Runtime Analysis of Evolutionary Algorithms for the Knapsack Problem with Favorably Correlated Weights

Frank Neumann[1] and Andrew M. Sutton[2]

[1] Optimisation and Logistics, School of Computer Science, The University of Adelaide, Australia
[2] Department of Computer Science, University of Minnesota Duluth, USA

Abstract. We rigorously analyze the runtime of evolutionary for the classical knapsack problem where the weights are favorably correlated with the profits. Our result for the (1+1) EA generalizes the one obtained in [1] for uniform constraints and show that an optimal solution in the single-objective setting is obtained in expected time $O(n^2(\log n + \log p_{\max}))$, where $p_{\max}$ is the largest profit of the given input. Considering the multi-objective formulation where the goal is to maximize the profit and minimize the weight of the chosen item set at the same time, we show that the Pareto front has size $n + 1$ whereas there are sets of solutions of exponential size where all solutions are incomparable to each other. Analyzing a variant of GSEMO with a size-based parent selection mechanism motivated by these insights, we show that the whole Pareto front is computed in expected time $O(n^3)$.

1 Introduction

Evolutionary algorithms [2] and other bio-inspired algorithms have been applied to a wide range of combinatorial optimization and engineering problems. They imitate the evolution process in nature in order to come up with good solutions for a given optimization or design problem. The advantage of evolutionary computation methods lies in their easy applicability to new problems and they often provide satisfying solutions to new problems at hand.

Evolutionary algorithms make uses of random decision in their main operators such as mutation and selection and the area of runtime analysis considers bio-inspired computing methods as a special class of randomized algorithms. A lot of progress has been made over the last 20 years regarding the theoretical understanding of bio-inspired computing techniques (see [3–5] for comprehensive presentations). One of the classical problems studied quite early in the literature are linear pseudo-Boolean functions. Although linear functions are easy to optimize the first proof that showed that a simple (1+1) EA optimizes any pseudo-Boolean linear function in expected time $O(n \log n)$ has been quite involved [6]. Later on, the result has been considerably improved and simplified by using drift analysis [7–9].

Considering linear functions with a linear constraint leads to the classical knapsack problem. Although some early studies have been carried out for constrained combinatorial optimization problems including the knapsack problem [10], the area has been somehow neglected in the area of runtime analysis until quite recently. The mentioned results mainly concentrated on special isolated problem instances. Considering general knapsack instances it has been shown in [11] that a multi-objective approach is able to achieve a 1/2-approximation for the knapsack problem when using helper objectives. Furthermore, the approximation ability of a specific multi-objective approach in terms of the structure of knapsack instances has been investigated in [12].

Recently, linear functions with a uniform constraint have been studied in [1]. This is equivalent to the knapsack problem where all items have a weight of 1. We extend these studies to the class of knapsack problem with favorably correlated weights, i.e for any two items i and j and with profits $p_i \geq p_j$ implies $w_i \leq w_j$ for the weights. We study the single-objective setting where the profit of the knapsack should be maximized subject to the weights of the items meeting a given capacity bound W. Furthermore, we investigate the multi-objective setting where the goal is to maximize the profit and minimize the weight of the chosen items at the same time. For the single-objective setting, we generalize the result on uniform weights given in [1] to knapsack instances with favorably correlated weights. Furthermore, we investigate the multi-objective problem of maximizing profit and minimizing weight for the knapsack problem with favorably correlated weights. We study a variant of Global SEMO (GSEMO) [13, 14] which is a baseline algorithm frequently investigated in the area of runtime analysis for evolutionary multi-objective optimization [15–18]. Investigating the multi-objective setting, we show that even favorably correlated weights can lead to an exponentially large set of search points that are all incomparable to each other. This can potentially lead to a large population in evolutionary multi-objective algorithms such as GSEMO that store at each time step all incomparable search points found so far. Based on this, we introduce size-based parent selection mechanism and show that GESEMO using this parent selection method computes the whole Pareto front for the multi-objective setting in expected time $O(n^3)$.

The outline of the paper is as follows. In Section 2, we introduce the knapsack problem with favorably correlated weights and the single- and multi-objective formulations studied in this paper. We investigate the structure of the objective space for the multi-objective setting and characterize optimal solutions for the single-objective problem in Section 3. In Section 4, we analyze the (1+1) EA with respect to its runtime behavior and analyze the runtime of a multi-objective approach to compute the whole Pareto front in Section 5. Finally, we finish with some conclusions.

2 Preliminaries

We consider the classical knapsack problem. The input is given by n items. Each item i has an integer profit $p_i \geq 1$ and an integer weight $w_i \geq 1$, $1 \leq i \leq n$. For

Algorithm 1: (1+1) EA

```
Choose x ∈ {0,1}^n uniformly at random.;
repeat
    Obtain x' from x by flipping each bit with probability 1/n.;
    If f(x') ≥ f(x), set x := x';
until stop;
```

our investigations, we consider the special class of instances where the weights are favorably correlated to the profits, i.e. for two items i and j, $p_i \geq p_j$ implies $w_i \leq w_j$. W.l.o.g, we assume $p_1 \geq p_2 \geq \ldots \geq p_n \geq 1$ and $1 \leq w_1 \leq w_2 \ldots \leq w_n$. This means that item i dominates each item j with $j > i$. We consider the search space $\{0,1\}^n$ and for a search point $x \in \{0,1\}^n$, we have $x_i = 1$ if item i is selected and $x_i = 0$ if it is not selected. We denote by $p(x) = \sum_{i=1}^{n} p_i x_i$ the profit and by $w(x) = \sum_{i=1}^{n} w_i x_i$ the weight of x. Furthermore, we denote by $p_{\max} = p_1$ is the largest profit and $w_{\max} = w_n$ is the largest weight of the given input.

The knapsack problem with favorably correlated weights is a generalization of the problem of optimizing a linear function with a uniform constraint investigated in [1] where we have $w_i = 1$, $1 \leq i \leq n$. As for the case of a uniform constraint, the knapsack problem with favorably correlated weights can be easily solved by a greedy algorithm which includes the items as they appear in the bit string. However, analyzing the behavior of evolutionary algorithms is interesting for this problem as it allows to understand the working behavior of this type of algorithms for the problem.

2.1 Single-Objective Optimization

In the single-objective case, we have given a weight bound W in addition to the given weights and profits. The goal is to compute a solution x with weight $w(x) \leq W$ and whose profit $p(x)$ is maximal among all solution meeting this weight constraint.

We investigate the classical (1+1) EA shown in Algorithm 1. It starts with a solution $x \in \{0,1\}^n$ chosen uniformly at random. In each iteration an offpring x' is produced by flipping each bit of the current solution x with probability $1/n$. The offspring replaces the current solution if it is not worth with respect to fitness

For the $(1+1)$ EA, we consider the single-objective fitness function

$$f(x) = (\min\{W - w(x), 0\}, p(x))$$

and maximize f with respect to the lexicographic order. This implies that for infeasible solutions the weight is reduced until a feasible solution is obtained. Furthermore, each feasible solution has a better fitness than any infeasible solution.

Algorithm 2: GSEMO Algorithm

```
choose x ∈ {0,1}^n uniformly at random;
determine f(x);
P ← {x};
repeat
    choose x ∈ P uniformly at random;
    create x' by flipping each bit x_i of x with probability 1/n;
    determine f'(x');
    if x' is not strongly dominated by any other search point in P then
        include x' into P;
        delete all other solutions z ∈ P with f'(z) ⪯ f'(x') from P
until stop;
```

Analyzing the runtime of the (1+1) EA, we consider the expected number of fitness evaluations until an optimal search point has been obtained for the first time. This is called the expected optimization time of the algorithm.

2.2 Multi-Objective Optimization

In the multi-objective setting, we aim to maximize the profit and minimize the weight at the same time. We consider the multi-objective fitness function $f'(x) = (w(x), p(x))$ which gives the profit and weight of a given solution x. A solution y (weakly) dominates a solution x ($y \succeq x$) iff $p(y) \geq p(x)$ and $w(y) \leq w(x)$. A solution y strongly dominates x if $y \succeq x$ and $f'(x) \neq f'(y)$. The notion of dominance translates to the objective vector.

The classical goal in multi-objective optimization is to compute for each non-dominated objective vector v a solution x with $f'(x) = v$. The set of all non-dominated objective vectors is called the Pareto front of the given problem.

We consider the algorithm called GSEMO given in Algorithm 2. The algorithm has been frequently investigated in the area of runtime analysis of evolutionary multi-objective optimization [14, 15]. It can be seen as a generalization of the (1+1) EA to the multi-objective case. It starts with a solution x chosen uniformly at random from the considered search space and stores in its population P for each non-dominated objective vector found so far a corresponding solution. In each iteration an individual $x \in P$ is chosen uniformly at random for mutation and x produces then an offspring x' by flipping each bit with probability $1/n$. In the selection step, x' is added to the population if it is not strongly dominated by any other individual in P. If x' is added to the population all individuals that are (weakly) dominated by x' are removed from P.

For many multi-objective optimization problems the number of non-dominated objective vectors can grow exponentially in the problem size n. We will show that the Pareto front for the knapsack problem with favorably correlated weights has size at most $n + 1$. However, we will show in Section 3 that there are sets of solutions with size exponential in n that are all incomparable to each other.

Algorithm 3: Size-based Parent Selection

1 Let $P_i = \{x \in P \mid |x|_1 = i\}$, $0 \leq i \leq n$ and $I = \{i \mid P_i \neq \emptyset\}$.;
2 Choose $j \in I$ uniformly at random and choose parent $x = \arg\max\{p(y) \mid y \in P_j\}$ with the largest profit.

These exponentially many trade-offs motivate the following parent selection mechanism to focus the search of GSEMO. We call the size of a solution x the number of items it contains, i.e. the size is given by the number of its 1-bits $|x|_1$. In our size-based parent selection (see Algorithm 3), we determine the number of 1-bits j that a parent should have by choosing it uniformly at random from the available sizes. Afterwards, we choose the solution with maximum profit from the solutions in P having exactly j 1-bits. GSEMO with size-based parent selection differs from GSEMO by using Algorithm 3 instead of line 5 in Algorithm 2. Note that the population of the algorithm may still grow exponentially with the problem size as the survival selection is not changed.

Analyzing the runtime of GSEMO with size-based parent selection, we consider the expected number of iterations until for each Pareto optimal objective vector a corresponding solution has been obtained. This is called the expected optimization time of the multi-objective algorithm.

3 Structure of the Objective Space

We now examine our class of problems in terms of the structure of solutions in the multi-objective objective space. Our investigations will also point out how optimal solutions look like in the single-objective setting.

We first show that the Pareto front for any choice of the profit and weights of our class of instances of the knapsack problem has a Pareto front of size $n+1$.

We define the set $X^* = \{x \mid x = 1^k 0^{n-k}, 0 \leq k \leq n\}$ of size $n+1$ which contains all strings that start with some (or no) 1-bits and have all remaining bits set to 0.

Theorem 1. *The Pareto front consists of the set of objective vectors* $F = \{f(x) \mid x \in X^*\}$.

Proof. Let x be any search point with $|x|_1 = k$ which is not of the form $1^k 0^{n-k}$. We have $p(x) \leq p(1^k 0^{n-k})$ and $w(x) \geq w(1^k 0^{n-k})$ as $p_1 \geq \ldots \geq p_n$ and $w_1 \leq \ldots \leq w_n$ which implies that x is (weakly) dominated by $1^k 0^{n-k}$. As every search point x with $|x|_1 = k$ is (weakly) dominated by $1^k 0^{n-k}$, $0 \leq k \leq n$, the Pareto optimal objective vectors are given by the set F. □

Let $x^* = \arg\max\{p(x) \mid x \in X^* \wedge w(x) \leq W\}$ be the feasible search point with the largest profit in X^*. This search point has the property that it is the feasible search point with the largest number of 1-bits in X^*. The following corollary shows that x^* is an optimal solution of the constrained single-objective problem.

Corollary 1. *Let $x^* \in X^*$ be the feasible solution with the largest number of 1-bits that is feasible. Then x^* is an optimal solution to the constrained single-objective knapsack problem with favorably correlated weights.*

Proof. From the proof of Theorem 1, we know that any search point is (weakly) dominated by a search point in X^*. X^* is the feasible solution with the largest profit in X^*. This implies that x^* has the maximum profit among all feasible solutions as no feasible dominated solution can have a larger profit than x^*. □

The previous observations show that all Pareto optimal objective vectors as well as the maximum profit for the constrained single-objective problem have corresponding solutions that all start with some (or no) 1-bits and have all remaining bits set to 0.

An important questions that arises when considering evolutionary multi-objective optimization is whether there can be many incomparable solutions even if the Pareto front is small. This might cause difficulties to the search of an evolutionary multi-objective algorithm. We now construct an exponentially large set of solutions and corresponding objective vectors that are incomparable to each other. We set $p_i = 2^{n-i}$ and $w_i = 2^{i-1}$, $1 \leq i \leq n$. Let $n = 4k$, $k \geq 1$ a positive integer. We define

$$S(k) = \{0110, 1001\}, \text{if } k = 1$$

and

$$S(k) = \{01x10, 10x01 \mid x \in S(k-1)\}, \text{if } k \geq 2.$$

This recursively defines sets of solutions for a fixed value of k based on sets for $k-1$. Sets for the k are obtained from sets for $k-1$ by adding two bits to the left and two bits to the right of each string. The values of these bits are chosen in the way that the strings obtained do not dominate each other. This can be done as profits are exponentially decreasing and weights are exponentially increasing according to their position in the string. An illustration of the (incomparable) objective vectors for $S(4)$ with $|S(4)| = 2^4 = 16$ and the corresponding Pareto front for $n = 16$ is given in Figure 1.

We now prove that the size of $S(k)$ grows exponentially in $n = 4k$ and that all solutions in the set are incomparable to each other.

Theorem 2. *Let $n = 4k$, $k \geq 1$ a positive integer. Then $S(k)$ contains $2^k = 2^{n/4}$ search points which are all incomparable among each other.*

Proof. The proof is by induction on k. For $k = 1$, we have two strings 0110 and 1001. We have $p(0110) = 6 < p(1001) = 9$ and $w(0110) = 6 < w(1001) = 9$ which implies that the two strings are incomparable. For the induction step we assume that the set $S(k)$ consists only of incomparable solutions and show that $S(k+1)$ also only includes incomparable solutions. For each string of $S(k)$ we add 2 digits to the left and 2 digits to the right and produce two new strings in $S(k+1)$ for each string in $S(k)$. Hence, we have $|S(k+1)| = 2 \cdot S(k)| = 2^{(n/4)+1}$.

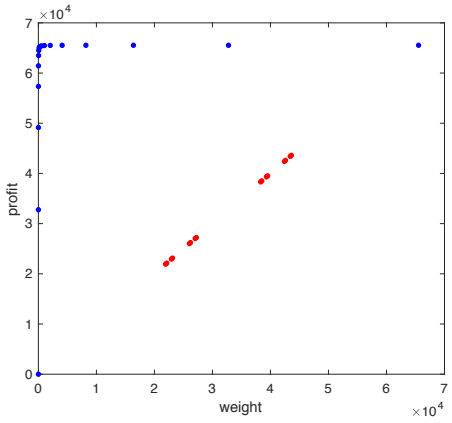

Fig. 1. Illustration of Pareto front (blue) and incomparable objective vectors of S(4) (red) for $p_i = 2^{n-i}$ and $w_i = 2^{i-1}$, $1 \leq i \leq n$ when $n = 16$.

Based on the assumption that all solution of $S(k)$ are incomparable, we show that all solutions in $S(k+1)$ are incomparable.

Assuming that the added bits all take on a value of 0, the profit and the weight of each solution in $S(k)$ increases by a factor of 4 where both values have been less than $2^n - 1$ and are therefore less than $4 \cdot 2^n - 1 = 2^{n+2} - 4$ after the four 0-bits have been added. We now have to take into account the effect of the 1-bits added in the two cases ($01x10$ and $10x01$). Adding 01 to the left and 10 to the right furthermore increases the profit by $2^{n+2} + 2$ and the weight by $2 + 2^{n+2}$. Adding 10 to the left and 01 to the right increases the profit by $2^{n+3} + 1$ and the weight by $1 + 2^{n+3}$. All solutions of $S(k+1)$ where the same pattern has been added are again incomparable after the addition as the profit and weight increased by at least $2^{n+2} + 2$ which is greater than the profit and weight of any solution in $S(n)$.

For the pattern $01x10$ all solutions have profit at most $2^{n+3} - 1$ and weight at most $2^{n+3} - 1$. For the pattern $10x01$ all solutions have profit at least $2^{n+3} + 1$ and weight at least $2^{n+3} + 1$. Hence, solutions belonging to different patterns are incomparable. This implies that all solutions of $S(k+1)$ are incomparable which completes the proof. □

Although the number of trade-offs that might occur is exponential, each non Pareto optimal solution can be improved towards the Pareto front. The reason for this is that any solution that is not Pareto optimal can be improved by a certain number of 2-bit flips. Let x with $|x|_1 = k$ be a non Pareto optima solution. We can transfer it into the Pareto optimal solution $1^k 0^{n-k}$ by a set of 2-bit flips where the first bit that is flipped consists of an arbitrary 0-bit among the first k bits and the second bit that is flipped consists of an arbitrary 1-bit among the last $n - k$ bits in x. Clearly each of these 2-bit flips is accepted as they produce an offspring that dominates x. This means that any evolutionary

algorithm flipping two randomly chosen bits can obtain a some progress towards the Pareto front even if it has to work with a large population.

4 Runtime Analysis of (1+1) EA

We first investigate the runtime behavior of the classical (1+1) EA and study the time to obtain an optimal solution. The proof considers two phases. In the first phase a feasible solution is obtained. After having obtained a feasible solution, the expected time until for the first time an optimal solution has been obtained is analyzed.

Theorem 3. *The expected optimization time of the (1+1) EA on the knapsack problem with favorably correlated weights is* $O(n^2(\log n + \log p_{\max}))$.

Proof. We first show that the expected time until the (1+1) EA has produced a feasible solution is $O(n^2)$ by adapting the $O(n^2)$ bound for optimizing linear functions. Let x be an infeasible solution and consider the function

$$g(x) = \left(\sum_{i=1}^{n} w_i\right) - w(x)$$

giving the weight of the bits set to 0 in x. A feasible solution has value

$$W - w(x) = W - \sum_{i=1}^{n} w_i + g(x) \geq 0.$$

For technical reasons set $w_{n+1} = 0$. We define fitness layer

$$A_i = \left\{ x \mid \sum_{j=(n+1)-i}^{n+1} w_j \leq g(x) < \sum_{j=(n+1)-(i+1)}^{n+1} w_j \right\}, 0 \leq i \leq n-1,$$

dependent on the value of $g(x)$. Having an infeasible solution of layer $x \in A_i$, one of the last $n-i-1$ bits of x is currently set to 1. Flipping this bit produces an offspring $x' \in A_j$ with $j > i$ as it increases $g(x)$ by at least w_{n-i-1}. The probability for such a mutation step is at least $1/(en)$. There are at most n improvements until a feasible solution with $W - w(x) = W - \sum_{i=1}^{n} w_i + g(x) \geq 0$ has been obtained which implies that a feasible solution is obtained after an expected number of $O(n^2)$ steps.

Having reached a feasible solution, we consider the difference

$$\Delta(x) = p(1^k 0^{n-k}) - p(x),$$

where $1^k 0^{n-k}$ is an optimal solution according to Corollary 1. We consider the drift on $\Delta(x)$. As done in [1] for the case of uniform weights, we define

$$\text{loss}(x) = \sum_{i=1}^{k} p_i \bar{x}_i \text{ and } \text{surplus}(x) = \sum_{i=k+1}^{n} p_i x_i.$$

Note that $\Delta(x) = \text{loss}(x) - \text{surplus}(x)$. Let r be the number of zeros among the first k bits in x (contributing to the loss) and s be the number of ones among the last $n-k$ bits in x (contributing to the surplus). Have have $p_i \geq p_j$ and $w_i \leq w_j$ iff $i \leq j$. This implies that each item belonging to the first k bits has a profit at least as high as any profit of the last $n-k$ bits and a weight at most as high as any of the last $n-k$ bits.

We consider different situations depending on the values of r and s. If $r > s$ then any of the r missing bits among the first k bits can be flipped and will be accepted. The sum of theses gains is $surplus(x) \geq \Delta(x)$ and the expected progress in this situation is at least

$$\frac{r}{en}\Delta(x) \geq \frac{1}{en}\Delta(x).$$

If $r = s$ then any of the missing r bits among the first k bits and any of the 1-bits among the last $n-k$ bits can be flipped to achieve an accepted solution. The sum of the progress obtainable by theses 2-bit flips is at least $\Delta(x)$ and the expected progress in this situation is at least

$$\frac{r^2}{en^2} \cdot \Delta(x) \geq \frac{1}{en^2} \cdot \Delta(x).$$

The situation $r < s$ is not possible for a feasible solution x as otherwise the search point $1^{k+1}0^{n-k-1}$ would be a feasible solution contradicting that 1^k0^{n-k} is an optimal solution.

Overall, we have at each time time step t where the (1+1) EA has obtained a feasible but non-optimal solution x of value $\Delta_t(x)$

$$E[\Delta_{t+1}(x) \mid \Delta_t(x)] \leq \left(1 - \frac{1}{en^2}\right) \Delta_t(x).$$

Using the multiplicative drift theorem [8] and the upper bound $np_{\max}$ and lower bound 1 on $\Delta(x)$ for any non-optimal solution x, we get the upper bound of $O(n^2(\log n + \log p_{\max}))$ on the expected time until the (1+1) EA has obtained an optimal solution. □

It should be noted that $\Omega(n^2)$ is a lower bound for the (1+1) EA for knapsack instances with favorably correlated weights as this bounds already holds for special instances where the weights are all 1 [1]. The reason for this lower bound are special 2-bit flips that are necessary in the case that a current non-optimal solution has a maximal number of 1-bits.

5 Runtime Analysis of GSEMO

We know analyze the expected runtime until GSEMO with size-based parent selection has computed the whole Pareto front. The proof works by considering a first phase in which a Pareto optimal solution is obtained. Afterwards missing Pareto optimal solutions can be produced by flipping a specific bit to obtain a still missing Pareto optimal objective vector.

Theorem 4. *The expected optimization GSEMO with Size-based Parent Selection on the knapsack problem with favorably correlated weights is $O(n^3)$.*

Proof. We first upper bound the time until the algorithm has produced the search point 1^n. This solution is Pareto optimal as it has the largest possible profit and once obtained will not be removed from the population. We follow the proof for the $O(n^2)$ bound of optimizing linear pseudo-Boolean functions and use fitness-based partitions weight respect to the profit of the solutions and define fitness layer

$$A_i = \left\{ x \mid \sum_{j=1}^{i} p_j \leq p(x) \leq \sum_{j=i+1}^{n} p_j \right\}, 0 \leq i \leq n-1,$$

as the set of all search points whose profit is at least as high as the profit of the first i profits and whose profit is less than the profits of the first $i+1$ profits. Furthermore, the search point 1^n constitute the optimal layer A_n of profit $p(1^n) = \sum_{i=1}^{n} p_i$.

Consider the solution with the largest profit in the population. The probability to choose this solution x for mutation is at least $1/(n+1)$ as the set I contains at most $n+1$ values and once it has been determined how many 1-bits the parent should have the individual with that number of 1-bits having the largest profit is selected. Assume that the solution of largest profit currently belongs to layer A_i, $0 \leq i \leq n-1$. In order to obtain a solution belonging to layer A_j, $j > i$, one of the leading $i+1$ bits is not set to 1 and can be flipped to obtain a profit increase of at least p_{i+1}. As $x \in A_i$ before the mutation, this leads to an offspring whose profit is increased by at least p_{i+1} and therefore belonging to layer A_j with $j > i$. The probability to select the individual with the largest profit in the population for mutation is $1/(n+1)$ and flipping the bit leading to an improvement in terms of fitness levels has probability at least $1/(en)$. There are $n+1$ different fitness layers which implies that the search point 1^n is produced after an expected number of $O(n^3)$ steps.

In the following, we work under the assumption that the search point 1^n has already been included in the population. The search point 1^n is Pareto optimal as it has the largest possible profit and will stay in the population once it has been obtained. Furthermore, for each Pareto optimal solution there does not exist any other solution in the population that has the same number of ones. As long as not all Pareto optimal objective vectors have been obtained, a new Pareto optimal objective vector can be obtained by selecting a Pareto optimal solution x with $|x| = i$ for which Pareto optimal objective vector with solution size $i+1$ or $i-1$ does not exist yet in the population. Flipping the 0-bit of x corresponding to a largest profit not yet been included in the solution leads to a Pareto optimal solution y with $|y| = i+1$ for $0 \leq i \leq n-1$. Similarly, flipping the 1-bit of x corresponding to a largest profit selected in the solution leads to a Pareto optimal solution y with $|y| = i-1$ for $1 \leq i \leq n$. Choosing such an individual x for mutation has probability is at least $1/(n+1)$ and flipping the bit necessary to increase the number of Pareto optimal objective vectors obtained

has probability at least $1/(en)$. Hence a new Pareto optimal objective vector is produced after an expected number of $O(n^2)$ steps. There are at most n Pareto optimal objective vector that have not been obtained after the search point 1^n has been included in the population. Hence, after an expected number of $O(n^3)$ the population consists of $n+1$ solutions, one for each Pareto optimal objective vector. □

It should be noted that using Size-based Parent Selection where in each step the solution with the smallest weight (instead of the largest profit) is selected would lead to the same result. Here one would show that the search point 0^n is included in the population after an expected number of $O(n^3)$ steps (maximizing $(\sum_{i=1}^{n} w_i) - w(x)$ and considering always the solution with the smallest weight in the population) and show that the other Pareto optimal objective vectors are included after an additional phase of an expected number of $O(n^3)$ steps.

6 Conclusions

Constrained combinatorial optimization problems play a crucial role in real-world applications and evolutionary algorithms have been widely applied to constrained problems. With this paper, we have contributed to the theoretical understanding of evolutionary algorithms for constrained optimization problems by means of rigorous runtime analysis. We generalized the result for the (1+1) EA obtained for uniform weights given in [1] to favorably correlated weights. Furthermore, we investigated the multi-objective formulation of the knapsack problem. Our results show that although the Pareto front has size $n+1$, there can be exponentially large sets of non Pareto optimal objective vectors that are all incomparable. Motivated by these insights, we introduced a size-based parent selection mechanism and have shown that GSEMO using this parent selection method is able to compute the whole Pareto front in expected time $O(n^3)$.

Acknowledgment

This work has been supported through Australian Research Council (ARC) grant DP160102401.

References

1. Friedrich, T., Kötzing, T., Lagodzinski, G., Neumann, F., Schirneck, M.: Analysis of the (1+1) EA on subclasses of linear functions under uniform and linear constraints. In Igel, C., Sudholt, D., Witt, C., eds.: Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA 2017, Copenhagen, Denmark, January 12-15, 2017, ACM (2017) 45–54
2. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series. Springer (2015)

3. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics: Foundations and Recent Developments. Volume 1. World Scientific (2011)
4. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization:Algorithms and Their Computational Complexity. 1st edn. Springer-Verlag New York, Inc., New York, NY, USA (2010)
5. Jansen, T.: Computational complexity of evolutionary algorithms. In Rozenberg, G., Bäck, T., Kok, J.N., eds.: Handbook of Natural Computing. Springer (2012) 815–845
6. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. **276**(1-2) (2002) 51–81
7. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artif. Intell. **127**(1) (2001) 57–85
8. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64** (2012) 673–697
9. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Combinatorics, Probability and Computing **22** (2013) 294–318
10. Zhou, Y., He, J.: A runtime analysis of evolutionary algorithms for constrained optimization problems. IEEE Trans. Evolutionary Computation **11**(5) (2007) 608–619
11. He, J., Mitavskiy, B., Zhou, Y.: A theoretical assessment of solution quality in evolutionary algorithms for the knapsack problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014, IEEE (2014) 141–148
12. Kumar, R., Banerjee, N.: Running time analysis of a multiobjective evolutionary algorithm on simple and hard problems. In Wright, A.H., Vose, M.D., Jong, K.A.D., Schmitt, L.M., eds.: Foundations of Genetic Algorithms, 8th International Workshop, FOGA 2005, Aizu-Wakamatsu City, Japan, January 5-9, 2005, Revised Selected Papers. Volume 3469 of Lecture Notes in Computer Science., Springer (2005) 112–131
13. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. IEEE Transactions on Evolutionary Computation **8(2)** (2004) 170–182
14. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC '03). IEEE Press (2003) 1918–1925
15. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Natural Computing **5**(3) (2006) 305–319
16. Friedrich, T., Neumann, F.: Maximizing submodular functions under matroid constraints by evolutionary algorithms. Evolutionary Computation **23**(4) (2015) 543–558
17. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica **65**(4) (2013) 754–771
18. Qian, C., Yu, Y., Tang, K., Yao, X., Zhou, Z.: Maximizing non-monotone/non-submodular functions by multi-objective evolutionary algorithms. CoRR **abs/1711.07214** (2017)