Journal of Theoretical and Applied Information Technology
30th November 2018. Vol.96. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPLICATION OF FITNESS SWITCHING GENETIC ALGORITHM FOR SOLVING 0-1 KNAPSACK PROBLEM

[1]KIM JUN WOO

[1]Associate Professor, Dong-A University, Department of Industrial and Management Systems Engineering, South Korea

E-mail: [1]kjunwoo@dau.ac.kr

ABSTRACT

Fitness switching genetic algorithm is a sort of genetic algorithm, which was initially developed for solving combinatorial optimization problems with rare feasible solutions. Compared to previous genetic algorithms, fitness switching genetic algorithm has three distinguishing procedures including fitness switching, fitness leveling and simple local search, which enable the infeasible solutions to be included within the population. Consequently, fitness switching genetic algorithm can effectively explore the search space of given problem by utilizing infeasible solutions, even if it is difficult to find arbitrary feasible solutions. On the contrary, 0-1 knapsack problem is a well-known combinatorial optimization problem that typically has many feasible solutions, and this paper aims to apply fitness switching genetic algorithm to solve this problem in order to investigate applicability of the algorithm. To this end, fitness switching, fitness leveling and simple local search procedures are tailored to 0-1 knapsack problem, and a revised algorithm structure is proposed. Consequently, this paper demonstrates that combinatorial optimization problems with many feasible solutions also can be solved by applying fitness switching genetic algorithm. Especially, fitness switching genetic algorithm is easy to implement in that it does not require repair or penalization procedures for handling infeasible solutions.

Keywords: *0-1 Knapsack Problem, Genetic Algorithm, Combinatorial Optimization, Metaheuristics, Operations Research*

1. INTRODUCTION

0-1 knapsack problem (KP) is a classical combinatorial optimization problem, which is characterized by n items with non-negative weight w_i s and value v_i s (i = 1, 2, …, n), and its objective is to find the optimal set of items that maximizes total value under total weight constraint [1][2][3]. Moreover, this paper assumes $w_i \geq 1$ and $v_i \geq 1$ for all i = 1, 2, …, n, for convenience.

Typically, 0-1 knapsack problem is formulated as an integer programming model as follows:

$$\text{Max. Total Value} = \sum_{i=1}^{n} v_i x_i$$

$$\text{Subject to } \sum_{i=1}^{n} w_i x_i \leq t \quad (1)$$

$$x_i \in \{0,1\},$$

where t denotes pre-specified upper bound of total weight. The optimal solution of 0-1 KP can be obtained by applying exact methods such as branch and bound [4], however, approximate methods such as genetic algorithm (GA) [5] and tabu search (TS) [6] can be more effective when n is large, due to NP-hardness of the problem [7][8].

It is straightforward that the feasibility of a solution of KP is dependent on its total weight, and a solution of KP is feasible if and only if its total weight is smaller than or equal to t. In general, arbitrary feasible solutions of given KP can be identified easily. Moreover, even if a solution is infeasible, it is not difficult to convert it into a feasible one by deleting some items included within the solution. Therefore, conventional population-based search methods, such as GA, for solving KP typically utilize the populations which consist of feasible solutions.

On the contrary, this paper applies fitness switching genetic algorithm (FSWGA) to 0-1 KP in order to effectively utilize the infeasible solutions during search procedure. Initially, FSWGA was developed to solve combinatorial optimization

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

problems with rare feasible solutions, and it allows the infeasible solutions to be included in population [9][10]. In this context, this paper has two main objectives: (i) To refine FSWGA to deal with combinatorial optimization problems with many feasible solutions. (ii) To investigate the applicability of FSWGA to a wide range of combinatorial optimization problems.

The remainder of this paper is organized as follows: Section 2 introduces the key concepts and features of FSWGA, which is refined to solve 0-1 KP in Section 3. The experiment results obtained by applying the refined FSWGA to 0-1 KP are reported in Section 4. Finally, Section 5 contains some concluding remarks and future research topics.

2. RESEARCH BACKGROUNDS

GA is a sort of stochastic search procedure proposed by Holland [5], and it has been successfully applied to a wide range of combinatorial optimization problems during past decades [11][12]. Typically, GA maintains a population of solutions for given problem and explores the search space by applying three genetic operators, selection, crossover and mutation [13]. Since there are many feasible solutions within the search space of general combinatorial optimization problems, conventional GAs generally utilize populations of feasible solutions. Moreover, the infeasible solutions generated during search procedure can be dealt with by applying two approaches, repair and penalization [11]. Note that repair procedure is used to convert an infeasible solution into a feasible one [14][15], while the role of penalization is to decrease the fitness value of infeasible solution [16][17][18]. However, designing repair or penalization procedures can be non-trivial task for some problems. Furthermore, such conventional GAs cannot deal with combinatorial optimization problems with rare feasible solutions effectively.

On the contrary, FSWGA is developed to solve combinatorial optimization problems with rare feasible solutions, such as maze-type shortest path problem (M-SPP) [9][10]. M-SPP is a variant of classical shortest path problem (SPP), which is associated with maze-type network with many dead-ends. Since it is not easy to find arbitrary feasible paths for such networks, FSWGA allows infeasible solutions to be included within population and provides three distinguishing procedures, fitness switching, fitness leveling and simple local search. Among them, the most important procedure is fitness switching, which is used to compute the fitness values of feasible and infeasible solutions in different manners. Let's assume that the fitness value of a solution s is $fitness^{+}(s)$ if it is feasible, while the solution has fitness value $fitness^{-}(s)$ if it is infeasible. Fitness switching procedure of FSWGA suggests that $fitness^{+}(s)$ should be inversely proportional to $fitness^{-}(s)$ as follows:

$$fitness^{+}(s) \propto K \cdot \frac{1}{fitness^{-}(s)} \quad (2)$$

Note that the term $K = 1$ for the initial version of FSWGA, developed to solve M-SPP.

It is straightforward that fitness switching operation should satisfy

$$fitness^{+}(s) \geq fitness^{-}(s). \quad (3)$$

However, too large difference between $fitness^{+}(s)$ and $fitness^{-}(s)$ can impose high selection pressure on infeasible solutions. Therefore, FSWGA maintains appropriate selection pressure by applying fitness leveling procedure, which is used to modify the initial fitness values of both feasible and infeasible solutions.

Inherently, the infeasible solutions cannot be the optimal solution for given problem, even though FSWGA allows them to be included within population. Hence, simple local search procedure of FSWGA is used to slightly modify the infeasible solutions in population in hopes that they would be converted into better solutions. In this paper, fitness switching, fitness leveling and simple local search procedures of FSWGA are refined to solve classical 0-1 KP.

3. FITNESS SWITCHING GENETIC ALGORITHM FOR SOLVING 0-1 KNAPSACK PROBLEM

3.1 Encoding Scheme and Initialization

This paper adopts binary string representation, which is the most widely used encoding scheme for representing solutions for combinatorial optimization problems such as KP. Hence, a solution for KP with n items is represented as a binary string with n genes, [x_1 x_2 … x_n], where x_i s are binary variables and x_i = 1 if and only if item i (i = 1, 2, …, n) is

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

included within a solution. Moreover, a solution of initial population can be easily created by using randomly generated x_i s. This initialization procedure is summarized in Figure 1, where N_p denotes population size. Note that no repair procedure is applied to the initial solutions in Figure 1.

```
01: void initialize(int N_p, int n)
02: {
03:   SET population = new int[ N_p ][ n ]
04:
05:   FOR k = 1 TO N_p
06:     FOR l = 1 TO n
07:       SET u = random real number in (0, 1)
08:
09:       IF u ≤ 0.5 THEN
10:         SET population[ k -1][ l -1] = 0
11:       ELSE
12:         SET population[ k -1][ l -1] = 1
13:       END IF
14:     NEXT l
15:   NEXT k
16: }
```

Figure 1: Initialization Procedure

3.2 Fitness Switching for 0-1 Knapsack Problem

Typically, $fitness^{+}(s)$ for KP is computed by using total value of the items included in solution s as follows:

$$fitness^{+}(s) = \sum_{i=1}^{n} v_i x_i \,, \tag{4}$$

if s satisfies

$$\sum_{i=1}^{n} w_i x_i \le t \,, \tag{5}$$

where t is pre-specified upper bound of total weight. A solution is infeasible for given KP if it does not satisfy (5), and its fitness value should be computed in different manner. Moreover, the fitness function for infeasible solution, $fitness^{-}(s)$ should have following features: First, an infeasible solution s_b is worse than any feasible solution s_a, even if it has larger total value. Thus, $fitness^{-}(s_b)$ cannot exceed $fitness^{+}(s_a)$. Second, quality of infeasible solution should be appropriately measured. For example, let $E(s)$ denote excess weight of a solution s as follows:

$$E(s) = \begin{cases} 0 & , \text{ if } \sum_{i=1}^{n} w_i x_i \le t \\ \sum_{i=1}^{n} w_i x_i - t & , \text{ otherwise} \end{cases} \tag{6}$$

If $0 < E(s_b) < E(s_c)$, both s_b and s_c are infeasible, however, s_b is probably better than s_c in that s_b has smaller excess weight and it can be converted into feasible solution more easily. In other words, $fitness^{-}(s_b)$ should be larger than $fitness^{-}(s_c)$. Thirdly, $fitness^{-}(s)$ has to satisfy

$$fitness^{-}(s) > 0 \,, \tag{7}$$

for every infeasible solution. In addition, designing a penalization procedure which has all of those three features can be a non-trivial task.

In this context, this paper proposes three types of $fitness^{-}(s)$ including

$$fitness_1^{-}(s) = \frac{1}{\sum_{i=1}^{n} v_i x_i} \,, \tag{8}$$

$$fitness_2^{-}(s) = \frac{1}{\sum_{i=1}^{n} w_i x_i} \,, \tag{9}$$

and

$$fitness_3^{-}(s) = \frac{1}{\sum_{i=1}^{n} v_i x_i} \,. \tag{10}$$

Note that

$$0 > \rho_2 > \rho_3 > \rho_1 = -1 \,, \tag{11}$$

where ρ_i is the correlation coefficient between $fitness^{+}(s)$ and $fitness_i^{-}(s)$. Moreover, fitness switching procedure of the original version of FSWGA, developed for solving M-SPP, is most similar to $fitness_1^{-}(s)$ in that $\rho_1 = -1$. However, the feasibility of a solution for KP is dependent on threshold of total weight t, while M-SPP does not

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

have any explicit threshold for solutions. Hence, this paper proposes alternative fitness functions $fitness_2^-(s)$ and $fitness_3^-(s)$, where total weight of solution s is considered. The fitness switching procedure for KP proposed in this paper is summarized in Figure 2.

```
01: single fitness_switching(int[] s, int n, int type)
02: {
03:   SET weight = array of weights
04:   SET value = array of values
05:
06:   SET total_weight = 0
07:   SET initial_fitness = 0
08:   SET t = upper bound of total weight
09:
10:   FOR k = 1 TO n
11:     SET total_weight += s[k-1] × weight[k-1]
12:   NEXT k
13:
14:   IF initial_fitness ≤ t THEN
15:     FOR k = 1 TO n
16:       SET initial_fitness += s[k-1] × value[k-1]
17:     NEXT k
18:   ELSE IF type = 1 THEN
19:     FOR k = 1 TO n
20:       SET initial_fitness += s[k-1] × value[k-1]
21:     NEXT k
22:
23:     SET initial_fitness = 1 / initial_fitness
24:   ELSE IF type = 2 THEN
25:     SET initial_fitness = 1 / total_weight
26:   ELSE IF type = 3 THEN
27:     FOR k = 1 TO n
28:       SET initial_fitness += s[k-1] × value[k-1]×weight[k-1]
29:     NEXT k
30:
31:     SET initial_fitness = 1 / initial_fitness
32:   END IF
33:
34:   RETURN initial_fitness
35: }
```

Figure 2: Fitness Switching Procedure

3.3 Fitness Leveling for 0-1 Knapsack Problem

One important limitation of fitness switching procedure described in previous section is that a feasible solution can have extremely large fitness value, while infeasible ones have relatively small fitness values when v_i, $w_i \geq 1$. Moreover, this disparity in fitness values brings about high selection pressure, which can lead to premature convergence to local optima.

In order to avoid this problem, FSWGA uses fitness leveling procedure to obtain adjusted fitness values,

$$fitness^+{}'(s) = 1 + L \times \frac{fitness^+(s) - \min\limits_{s \in F} fitness^+(s)}{\max\limits_{s \in F} fitness^+(s) - \min\limits_{s \in F} fitness^+(s)} \quad (12)$$

and

$$fitness^-{}'(s) = (1-\alpha) \times \frac{fitness^-(s)}{\max\limits_{s \in I} fitness^-(s)}, \quad (13)$$

where F and I denote the sets of feasible and infeasible solutions within population, respectively, and $F \cup I$ = population. In addition, factor L is used to prioritize feasible solutions over infeasible ones, and $L \geq 1$. Note that the selection pressure of FSWGA is directly proportional to factor L.

The role of factor α is used to slightly decrease the fitness values of infeasible solutions, and $0 < \alpha < 1$ so that the adjusted fitness values $fitness^+{}'(s)$ and $fitness^-{}'(s)$ satisfy

$$0 \leq fitness^-{}'(s) < 1 \leq fitness^+{}'(s) \leq L. \quad (14)$$

The fitness leveling procedure proposed in this paper is summarized in Figure 3.

3.4 Simple Local Search for 0-1 Knapsack Problem

The fitness leveling procedure helps the competitive infeasible solutions to survive in selection phase of GA. However, the infeasible solutions are not suitable for given problem, inherently. Therefore, FSWGA uses simple local search procedure to slightly modify the infeasible solutions in hopes that they would be converted into better infeasible solutions or feasible solutions.

In solving KP, simple local search procedure can be used to exclude a randomly

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

chosen item from an infeasible solution, and this procedure is summarized in Figure 4.

```
01: single fitness_leveling(int[] s , single
initial_fitness, single L , single α )
02: {
03:   SET max_feasible_fitness = maximum
          fitness value of feasible solutions
04:   SET min_feasible_fitness = minimum
          fitness value of feasible solutions
05:   SET max_infeasible_fitness = maximum
          fitness value of infeasible solutions
06:
07:   SET adjusted_fitness = 0
08:
09:   IF initial_fitness < 0 THEN
10:     SET adjusted_fitness = (1 – α )
        ×initial_fitness/max_infeasible_fitness
11:   ELSE
12:     SET adjusted_fitness
        = 1 + L ×(initial_fitness – min_feasible_
        fitness)/(max_feasible_fitness – min_
        feasible_fitness)
13:   END IF
14:
15:   RETURN adjusted_fitness
16: }
```

Figure 3: Fitness Leveling Procedure

```
01: single simple_local_search(int[] s , int n , int
type )
02: {
03:   SET cur_item_set = ϕ
04:
05:   FOR k = 1 TO n
06:     IF s [k –1 ] = 1 THEN
07:       ADD k –1 to cur_item_set
08:     END IF
09:   NEXT k
10:
11:   SET del_item = a randomly chosen
                     element of cur_item_set
12:
13:   SET s [del_item] = 0
14:
15:   RETURN fitness_switching( s , n , type )
14: }
```

Figure 4: Simple Local Search Procedure

Note that simple local search procedure calls fitness switching procedure after an infeasible solution is slightly modified. That is, the fitness value of the solution is also modified after simple local search procedure is done.

3.5 Selection, Crossover and Mutation

FSWGA for solving KP proposed in this paper is based on the common standard GA (SGA) framework, which uses selection, crossover and mutation operators in order to generate new population from current one. Moreover, this paper uses conventional genetic operators widely used in GAs for various combinatorial optimization problems.

Selection operator is used to generate a mating pool, a set of solutions selected from current population, where mating pool and population have identical size and one solution in current population can be selected two or more times. In this paper, well-known roulette wheel method is used, which indicates that an individual solution s_k in current population is selected with a probability proportional to its own fitness value,

$$\frac{\text{fitness of } s_k}{\sum_{i=1}^{N_p} \text{fitness of } s_i}. \tag{15}$$

The role of crossover operator is to recombine the genes of two solutions in mating pool, parents, in order to generate two new solutions, offspring. This paper uses uniform crossover, which is known as a more exploratory approach than traditional single-point or two-point crossover approaches. For example, let $x_{p1,i}$ and $x_{p2,i}$ denote the i th genes of parent 1 (father) and parent 2 (mother), respectively. Then, the uniform crossover suggests that $x_{o1,i} = x_{p1,i}$ and $x_{o2,i} = x_{p2,i}$ with a probability of 0.5, and $x_{o1,i} = x_{p2,i}$ and $x_{o2,i} = x_{p1,i}$ with a probability of 0.5, where $x_{o1,i}$ and $x_{o2,i}$ denote the i th genes of offspring1 and offspring2, respectively.

The mutation operator is used to randomly modify the offspring with small probability, and it helps to maintain the diversity of solutions within population and avoid the premature convergence to local optima. In this paper, common bit flip mutation operator is used, where value of each gene is changed from 1 to 0 and vice versa with small probability.

After mutation operator is applied, the current population is replaced by mating pool, and we have to evaluate fitness of the solutions in new

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

population. The evaluation phase of FSWGA is described in next section. In addition, note that the new population obtained in this manner can contain a number of infeasible solutions but no additional repair or penalization procedures are used in this paper.

3.6 Evaluation

In the original version of FSWGA, fitness leveling procedure is applied to selection phase of GA, while fitness switching and simple local search procedures are incorporated into evaluation phase. On the contrary, all of those procedures are applied to evaluation phase in this paper, as shown in Figure 5. In other words, FSWGA can be implemented with existing selection, crossover and mutation operators, if evaluation phase is appropriately designed.

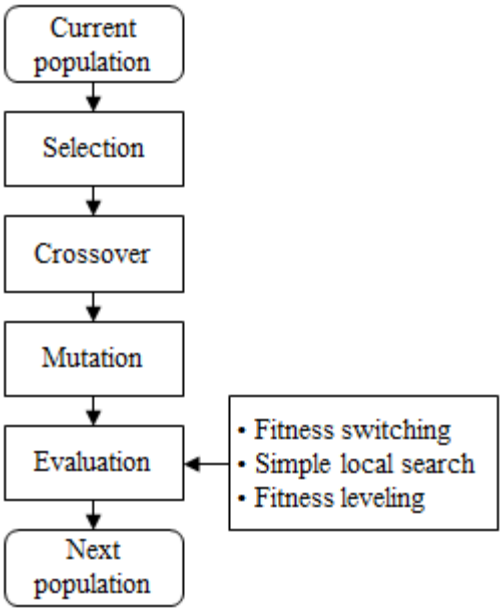

Figure 5: Overall Structure of FSWGA

The objective of evaluation phase is to compute the fitness values of the solutions within current population, and the evaluation phase of FSWGA proposed in this paper is summarized in Figure 6.

At first, fitness switching procedure is used to compute initial fitness values. If a solution in current population is feasible, its initial fitness value, computed by (4), will be larger than 1, while an infeasible solution has initial fitness value smaller than 1, which is computed by (8)~(10).

Next, simple local search procedure is applied to the solutions with initial fitness values smaller than 1, in order to convert them into better solutions by slight modifications. Moreover, their fitness values are also modified, and the modified fitness values are larger than the initial fitness values. If an infeasible solution is converted into a feasible solution, its modified fitness value will be larger than 1. On the contrary, if the modified solution is still infeasible, its modified fitness value cannot exceed 1.

Finally, fitness leveling procedure is applied to both feasible and infeasible solutions within current population in order to maintain appropriate selection pressure, which is directly proportional to factor L. Consequently, each solution in current population will have adjusted fitness value, which is used by next selection operator.

```
01: void evaluation(int type, single L, single α)
02: {
03:   SET n = number of items
04:   SET N_p = population size
05:   SET fitness = new single[N_p]
06:   SET population = current population
07:
08:   FOR k = 1 TO N_p // Fitness switching
09:     SET fitness[k-1] = fitness_switching(
          population[k-1], n, type)
10:   NEXT k
11:
12:   FOR k = 1 TO N_p //Simple local search
13:     IF fitness[k-1] < 1 THEN
14:       SET fitness[k-1] = simple_local_
          search(population[k-1], n, type)
15:     END IF
16:   NEXT k
17:
18:   FOR k = 1 TO N_p //Fitness leveling
19:     SET fitness[k-1] = fitness_leveling(
          population[k-1], fitness[k-1], L, α)
20:   NEXT k
21: }
```

Figure 6: Evaluation Phase

Note that fitness leveling procedure has to be applied after fitness switching and simple local procedures have been done, since it utilizes $\max_{s\in F} fitness^{+}(s)$, $\min_{s\in F} fitness^{+}(s)$ and $\max_{s\in I} fitness^{-}(s)$.

Journal of Theoretical and Applied Information Technology
30th November 2018. Vol.96. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. EXPERIMENT RESULTS

In order to investigate the performance of FSWGA for KP, three versions of FSWGAs with $fitness_1^-(s)$, $fitness_2^-(s)$ and $fitness_3^-(s)$ had been applied to a KP with 50 items, listed in Table 1, with varying upper bound of total weight t. For each case, 10 repetitions of experiments had been performed under population size N_p = 50, crossover rate = 0.8, mutation rate = 0.01, maximum iteration number = 500, L = 2 and α = 0.1. Moreover, elitism policy for preserving the 5 best solutions within each generation was applied.

Table 1: Knapsack Problem With 50 Items

Item ID	Weight	Value	Item ID	Weight	Value
1	39	66	26	33	45
2	19	60	27	35	16
3	34	21	28	38	130
4	13	139	29	35	58
5	15	65	30	13	37
6	17	95	31	27	116
7	31	6	32	20	64
8	12	91	33	14	43
9	21	12	34	14	108
10	24	85	35	13	75
11	23	12	36	23	121
12	11	53	37	35	67
13	40	99	38	25	60
14	32	48	39	33	33
15	36	96	40	34	48
16	38	81	41	23	98
17	20	70	42	15	91
18	13	122	43	24	102
19	28	83	44	39	120
20	17	129	45	23	61
21	26	54	46	28	58
22	36	31	47	23	119
23	33	52	48	20	122
24	29	96	49	32	71
25	17	57	50	28	64

Note that this KP is rather hard to solve in that item values and item weights are not strongly correlated, as shown in Figure 7. On the contrary, the item values have weak inverse correlation with item weights, and the correlation coefficient between them is about -0.2. In this paper, three upper bound values listed in Table 2 are considered, where Optimal-Total-Value is the total value of optimal solution, and MS-Excel Solver failed to find the optimal solution when t = 500.

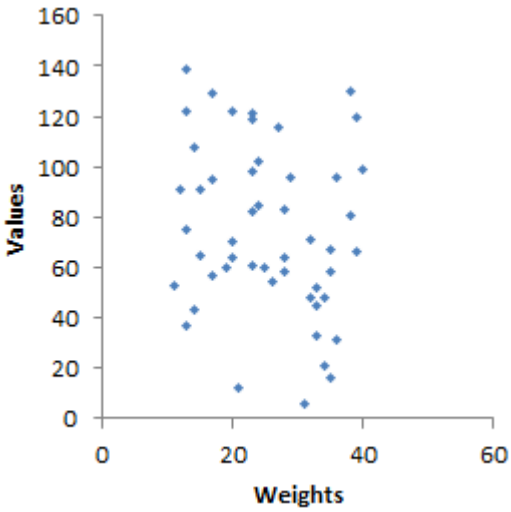

Figure 7: Item Value Versus Item Weight

Table 2: Upper Bound Of Total Weight

ID	Upper bound of total weight (t)	Optimal-Total-Value
1	200 (Low)	1282
2	500 (Moderate)	2373
3	1000 (High)	3465

The experiment results are summarized in Table 3~Table 5, where Best-Total-Value is the total value of the best solution identified by each version of FSWGA. Moreover, success rate is the proportion of experiments where the optimal solution is identified.

In Table 3~Table 5, we can get following observations: Firstly, $fitness_2^-(s)$ shows superior performances to the others in that it produces the highest average Best-Total-Value, the smallest standard deviation of Best-Total-Value and the highest success rate in all cases. Since $fitness_2^-(s)$ is inversely proportional to total weight, a solution s is infeasible if $fitness_2^-(s) < 1$. In other words, $fitness_2^-(s)$ evaluates the fitness of a solution most appropriately. Meanwhile, $fitness_1^-(s)$ is inversely proportional to total value, which is generally proportional to total weight. However, as mentioned earlier, item values and item weights of given KP are not strongly correlated, and $fitness_1^-(s)$ sometimes produce wrong fitness values. For example, let's consider two infeasible solution, s_1^- and s_2^-. If s_1^- has smaller total value and larger total weight, it is straightforward that s_1^- is worse infeasible solution than s_2^-. Nevertheless, $fitness_1^-(s_1^-) > fitness_1^-(s_2^-)$, since s_1^- has smaller

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

total value. Similarly, $fitness_3^-(s)$ produces wrong fitness values more often, and this indicates that $fitness^-(s)$ for a given problem should be carefully chosen.

Secondly, the performance of FSWGA is significantly affected by the upper bound of total weight t. Especially, the three versions of FSWGA performs badly when $t = 500$, as shown in Table 4. Of course, other solution methods also have difficulties in solving highly complex combinatorial optimization problems, and this indicates that FSWGA can be further revised to deal with such complexities.

Nevertheless, we can see that all FSWGAs have identified optimal or nearly optimal solutions in almost experiments, and this suggests that the search strategy of FSWGA can be used to solve combinatorial optimization problems with many feasible solutions, such as KP.

Table 3: Best-Total-Values For Low Upper Bound Of Total Weight ($t = 200$)

	$fitness_1^-(s)$	$fitness_2^-(s)$	$fitness_3^-(s)$
Average	1280.2	**1281.4**	1275.2
Max.	1282	1282	1282
Min.	1276	1276	1256
Std.Dev.	2.75	**1.80**	9.47
Success rate	70%	**90%**	50%

Table 4: Best-Total-Values For Moderate Upper Bound Of Total Weight ($t = 500$)

	$fitness_1^-(s)$	$fitness_2^-(s)$	$fitness_3^-(s)$
Average	2363.7	**2370.4**	2362.7
Max.	2373	2373	2373
Min.	2341	2360	2344
Std.Dev.	9.72	**3.85**	10.21
Success rate	10%	**40%**	10%

Table 5: Best-Total-Values For High Upper Bound Of Total Weight ($t = 1000$)

	$fitness_1^-(s)$	$fitness_2^-(s)$	$fitness_3^-(s)$
Average	3460.9	**3462.8**	3459.8
Max.	3465	3465	3465
Min.	3451	3460	3453
Std.Dev.	4.64	**2.27**	3.25
Success rate	40%	**50%**	10%

Figure 8 shows changes in the maximum total value during search procedure of a single experiment with FSWGA based on $fitness_2^-(s)$ and $t = 500$. Since the maximum total value is consistently increased, we can conclude that FSWGA is useful to explore the search space of KP.

Similarly, Figure 9 shows changes in infeasible ratio, the proportion of infeasible solutions within population, during search procedure of same experiment. In Figure 9, the infeasible ratios of first two populations are 90.0% and 66.0%, respectively. However, the infeasible ratio is rapidly decreased, and later populations have quite lower infeasible ratios. What is important is that such infeasible solutions are not "destroyed" by repair procedure, and they are dealt with by the three operations of FSWGA, instead of penalization procedure, which requires careful configuration.

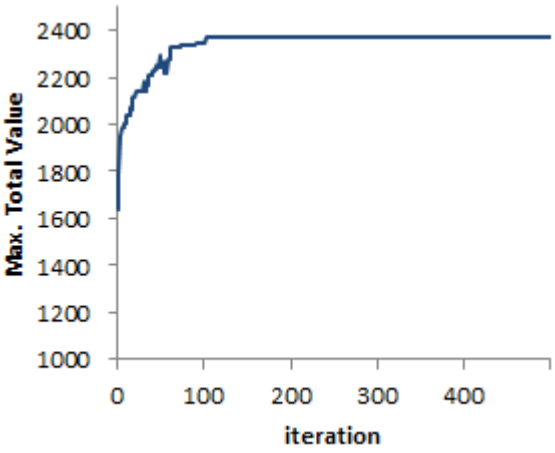

Figure 8: An Example Of Change In Maximum Total Value During Search Procedure

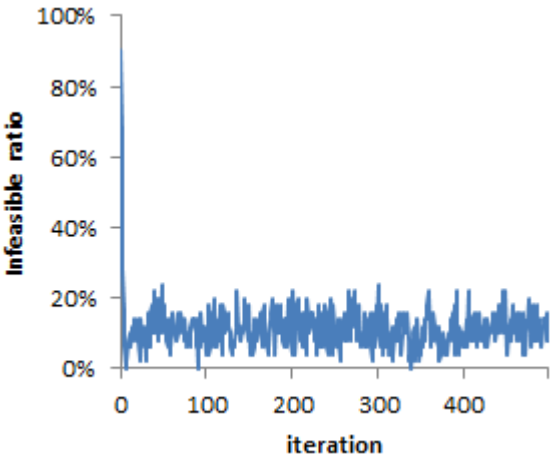

Figure 9: An Example Of Change In Infeasible Ratio During Search Procedure

Note that the initial population contains a number of feasible solutions in Figure 9, which is not the case with original version of FSWGA applied to M-SPP. In other words, early populations of FSWGA may contain only infeasible solutions if given problem has rare feasible solutions. In such

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

cases, FSWGA has two objectives, one is to find any feasible solutions and the other is to choose the optimal solution among feasible ones. On the contrary, FSWGA in this paper can focus on the second objective, since it is very easy to find arbitrary feasible solutions for given KP.

5. CONCLUSIONS

The most important benefit of FSWGA is that it can be applied to solve combinatorial optimization problems with rare feasible solutions. In addition, FSWGA can be easily designed and implemented in that it does not require any problem-dependent repair or penalization procedures, however, this additional benefit of FSWGA was not appropriately investigated in previous literature.

In this context, this paper develops refined FSWGAs and applies them to solve one of well-known combinatorial optimization problem with many feasible solutions, classical 0-1 KP. The experiment results demonstrate that the search strategy of FSWGA is also useful even if given problem has many feasible solutions. In addition, this paper suggests that the fitness switching procedure is the most important element of FSWGA, and $fitness^{-}(s)$ must be carefully chosen.

Thus, the author plans to apply FSWGA to a wider range of combinatorial optimization problems develop more effective fitness switching procedures in future.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(Ministry of Science, ICT & Future Planning) (NRF-2017R1C1B1008650).

REFRENCES:

[1] D. Zou, L. Gao, S. Li, and J. Wu, "Solving 0-1 Knapsack Problem by a Novel Global Harmony Search Algorithm", *Applied Soft Computing*, Vol.11, No.2, 2011, pp.1556-1564.

[2] A.J. Umbarkar, and M.S. Joshi, "0/1 Knapsack Problem using Diversity based Population Genetic Algorithm", *International Journal of Intelligent Systems and Applications*, Vol.6, No.10, 2014, pp.34-40.

[3] K.K. Bhattacharjee, and S.P. Sarmah, "Shuffled Frog Leaping Algorithm and Its Application to 0/1 Knapsack Problem", *Applied Soft Computing*, Vol.19, 2014, pp.252-263.

[4] S. Martello, D. Pisinger, and P. Toth, "New Trends in Exact Algorithms for the 0-1 Knapsack Problem", *European Journal of Operational Research*, Vol.123, No.2, 2000, pp.325-332.

[5] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Michigan, 1975.

[6] F. Glover, "Tabu Search: A Tutorial", *Interfaces*, Vol.20, No.4, 1990, pp.74-94.

[7] D. Pisinger, "Where Are the Hard Knapsack Problems?", *Computers and Operations Research*, Vol.32, No.9, 2005, pp.2271-2284.

[8] J.W. Kim, "Performance Comparison of Neighborhood Structures of Tabu Search Algorithms for Sequencing Problems", *Advanced Science Letters*, Vol.23, No.10, 2017, pp.10423-10426.

[9] J.W. Kim, and S.K. Kim, "Fitness Switching Genetic Algorithm for Solving Combinatorial Optimization Problems with Rare Feasible Solutions", *Journal of Supercomputing*, Vol.72, No.9, 2016, pp.3549-3571.

[10] J.W. Kim, and S.K. Kim, "Genetic Algorithms for Solving Shortest Path Problem in Maze-type Network with Precedence Constraints", *Wireless Personal Communications*, forthcoming.

[11] J.W. Kim, "Candidate Order based Genetic Algorithm (COGA) for Constrained Sequencing Problems", *International Journal of Industrial Engineering: Theory, Applications and Practice*, Vol.23, No.1, 2016, pp.1-12.

[12] M. Kumar, M. Husian, N. Upreti, and D. Gupta, "Genetic Algorithm: Review and Application", *International Journal of Information Technology and Knowledge Management*, Vol.2, No.2, 2010, pp.451-454.

[13] Z. Michalewicz, *Genetic Algorithm+Data Structures=Evolution Programs*, Springer Science and Business Media, Heidelberg, 2013.

[14] P. Chootinan, and A. Chen, "Constraint Handling in Genetic Algorithms using a Gradient-based Repair Method", *Computers and Operations Research*, Vol.33, No.8, 2006, pp.2263-2281.

[15] S. Salcedo-Sanz, "A Survey of Repair Methods Used as Constraint Handling Techniques in Evolutionary Algorithms", *Computer Science Review*, Vol.3, No.3, 2009, pp.175-192.

[16] D.W. Coit, A.E. Smith, and D.M. Tate, "Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Problems", *INFORMS Journal on Computing*, Vol.8, No.2, 1996, pp.173-182.

[17] Ö, Yeniay, "Penalty Function Methods for Constrained Optimization with Genetic Algorithms", *Mathematical and Computational Applications*, Vol.10, No.1, 2005, pp.45-56.

[18] M. Schlüter, and M. Gerdts, "The Oracle Penalty Method", *Journal of Global Optimization*, Vol.47, No.2, 2010, pp.293-325.