W_3}{W'_3}$ . Let us consider the bit vector 110. This individual has the first and second weights set to 1. Individuals whose bits set to 1 are a subset of the above individuals, for example 100, will have both  $w$  and  $w'$  less than 110, and hence is incomparable to it. Individuals which are not a subset of 110, like for example 101 will be dominated by 110 or is incomparable to it. As an example,  $w(110) = 35$ ,  $w'(110) = 15$  and  $w(101) = 39$ ,  $w'(101) = 20$ , these two individuals are incomparable. This hold for any individual which is not a subset of 110.

**Proposition 3.** The size of the Pareto-optimal set for the most general case of a linear function  $F(x)$  lies between  $n + 1$  and  $2^n$ .

**Proof.** It is clear from lemma 1 that the lower bound on the number of Pareto-optimal individuals for  $F(x)$  is  $n + 1$ . Moreover, the upper bound holds for cases where all the bit vectors are Pareto-optimal. We next show that there are examples which fit into the above bounds.

**Case 1:** Let us consider a linear function such that  $w_1 > w_2 > w_3 > \dots > w_n$  and  $w'_1 < w'_2 < w'_3 < \dots < w'_n$ . Each Pareto-optimal point is of the form  $X = 1^i 0^{n-i}$  where  $0 \leq i \leq n$ . It is clear that individuals of the form  $X$  represent a Pareto-optimal solution because it contains the  $i$  largest weights of  $f_1$  and the  $i$  smallest weights of  $f_2$ . Flipping the left-most 0-bit of  $X$  to 1 or the right-most 1-bit to 0 creates an individual which is incomparable to  $X$ . Moreover, any individual with a 0 followed by a 1 cannot be Pareto-optimal as it can be improved in both objectives by simply swapping the bits. The Pareto-optimal set of such a function thus contains  $n + 1$  individuals.

**Case 2:** For the other extreme case, let us consider a linear function for which  $\frac{w_1}{w'_1} = \frac{w_2}{w'_2} = \frac{w_3}{w'_3} = \dots = \frac{w_n}{w'_n}$  and  $w_1 > w_2 > w_3 > \dots > w_n$ . It is clear that for such a

function all the points in the decision space  $\{0, 1\}^n$  are Pareto-optimal. Thus, the total number of Pareto points for this case is  $2^n$ .  $\square$

Consequently, for any randomized algorithm the expected runtime to find the entire Pareto-optimal set for the above case of bi-objective linear functions is  $\Omega(2^n)$ .

### 6.2 Knapsack Problem

Next, we show that the above problem of the conflicting objectives for a linear function can be interpreted as the 0 - 1 Knapsack problem.

**Definition 11. 0–1 Knapsack Problem:** The *knapsack problem* with  $n$  items is described by the knapsack of size  $b$  and three sets of variables related to the items: decision variables  $x_1, x_2, \dots, x_n$ ; positive weights  $W_1, W_2, \dots, W_n$ ; and profits  $P_1, P_2, \dots, P_n$ ; where, for each  $1 \leq i \leq n$ ,  $x_i$  is either 0 or 1. The  $W_i$  and  $P_i$  represent the weight and profit, as integers, of the  $i^{th}$  item respectively.

The single-objective knapsack problem can be formally stated as:

$$\begin{aligned} &\text{Maximize } P = \sum_{i=1}^n P_i x_i \\ &\text{subject to } \sum_{i=1}^n W_i x_i \leq b, \\ &\text{where } x_i = 0 \text{ or } 1 \end{aligned}$$

In order to recast the above single-objective problem along with one constraint on weights of the items into a bi-objective problem, we use the formulation similar to the linear function described above. Thus, a bi-objective knapsack problem of  $n$  items with two conflicting objectives (maximizing profits and minimizing weights) is formulated as:

$$\text{Maximize } P = \sum_{j=1}^n P_j x_j \quad \text{and} \quad \text{Minimize } W = \sum_{j=1}^n W_j x_j$$

Therefore, if the set of items is denoted by  $I$ , the aim is to find all sets  $I_i \subseteq I$ , such that for each  $I_j$  there is no other set which has a larger profit than profit ( $I_j$ ) for the given weight bound  $W(I_j)$ . Thus, it is equivalent to finding the collections of items with the maximum profit in a knapsack with capacity  $W(I_j)$ .

The 0 – 1 knapsack problem, in the optimization form above is NP-complete.

In the following section we formalize a  $(1 + \epsilon)$ - approximate set for the above knapsack problem. We work under the assumption that the items are arranged in a strictly decreasing value of  $\frac{P}{W}$ .

**Lemma 2.** Let us define a set  $X_j^i$  as the set of  $i$  most efficient items (efficiency defined by  $\frac{P}{W}$ ) among the  $j$  smallest weight items if the sum of the weights of the  $j$  items is less than the  $(j + 1)^{st}$  item. Let  $A_1$  be the set of all such  $X_j^i$  and the following constraint be imposed on weights of the items:  $\forall X_j^i$ , if  $\{I_1^j, \dots, I_i^j\}$  represents the set of items in  $X_j^i$ , then  $W_{I_{i+1}} < \epsilon \cdot \sum_{k=1}^{k=i} W_{I_k}$  for  $i < j$ . Now, if  $A_2$  represents a singleton set,  $\{0^i 10^{n-(i+1)}\}$  where the 1 is set at the position of the lightest item. Then,  $A_1 \cup A_2$  represents a  $(1 + \epsilon)$ -approximation of the Pareto-optimal set for the knapsack problem.

**Example 2.** Let us consider a knapsack with three items. The profits of the items is given by  $P = \{40, 20, 10\}$  and the weights by  $W = \{20, 11, 6\}$ . The knapsack satisfies the constraint on the weights (given in Lemma 2) for  $\epsilon = 0.6$ .  $X_1$  is a trivial case as it contains the lightest item. For  $X_2^i, 1 \leq i \leq 2$ , we consider the second and the third items. Clearly  $W_3 < \epsilon \cdot W_2$  and hence it satisfies the constraint. For  $X_3^i, 1 \leq i \leq 3$  we have all the items. Since,  $W_2 < \epsilon \cdot W_1$  and  $W_3 < \epsilon \cdot (W_1 + W_2)$ , the knapsack weights satisfy the given constraint.

**Proof.** Let  $(P^o, W^o)$  represent any arbitrary Pareto-optimal solution for the knapsack problem. We need to prove, that corresponding to such a solution we can always find a solution  $(P', W')$  in  $A_1 \cup A_2$  such that  $(P^o, W^o) \prec^{1+\epsilon} (P', W')$ . If  $W^o$  is the item with the smallest weight then the element in set  $A_2$  weakly dominates  $(P^o, W^o)$  and hence  $(1 + \epsilon)$ -dominates  $(P^o, W^o)$ . Now, let us consider a more general case. Let  $\pi$  define the permutation of the items which reorders them according to their weights. Corresponding to any  $W^o$ , we aim to find an item  $I_{\pi(j)}$  such that  $W_{\pi(j)} \leq (1 + \epsilon) \cdot W^o < W_{\pi(j+1)}$ . If such an item cannot be found then  $j = n + 1$ . It is clear that for this  $j + 1$ , an  $X_{j+1}^i \in A_1$ , a set of items  $I_1^{j+1}, I_2^{j+1}, \dots, I_i^{j+1}$  ( $i$  can be equal to  $j$ ), can always be found such that  $\sum_{k=1}^{k=i} W_{I_k} \leq (1 + \epsilon) \cdot W^o < \sum_{k=1}^{k=i+1} W_{I_k}$ . We claim that  $W^o \leq \sum_{k=1}^{k=i} W_{I_k} \leq (1 + \epsilon) \cdot W^o$ . This holds because of the constraint on the weights imposed in the lemma. It is clear from the constraint that if  $W_{I_{i+1}} \leq \epsilon \cdot \sum_{k=1}^i W_{I_k}$  and adding of the  $(i + 1)^{st}$  item into the knapsack increases the weight of the knapsack above  $(1 + \epsilon) \cdot W^o$ , the sum of the weights of the items in  $X_{j+1}^i$  is at least  $W^o$ . Since, the sum of the weights in the set  $X_{j+1}^i$  obeys the weight bound  $(1 + \epsilon) \cdot W^o$ , with respect to the weights,  $(P^o, W^o) \preceq^{1+\epsilon} (P', W')$ . Now, we know that  $W' \geq W^o$ . Let  $S_c$  denote the items common in both the solution  $(P', W')$  and  $(P^o, W^o)$  and  $S_{un}$  denote the set of items in the solution  $(P', W')$  and not in  $(P^o, W^o)$ . Let  $A = \sum_{I_m \in S_c} P_{I_m}$  and  $B = \sum_{I_m \in S_{uc}} W_{I_m}$ . Now,  $W^o < W_{\pi(j+1)}$ , therefore  $(P^o, W^o)$

contains items in the set  $X_{j+1}$  (which is the set of the first  $j + 1$  lightest items). Since  $X_{j+1}^i$  is the set of the  $i$  most efficient items in  $X_{j+1}$  (efficiency defined as  $\frac{P}{W}$ ),  $P^o \leq A + B \cdot \frac{P_{i+1}}{W_{i+1}} \leq A + B \cdot \frac{P_i}{W_i} \leq P'$ , thus proving that  $(P^o, W^o) \preceq^{1+\epsilon} (P', W')$ , hence the lemma.  $\square$

In the next lemma, we extend the results obtained in lemma 2 to formalize a  $(1 + \epsilon)$ -approximate set for any knapsack.

**Lemma 3.** Let  $I = \{I_1, I_2, \dots, I_n\}$  represent the set of items of the knapsack. We partition the set  $I$  into  $m$  blocks ( $1 \leq m \leq n$ ),  $B_1, B_2, \dots, B_m$  which satisfy the following conditions:

1. Each block consists of a set of items and all pairs of blocks are mutually disjoint. However,  $B_1 \cup B_2 \dots \cup B_m = I$ .
2. The items in each block satisfy the weight constraint described in lemma 3 and are arranged in their strictly decreasing  $\frac{P_i}{W_i}$  ratio in the block.

Sets  $A_1$  and  $A_2$ , similar to those defined in lemma 2 are defined for every block  $B_m$ . Let  $A^m = A_1^m \cup A_2^m$  for the  $m^{th}$  block. If  $S$  denotes the set of items formed by taking

one set from every  $A^m$ , then, the collection of all such  $S$  sets, represented by  $S_{comp}$ , denotes a  $(1 + \epsilon)$ -approximation of the Pareto-optimal set for any knapsack problem. If  $m = n$  the set reduces to a power set of the item-set.

**Example 3.** Let us consider a knapsack of four items,  $\{I_1, I_2, I_3, I_4\}$ . The profits of the items are given by  $P = \{40, 20, 10, 15\}$  and the weights are given by  $W = \{20, 11, 6, 40\}$ . The items of the above knapsack can be divided into blocks  $B_1 = \{I_1, I_2, I_3\}$  and  $B_2 = \{I_4\}$ . The block  $B_1$  satisfies the constraint as it is the same set of items described in example 2 and block  $B_2$  satisfies the constraint trivially as it has only one item. Hence, the blocks describe a valid partitioning.

**Proof.** Let us consider any Pareto-optimal collection of items  $U$  with objective values  $(P^o, W^o)$ . We aim to prove that corresponding to every  $U$  we can find a solution  $V$  of the form  $S$  (defined above) which  $(1 + \epsilon)$ -dominates  $U$ .

We partition both solutions  $U$  and  $V$  into blocks of items as defined above. Let us consider a particular block of items  $B_i$ , and denote the set of items in  $U$  and  $V$  in the block as  $B_i^U$  and  $B_i^V$  respectively. Since, the block consists of items which satisfy all the conditions of lemma 3, the solution represented by  $B_i^U$   $(1 + \epsilon)$ -dominates  $B_i^V$  (irrespective of whether  $B_i^U$  represents a Pareto-optimal solution for the items in  $B_i$ ). It is clear that the above argument holds for every block. Now,  $\text{weight}(U) = \sum_{i=1}^m W(B_i)$  and  $\text{profit}(U) = \sum_{i=1}^m P(B_i)$ . Since, for every block  $B_i$ ,  $\text{weight}(B_i(V)) < (1 + \epsilon) \cdot \text{weight}(B_i(U))$  and  $(1 + \epsilon) \cdot \text{profit}(B_i(V)) > \text{profit}(B_i(U))$ ,  $\text{weight}(V) < (1 + \epsilon) \cdot \text{weight}(U)$  and  $(1 + \epsilon) \cdot \text{profit}(V) > \text{profit}(U)$  by a simple summation of the profits and weights of items in every block. Therefore,  $U \preceq^{1+\epsilon} V$ , proving the lemma.  $\square$

**Lemma 4.** The total number of individuals in the  $(1 + \epsilon)$ -approximate set (defined in Lemma 3) is upper bounded by  $(\frac{n}{m})^{2m}$ , where  $m$  is the number of blocks ( $m$  defined in lemma 3) and  $n$  is the number of items in the knapsack, if  $n \neq m$ .

**Proof.** Let the number of individuals in the  $k^{\text{th}}$  block be  $n_k$ . The number of sets of the form  $X_j^i$  in the  $k^{\text{th}}$  block is of the order  $O(n_k^2)$ . Thus, the total number of sets possible for all the blocks is upper bounded by  $(n_1 \cdot n_2 \cdot \dots \cdot n_m)^2$  if  $m$  is not a constant.

Now,  $(n_1 \cdot n_2 \cdot \dots \cdot n_m)^{\frac{1}{m}} \leq \frac{\sum_{i=1}^m n_i}{m}$  (Arithmetic mean  $\geq$  Geometric Mean). Thus,  $(n_1 \cdot n_2 \cdot \dots \cdot n_m)^2 \leq (\frac{\sum_{i=1}^m n_i}{m})^{2m} = O((\frac{n}{m})^{2m})$ , if  $m \neq n$ . However, if  $n = m$ , by the partitioning described in Lemma 3, all the bit vectors represent the  $(1 + \epsilon)$ -approximate set for the knapsack. Hence, in such a case the total number of individuals in the set is  $2^n$ .  $\square$

## 7 Analysis of REMO on Knapsack Problem

**Lemma 5.** If  $P_{knapsack}$  is the sum of all the profits of the items in the knapsack, then the total number of individuals in the population and archive is at most  $P_{knapsack} + 1$ .

**Proof.** It is clear that at any time the population and archive consists of individuals which are incomparable to each other as in the case of SEMO. We aim to prove that any two individuals of the population and archive will have distinct profit values. We try to prove the claim by contradiction. Let us assume that there are two individuals  $a$  and  $b$  in  $P \cup A$  which have the same profit values. As the algorithm does not accept any weakly dominated individuals either  $\text{weight}(a) > \text{weight}(b)$  or  $\text{weight}(a) < \text{weight}(b)$ . However, this contradicts our initial assumption that the population consists of individuals which are incomparable to each other. Hence,  $a$  and  $b$  have distinct profit values. As all the items have integer profits and the total profit can be zero, the total number of individuals are bounded by the sum of the profits  $P_{knap} + 1$ .  $\square$

**Theorem 1.** The expected running time for REMO to find a  $(1 + \epsilon)$ -approximate set of the Pareto-front (formalization given in lemma 3) for any instance of the knapsack problem with  $n$  items is  $O(\frac{n^{2m+1} P_{knap}}{m^{2m+1}})$  where  $m$  is the number of blocks into which the items can be divided (blocks are defined in lemma 3) and  $P_{knap}$  refers to the sum of the profits of the items in the knapsack and  $n \neq m$ . Moreover, the above bound holds with an overwhelming probability. It is worth noting that the expected waiting time is a polynomial in  $n$  if the sum of the profits and the number of partitions  $m$  is a polynomial in  $n$ , which in turn depends on the  $\epsilon$  value. It is difficult to derive any general relationship between  $\epsilon$  and  $m$ , since it depends on the distribution of the weights. However, if the  $\epsilon$  is large then the number of items in a particular block is likely to be large and hence the number of blocks will be small and the running time can be polynomial in  $n$ .

**Proof.** We divide the analysis into two phases. The first phase continues till  $0^n$  is in the population or the archive. Phase 2 continues till all the vectors in set  $S$  is in  $P \cup A$ . (We take  $\epsilon$  as an input.) In the first phase, the aim is to have a all 0s string in the population. We partition the decision space into fitness layers. A fitness layer  $i$  is defined as a solution which has the  $i$  smallest weight items in the knapsack. At any point of time in phase 1, there is a maximum of one solution  $Z$  which has the  $i$  smallest weights in the knapsack. Removal of any one of these items from  $Z$  reduces the weight of the knapsack and hence produces a solution which is accepted. With a probability of  $\frac{1}{2}$  (as in the handler function), an individual is selected from the archive at random. Therefore, the probability of selection of  $Z$  for mutation is  $\frac{1}{4(P_{knap}+1)}$  ( $P_{knap} + 1$ ) is the bound on the population size, by lemma 5). If we flip the 1-bit corresponding to the heaviest item in  $Z$  (which occurs with a probability of  $\frac{1}{n}$ ), the mutated individual is accepted. Thus, the probability of producing an accepted individual is  $\frac{1}{4n(P_{knap}+1)}$ . Therefore, the expected waiting time till a successful mutation occurs is at most  $4n(P_{knap} + 1)$ . Since, a maximum of  $n$  successes in Phase 1 assures that  $0^n$  is in the population, the expected time for completion of Phase 1 is  $4n^2(P_{knap} + 1)$ . Therefore, Phase 1 ends in  $O(n^2 P_{knap})$  time. If we consider  $8n^2(P_{knap} + 1)$  steps the expected number of successes is at least  $2n$ . By Chernoff's bound the probability of number of successes being less than  $n$  is at most  $e^{-\Omega(n)}$ .

The second phase starts with the individual  $0^n$  in the population. An individual that is required to be produced can be described as a collection of items  $I_{coll} = C^1 \cup C^2 \cup$

...  $\cup C^m$ , where,  $C^k$  is either  $X_j^i$  or one item (of the smallest weight) in the  $k^{th}$  block. If  $X_j^{i^k}$  refers to the set  $X_j^i$  in the  $k^{th}$  block, it is clear to see that  $H(0^n, X_j^{1^k}) = 1$ , where  $H$  refers to the Hamming distance. Thus, by a single bit flip of  $0^n$  we can produce a new desired point. Since, the algorithm accepts bit vectors which are  $(1 + \epsilon)$ -approximations of the Pareto-optimal points, the point generated will be taken into the population and will never be removed. It is also clear that  $H(X_j^{i-1^k}, X_j^{i^k}) = 1$ . Hence, corresponding to every bit vector  $R$  which belongs to the  $(1 + \epsilon)$ -approximate set of the Pareto front there is a bit vector in the population or the archive which by a single bit flip can produce  $R$ . Thus, from individuals like  $I_{coll}$ , we can produce another desired individual (which belongs to the  $(1+\epsilon)$ -approximate set) by a single bit flip. With a probability of  $\frac{1}{2}$  (Handler function), two individuals in the population are chosen at random. Therefore, the probability of the individual like  $I_{coll}$  being chosen into the population is  $\frac{1}{2(P_{knap}+1)}$  ( $P_{knap} + 1$  is the bound on the population size, proven in lemma 5). The probability of  $I_{coll}$  being chosen for mutation is thus  $\frac{1}{4(P_{knap}+1)}$ . Flipping a desired 1-bit or 0-bit in any of the  $m$  blocks of  $I_{coll}$  will produce another desired individual. Thus, the probability that  $I_{coll}$  will produce another desired individual is  $\frac{m}{4n(P_{knap}+1)}$ . The expected number of waiting steps for a successful mutation is thus, at most  $\frac{4n(P_{knap}+1)}{m}$ . If  $R'$  is the new individual produced, since, no items have equal  $\frac{P_i}{W_i}$ , there cannot be any individual in the population or the archive which weakly dominates  $R'$ . Hence,  $R'$  will always be accepted. As every individual in the  $(1 + \epsilon)$ -approximate set can be produced from some individual in the population or the archive by a single bit mutation, the total time taken to produce the entire  $(1 + \epsilon)$ -approximate set is upper bounded by  $\frac{4n^{2m+1}(P_{knap}+1)}{m^{2m+1}}$  (total size of the  $(1 + \epsilon)$ -approximate set is bounded by  $O((\frac{n}{m})^{2m})$  by lemma 5).

If we consider a phase of  $\frac{8(P_{knap}+1)n^{2m+1}}{m^{2m+1}}$  steps, then by Chernoff's bounds, the probability of there being less than  $\frac{n^{2m}}{m^{2m}}$  successes is bounded by  $e^{-\Omega(n)}$ . Altogether both the phases take a total of  $O(\frac{P_{knap}n^{2m+1}}{m^{2m+1}})$  for finding the entire  $(1 + \epsilon)$ - approximate set.

For the bound on the expected time we have not assumed anything about the initial population. Thus, we notice that the above bound on the probability holds for the next  $\frac{P_{knap}n^{2m+1}}{m^{2m+1}}$  steps. Since the lower bound on the probability that the algorithm will find the entire  $(1 + \epsilon)$ -approximate set is more than  $\frac{1}{2}$  (in fact exponentially close to 1) the expected number of runs is bounded by 2.

Combining the results of both the phases yields the bound in the theorem. It is worth noting that for cases when the value of the number of partitions into which the set of items have to be partitioned is a *constant* the expected running time is a polynomial in  $n$ . □

**Comments:** An important point to note in the analysis of the algorithm is that for the knapsack problem the selection of the individual in the Handler function is done at random. However, this occurs with a probability of 0.5. It is likely that the specialized *Fit* function is able to find individual which is a subset of the approximate set in time which is faster than the random selection. Note that  $1^i 0^{n-i}$  is a subset of the approximate so-

lution which can be efficiently found by the algorithm as in the case of LOTZ. Thus, though we find a worst case upper bound by considering a total random selection, the actual running time bound may be much better with the *Fit* selection scheme. Moreover, it is also worth noting that the algorithm is adapted to get a  $(1 + \epsilon)$  as well as an optimal set for with  $\epsilon = 0$ .

## 8 Conclusions

In this paper, an archive based multiobjective evolutionary optimizer (REMO) is presented and a rigorous runtime complexity analysis is carried out of the algorithm on simple discrete boolean functions (the LOTZ function [11] and quadratic function [25]) and a NP-Complete problem (0 - 1 knapsack problem). The key feature of REMO is its special restricted population for mating and a separate archive. Such algorithms have been widely used in solving real world applications. The idea is to restrict the mating pool to a constant  $c$ . The value of 2 for  $c$  is sufficient for most simple functions. In case of certain functions a single individual population with a similar selection scheme as REMO may suffice. However, two bit vectors may be required for functions where the Pareto front can be reached via two paths as is the case of the quadratic function. However, for more complicated functions like the knapsack it is better to use a more random selection mechanism since the specialized selection may lead to the algorithm getting trapped in a local optima.

## Acknowledgements

The authors would like to thank Ingo Wegener for valuable discussions during the course of this work. The authors thank Oliver Giel for his invaluable help in proving some lemmas in the knapsack problem. The authors would also like to thank Lothar Schmitt, Kenneth Jong, Alden Wright and anonymous reviewers for suggestions and corrections.

The part of the work was done while Nilanjan Banerjee was at University of Dortmund during summer 2003; his visit was financed by Professor Ingo Wegener's chair and the German Academic Exchange Service (DAAD). Rajeev Kumar acknowledges support from the Ministry of Human Resource Development, Government of India, project during the period of this work.

## References

1. Garey, M.R., Johnson, D.S.: Computers and Interactability: A Guide to the Theory of NP-Completeness. San Francisco, LA: Freeman (1979)
2. Hochbaum, D.: Approximation Algorithms for NP-Hard Problems. Boston, MA: PWS (1997)
3. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evolutionary Computation **3** (1999) 257–271
4. Knowles, J.D., Corne, D.W.: A Comparison of Encodings and Algorithms for Multiobjective Minimum Spanning Tree Problems. In: Proc. Congress on Evolutionary Computation (CEC-01). Volume 1. (2001) 544–551

5. Kumar, R., Singh, P.K., Chakrabarti, P.P.: Improved quality of solutions for multiobjective spanning tree problem using evolutionary algorithm. In: Proc. Int. Conf. High Performance Computing (HiPC-04), LNCS 3296. (2004) 494–503
6. Kumar, R., Rockett, P.I.: Multiobjective genetic algorithm partitioning for hierarchical learning of high-dimensional pattern spaces: A learning-follows-decomposition strategy. *IEEE Trans. Neural Networks* **9** (1998) 822–830
7. Kumar, R.: Codebook design for vector quantization using multiobjective genetic algorithms. In: Proc. PPSN/SAB Workshop on Multiobjective Problem Solving from Nature. (2000)
8. Kumar, R., Parida, P.P., Gupta, M.: Topological design of communication networks using multiobjective genetic optimization. In: Proc. Congress Evolutionary Computation (CEC-2002). (2002) 425–430
9. Kumar, R., Banerjee, N.: Multicriteria network design using evolutionary algorithm. In: Proc. Genetic and Evolutionary Computations Conference (GECCO-03), LNCS 2023. (2003) 2179–2190
10. Laumanns, M., Thiele, L., Zitzler, E., Welzl, E., Deb, K.: Running time analysis of multiobjective evolutionary algorithms on a discrete optimization problem. In: Proc. Parallel Problem Solving from Nature (PPSN VII), LNCS 2439. (2002) 44–53
11. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of evolutionary algorithms on pseudo-boolean functions. *IEEE Trans. Evolutionary Computation* **8** (2004) 170–182
12. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Natural Computing* **3** (2004) 37–51
13. Droste, S., Jansen, T., Wegener, I.: On the Optimization of Unimodal Functions with the (1+1) Evolutionary Algorithm. In: Proc. Parallel Problem Solving from Nature (PPSN-V), LNCS 1498. (1998) 13–22
14. Jagersk pper, J.: Analysis of simple evolutionary algorithm for minimization in euclidean spaces. In: Proc. 30<sup>th</sup> Int. Colloquium Automata, Languages and Programming (ICALP 2003), LNCS 2719. (2003) 1068–1079
15. Droste, S., Jansen, T., Wegener, I.: On the Analysis of the (1+1) Evolutionary Algorithm. *Theoretical Computer Science* **276** (2002) 51–81
16. Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. *Evolutionary Computation* **7** (1999) 167–203
17. Rudolph, G.: How mutation and selection solve long path problems in polynomial expected time. *Evolutionary Computation* **4** (1996) 207–211
18. Wegener, I., Witt, C.: On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions. *J. Discrete Algorithms* (2002)
19. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms: a proof that crossover really can help. *Algorithmica* **34** (2002) 47–66
20. Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: *Evolutionary Algorithms for Solving Multiobjective Problems*. Boston, MA: Kluwer (2002)
21. Deb, K.: *Multiobjective Optimization Using Evolutionary Algorithms*. Chichester, UK: Wiley (2001)
22. Rudolph, G.: *Convergence Properties of Evolutionary Algorithms*. Hamburg, Germany: Verlag Dr. Kova  (1997)
23. Rudolph, G.: Evolutionary search for minimal elements in partially ordered finite sets. In: Proc. Annual Conference on Evolutionary Programming. (1998) 345–353
24. Rudolph, G., Agapie, A.: Convergence properties of some multiobjective evolutionary algorithms. In: Proc. Congress on Evolutionary Computation. (2000) 1010–1016
25. Giel, O.: Runtime analysis for a simple multiobjective evolutionary algorithm. Tech-Report, Dept. Computer Science, Univ. Dortmund, Germany (2003)

26. Thierens, D.: Convergence time analysis for the multi-objective counting ones problem. In: Proc. Conf. Evolutionary Multiobjective Optimization (EMO-03), LNCS 2632. (2003) 355–364
27. Asho, I.: Interactive Knapsacks: Theory and Applications. Ph.D. Thesis, Tech Report No.: A-2002-13, Department of Computer and Information Sciences, University of Tampere (2002)
28. Frieze, A., Clarke, M.: Approximation algorithms for  $m$ -dimensional 0-1 knapsack problem: Worst case and probabilistic analysis. *European J. Operations Research* **15** (1984) 100–109
29. Erlebach, T., Kellerer, H., Pferschy, U.: Approximating Multiobjective Knapsack Problems. *Management Science* **48** (2002) 1603–1612
30. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problem. *J. ACM* **22** (1984) 463–468
31. Beyer, H.G., Schwefel, H.P., Wegener, I.: How to Analyse Evolutionary Algorithms? *Theoretical Computer Science* **287** (2002) 101–130
32. Scharnow, J., Tinnefeld, K., Wegener, I.: Fitness landscapes based on sorting and shortest path problems. In: Proc. Parallel Problem Solving From Nature (PPSN VII), LNCS 2439. (2002) 54–63
33. Droste, S., Jansen, T., Tinnefeld, K., Wegener, I.: A new framework for the valuation of algorithms for black-box optimization. In: Proc. Foundations of Genetic Algorithms Workshop (FOGA VII). (2002) 197–214
34. Deb, K., Others: A Fast Non-Dominated Sorting Genetic Algorithm for Multiobjective Optimization: NSGA-II. In: Proc. Parallel Problem Solving from Nature (PPSN-VI), LNCS. (2000) 849–858
35. Knowles, J.D., Corne, D.W.: Approximating the Non-Dominated Front Using the Pareto Achieved Evolution Strategy. *Evolutionary Computation* **8** (2000) 149–172
36. Kumar, R., Rockett, P.I.: Improved Sampling of the Pareto-front in Multiobjective Genetic Optimization by Steady-State Evolution: A Pareto Converging Genetic Algorithm. *Evolutionary Computation* **10** (2002) 283–314
37. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Proc. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN). (2001)
38. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evolutionary Computation* **7** (2003) 117–132