Constraint Handling of an Optical Components Selection Problem using a new Genetic Crossover Scheme

Mohammad Amin Dallaali and Malin Premaratne

Advanced Computing and Simulation Laboratory (AXL)
Department of Electrical and Computer System Engineering
P.O. Box: 35 Monash University, Clayton Victoria 3800 Australia
{Amin.Dallaali and Malin.Premaratne}@eng.monash.edu.au

Abstract. This paper proposes a new crossover scheme called Controlled Content Crossover (CCC). CCC is applied to solve a commercially important optimization problem associated with the selection of the optimum set of components in optical network. CCC searches for the optimized solution and at the same time controls the value of the constrained parameter. This parameter is the optical component dispersion value in this paper. Therefore, by preserving the feasibility of the search points, CCC performs a genetic search over the feasible space. The simulation results show that the performance of the CCC is comparable with the answers obtained from CPLEX software and is better than the simple repair-based Genetic Algorithm proposed in the paper.

Keywords: Genetic Algorithms, Controlled Content Crossover, Feasibility, Dispersion Compensation, Optical Networks.

1 Introduction

The size and the complexity of the current and perceived future optical communication systems have forced us to look for efficient but effective ways for designing and subsequently supporting them. However, due to the limitations in availability of specialists relative to the demand and the associated cost, telecommunications carriers are looking for ways of automating the design of optical networks using computer aided design (CAD) tools. Building CAD tools in large optical networks design has become a major challenge because most of the algorithms associated with the device placement and subsequent configuration tasks belong to NP-complete or NP-hard complexity classes, which are in the category of the toughest problems in computer science. It is well known that no exact methods for solving such problems in polynomial time can be found using conventional serial computing techniques. However, reasonably accurate solutions for such NP-hard problems can be found by using techniques that mimics the biological evolutionary

path such as Genetic Algorithms (GAs). Therefore, GAs have found increasing usages in such massive optimization problems. The basic rules of the genetic algorithms are very simple and hence provide the opportunity for a wide range of the applications to use the algorithms [1]. This robustness however, does not remove the necessity of designing specific genetic operators. These operators for example are used in constraint handling methods.

Using special operators was considered by Michalewicz [2] as one of the classified constrained handling approaches for mathematical programming problems. In [2], Michalewicz highlighted that in the area of nonlinear programming, evolutionary computations did not address the issue of constraints in a systematic way. For the penalty-based method, Genetic Algorithm penalized unfeasible solutions , although there is no guideline to design penalty function and any generalization effort may cause computational overhead [2]. In [3] Koziel and Michalewic classified the "constraints handling methods" and counted one of their category as "the techniques based on preserving feasibility of solutions". They also talked about techniques where a "chromosome" gives instructions on how to build a feasible solution". It continued: "for example, a sequence of items for the Knapsack problem can be interpreted as: "take an item if possible"; such interpretation would lead always to a feasible solution." A classification of the methods for handling unfeasible solutions for continuous numerical optimization problem was made by Richardson (1989) and reported by Michelawicz [4]. Richardson considered two different paradigms: modifying the genetic operators and penalizing the strings that fail to satisfy all the constraints. In [5], Hinterding and Michelawicz claimed that: "to maintain feasibility of an individual, a specialized operator (which incorporates the knowledge about problem-specific constraints) should be used. Furthermore, Sakawa et al. [6] used double string coding in their work for integer programming problem. They used a double string coding mainly to overcome the difficulties associated with the Partially Matched Crossover scheme. Chu and Beasley [7] dealt with a similar situation when they developed a binary representation that had the same concept as the double string coding. They used this coding scheme for solving the Knapsack problems.

In this paper, we present Controlled Content Crossover (CCC), a new crossover method that maintains the feasibility of the solutions in genetic algorithm implementation of configuration problem in optical network design. The algorithm has been designed so that it looks for the optimal solution inside the feasible region. The concept of double string coding is used for the indexing of the bit strings so that the labeled elements maintain the same position when they pass through the algorithm operators. As the result, the identities of the components are saved and they are trackable in any stage of the process. Although there is a penalty based cleaning phase at the end of the selection part of the algorithm, but it is not used frequently and therefore it does not make a major effect on the whole process. This cleaning phase removes the strings with the dispersion values less than the minimum limit. Because of the small effect of this part on the whole algorithm, CCC does not cause the latency that the death-penalty type methods cause for the algorithms. In this paper, the

proposed crossover scheme is implemented to solve the problem of selection of components for a point-to-point optical network. Dallaali and Premaratne [8] studied such a problem using genetic algorithms where they examined several features of their genetic processor. However, in that paper, the implemented idea was a penalty-based method that could not perfectly find the optimized target. By using CCC method, we show that the results in [8] can be improved dramatically. The rest of the paper is organized as follows: In Section 2, formulation of the problem is given. In Section 3 details of the Controlled Content Crossover scheme is described for the optical network component allocation problem. Section 4 presents the simulation and results. In section 5 conclusions are provided and in section 6 further work is outlined.

2 Problem Definition

The problem is to find the optimal selection of the available off-the-shelf optical network components so that this selection satisfies the problem constraints. The constraints are defined as limitations over the parameters such as dispersion and attenuation values of the optical components. These components include the Dispersion Compensation Modules (DCM) and amplifiers. This has the mathematical form of an Integer Programming problem, which can be formulated as:

$$\text{Minimize:} \quad \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\text{Subject to:} \quad D_{\min} = D - \frac{\Delta D}{2} \le \sum_{i=1}^{n} d_i x_i \le D + \frac{\Delta D}{2} = D_{\max} \tag{2}$$

$$\sum_{i=1}^{n} \gamma_i x_i \le \Gamma \tag{3}$$

$$\sum_{i=1}^{n} x_i < N \tag{4}$$

Where x_i is an integer number that represents the number of times that the component type i is selected. c_i, d_i and γ_i are respectively the cost, dispersion and attenuation values of the component i. n is the number of different component types are selected and N is a maximum limit for the total number of the components. Γ is the maximum value used for the upper bound limit of the total attenuation and D is a constant dispersion value which is symmetrically centered between the upper limit $D_{\max}$ and the lower limit $D_{\min}$. ΔD is the difference of the upper and the lower limits compared to D. In this paper however, the algorithm is developed and performed with one constraint which is the dispersion value. Therefore, all explanations are provided with the constraints on the dispersion value and hence the problem can be rewritten without considering formulas (3) and (4). More constraints can be added as needed for future work.

3 The Algorithm

Figure 1 shows an overview of the algorithm. It starts with an initial population selection phase and then enters the iterative loop which is the genetic mating part of the algorithm. The exchange unit is the main part of the algorithm which is designed to control the constraint on the dispersion value.

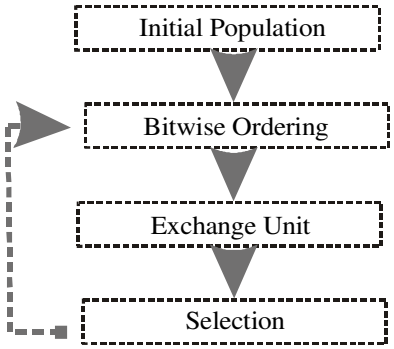

Fig. 1. The overall flowchart of the algorithm

3.1 Initial Population

As it is shown in figure 1 the algorithm starts with an initialization phase where the initial population is randomly selected. The initial population composed of 12 strings each with 63 bits. Bits are integer numbers representing the number of times that each component type is selected. One of these strings is shown in figure 2. The number of the initial strings, 12, and the number of the bits that they have, 63, are arbitrarily selected for the test purposes.

3.2 Coding and Data Binding

The initial 12 strings make the first group of parents of the genetic generations. They are the first chromosomes of the genetic process that are consequently followed by the chromosomes generated during the mating iterations. These strings must be accompanied by some supplementary information in order to enter the genetic iteration loop. The supplementary information consists of the two new types of strings: the attached indexes of the elements of the array and the proportion of the dispersion value that each bit contributes to the total dispersion value of the string. The index of the components must be attached to the other values and must accompany the rest of the information during the genetic process. As formula (5) shows, the proportion of the dispersion value for each bit in the total dispersion value of the string is calculated for all the bits in all of the strings and therefore there will be

12 strings of 63 bits accompanying their counterparts. These values are shown in figure 2 with the notation of PD_i. Later in the exchange section, it will be explained that how these values will be the deterministic factors for sorting the bits of the strings.

$$Perc\_Disp(str,k) = \frac{d_k \times x_k}{\sum_{i=1}^{N} d_i \times x_i} \tag{5}$$

Where the $Perc\_Disp$ (Percentage of Dispersion) value is calculated for the k th bit of the str th string. As it is shown in figure 2 the $Perc\_Disp$ array, the array containing the component indexes and the array containing the number of the selected items are attached together and make three attached arrays of 63 bits for each chromosome. Therefore, at the end of the coding phase, there will be 12 packs of the complex shown in figure 2.

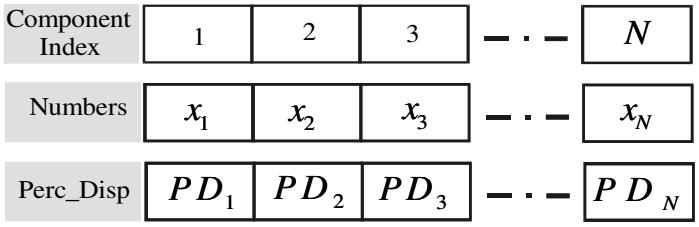

Fig. 2. The initial arrangement of a parent string

3.3 Mating and Breeding Iterations

3.3.1 Ordering

Single run of the genetic iteration is explained in this section. Procedure described here is executed iteratively until the satisfactory solution is obtained. A bit-wise sorting is performed on the input strings so that the 63 bits in each of the three arrays are sorted based on the descending order of $Perc\_Disp$.

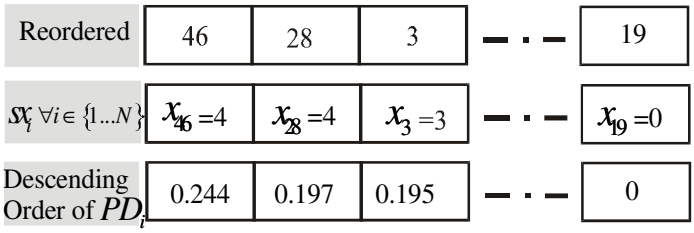

Fig. 3. Reordering of the string and its affiliates before the exchange phase

As the result, the elements with more dispersion values are awarded higher ranks. This reordered input data set is considered as a parent for the next generation. Larger number of selected items and a bigger dispersion value per item are the two factors that cause a larger $Perc\_Disp$ value. Figure 3 illustrates the arrangement of the three arrays of one string that have been sorted according to the $Perc\_Disp$ values.

3.3.2 The Exchange Unit

The exchange unit is the main part of CCC. First, strings are randomly selected for mating. For example as it is shown in the figure 4, string number one and string number 12 make a family and produce two offspring. The first bits of the parents are allocated to the children respectively so that child x' get the first bit of the parent x and child y' get the first bit of the parent y .

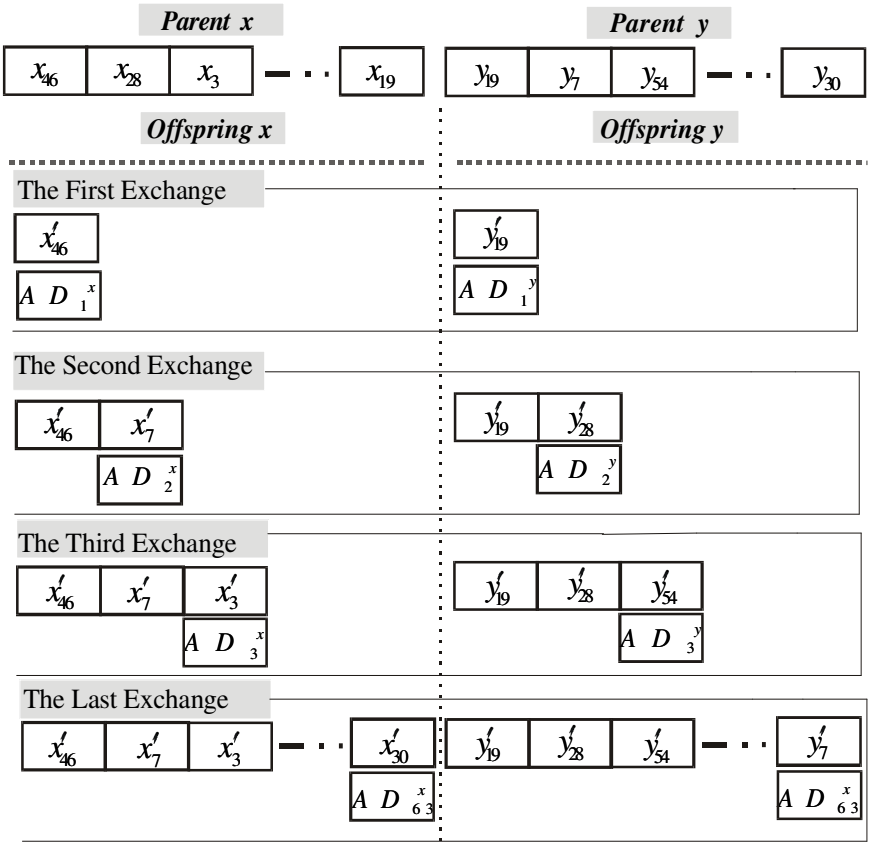

Fig. 4. The exchange procedure of the CCC

The exchange procedure starts from the second bits. In order to do that, a new parameter called Acu_Dsip (Accumulated Dispersion) is defined for all of the bit positions and noted as AD_i. There is a unique Acu_Dsip value for each bit in all strings of the offspring that formulated as:

$$Acu\_Disp(str,k) = \sum_{i=1}^{k} rd_i \times rx_i \tag{6}$$

where rd_i and rx_i are respectively the reordered dispersion values and the reordered bits that contain the integer number of selected items. Both of these arrays have been already ranked based on the descending order of the PD_i values. For the second turn of the exchange process, the AD_i value of the i th position is calculated first and then the $i+1$ th bit with the greater PD_{i+1} value is allocated to the offspring with the smaller AD_i. Simultaneously, the $i+1$ th bit with the smaller PD_{i+1} value is allocated to the offspring with the greater AD_i. With this allocation, it is tried to equally distribute the total dispersion values of the two parents over the two children. Accordingly, as it is shown in figure 4, for the second turn of the exchange phase, the second bit of the x parent goes to the y' offspring and the second bit of the y parents sits at the second position of the x' offspring. Besides the balancing of the dispersion values over the two offspring, there is another rule that should be considered while swapping the bits: the rule was mentioned above is valid if swapping the bits doesn't create the situation that two integers from the same type of component are located in one string. In order to avoid this situation, the algorithm compares the index of the bit which is going to sit in the i th position of the offspring with all of the bits from the first to the $i-1$ th position. If the exchanging integer is equal to one of the settled bits, the destination is changed. This means that CCC sacrifices the equality of the dispersion values in the two offspring in order to prevent the state of having same component type in one string. This case can be seen for the last exchange turn in figure 4.

3.4 Mutation

Two cascaded turns of mutation are performed on the offspring obtained from the exchange phase. Based on the mutation algorithm, any bit greater than zero is replaced by zero and any bit equal to zero is replaced by a random number between zero to five. Experiments show that the minimizing procedure is not done properly without mutation. Therefore, it can be concluded that the mutation here is the optimizing engine of the algorithm and at the same time provides the diversity over the search space while the exchange scheme performs the search over the feasible space.

3.5 Selection

The 12 strings with the best fitness function or equally with the smallest cost value are selected. A penalty based cleaning phase is performed before the selection phase in order to remove the offspring with the total dispersion value less then the limit. This is achieved by increasing the cost of the string with the dispersion value less than the limit to a reasonably large value more than the cost values of all of the strings. Therefore such a string is ignored in the selection process.

4 Simulations and Results

CCC has been applied to several test data sets. Among them, the results of three experiments are presented and analyzed here. In figures 5 to 7 the results of the simulations with a random array of 63 components are shown. The cost and the dispersion values of the test components are selected randomly, although the random values are generated within some limits. In order to create realistic data, these limits have been adopted from the commercial datasheets. Figure 5 shows how the cost function gets minimized by CCC and achieves its final stability after passing through the decreasing transient phase. Figure 5 also compares the final result of the CCC with the result obtained from the CPLEX software. The reason that CCC can not obtain the same result as CPLEX is because the last rule of the exchange scheme described in the section 3.3.2 is not respected perfectly. The implemented scheme can not avoid the situation when the similar integers of both of the exchanging bits (coming from the parents) are in a same offspring string. Therefore the algorithm pays off the similarity of the integers in one string in order to save the balance of the dispersion values over the offspring.

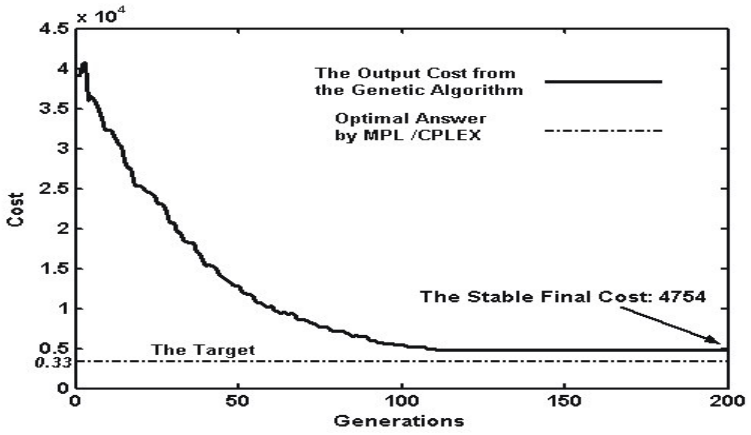

Fig. 5. CCC performance and the target line obtained from CPLEX

Figure 6 shows another comparison between the result of the CCC and a penalty-based GA from our previous work [8]. This method is composed of a simple single-point crossover, a mutation scheme and a selection phase. After mutation, the total dispersion value of each offspring is calculated and if it violates the dispersion constraint, the algorithm adds a penalty value to the total cost of the string. This decreases the fitness of the infeasible strings and consequently their ranks for selection phase. As it is seen CCC outperforms the penalty-based method.

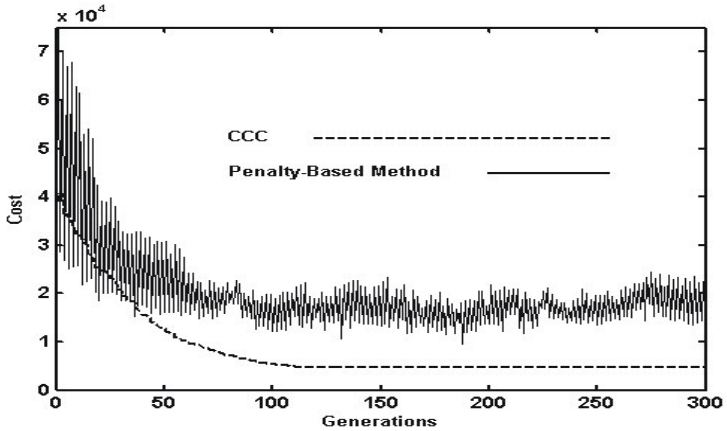

Fig. 6. The output of the CCC and the Penalty-Based genetic algorithm

Figure 7 clears that after the GA passes the transient phase, the dispersion values of the generations satisfy the goal of the algorithm regarding the feasibility of the solutions. In order to show that, the minimum and the maximum total dispersion values of all 12 selected parents have been recorded for each generation and are sketched as a function of the generations. As it is seen, it converges toward a value very close and greater than the minimum dispersion limit.

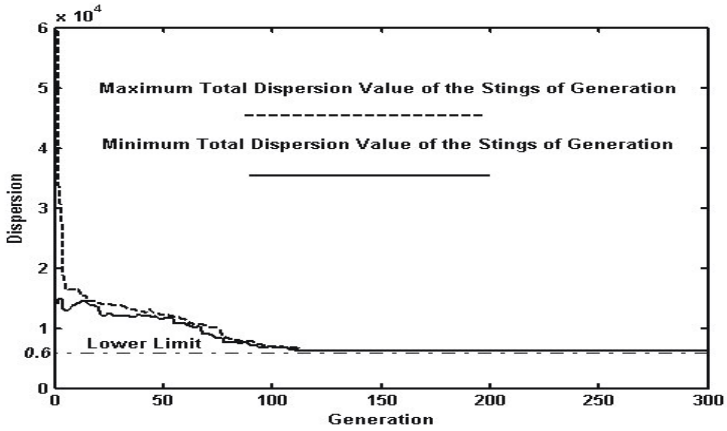

Fig. 7. The maximum and the minimum total dispersion values

Therefore, such a performance can be accepted for any double-bound constrained problem with an upper bound more than the final stabilized dispersion value. In the performed test experiment the lower limit is 6000 and the final obtained value is 6172. Considering the grain size of the dispersion values of the test data which are random numbers between 100 and 1000, this performance is acceptable and satisfying. The performance with smaller grain size data will be considered in future.

Finally figures 8 and 9 show two more experiments with other test data sets. In the figure 8 a relatively small size data set of 10 components is examined. The lower dispersion limit is still 6000 and the values of the input data are arbitrary selected. For figure 9 a data set of 63 components is tried but the lower dispersion limit is 3000 and the dispersion values are changing as a row-tooth function of the cost values. The comparison of the CCC results with the CPLEX answers are shown that for both of the experiments CCC minimizes the cost to a value close to the CPLEX answers. There is a higher difference for the first experiment comparing to the second one. It can be because of the large number of the selected items which may increase the effect of the problem which was explained for the figure 5.

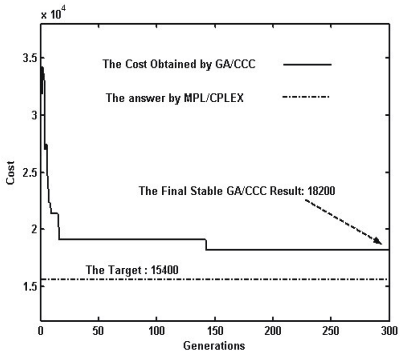

Fig. 8. CCC is implemented on the data set of 10 components

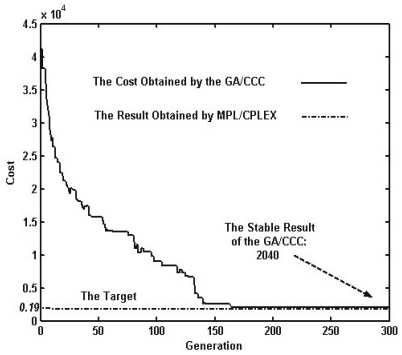

Fig. 9. CCC is implemented on the Third type of data set

5 Conclusions

A new crossover scheme called Controlled Content Crossover (CCC) was introduced and applied to a commercially significant optical component allocation problem. The simulations showed that it searched for the optimum answer while maintaining the feasibility conditions of the underlying equations. This was achieved by specifically designing the genetic operators to satisfy the constraints while searching for the answer. The total dispersion values of the chromosomes considered as the constraint of the problem and CCC kept them more than the lower boundary with a reasonably small offset. This was shown in the simulation results and it meant that by defining a double-sided constraint, CCC is able to follow the procedures toward the optimum answer and at the same time satisfy the constraints.

The results of simulations for three different sets of data and parameters were shown and compared with the answers obtained from CPLEX software. Furthermore, the performance of the CCC was compared with the performance of a penalty-based and CCC outperformed the penalty-based scheme.

6 Further Work

Solving the problem of the exchange scheme is first task to follow for the future. As it was already explained, when two similar integers which are going to be replaced from the parents to the offspring are in a same destination string, both of the two bits tend to go to the opposite string which does not contain their twins. If this happens, one string will receive two bits and the other one does not receive any bit. Performing CCC with several constraints is the second ossible idea for the further work. It is also interesting to perform an accurate analysis for the different offset errors of the test data sets with different grain sizes.

7 References

[1] Goldberg, D. E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Pub. Co. (1989)

[2] Michalewicz, Z.: Genetic Algorithm, Numerical Optimization, and Constraints. Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann Publisher (1995), 151-158

[3] Koziel, S. and Michalewicz, Z: Evolutionary Algorithms, Homomorphous Mapping, and Constrained Parameter Optimization, Evolutionary Computation, Vol. 7. No.1 (1999) 19-44.

[4] Michelawicz, Z.: A survey of the Constraint handling techniques in Evolutionary Computation Methods, Proc. of the 4th Annual Conf. on Evolutionary Programming, MIT Press (1995). 135-155

[5] Hinterding, R. and Michelawicz, Z.: Your Brains and My Beauty: parents matching for Constrained Optimization, Proc. of the 5th Int. Conf. on Evolutionary Computation (1998) 810-815

[6] Sakawa, M. and Kato, K.: Integer programming through Genetic Algorithms with Double Strings Based on Reference Solution Updating, Industrial Electronics Society, IECON 2000. 26th Annual Conference of the IEEE, Vol. 4. (2000) 2744 - 2749

[7] Chu, P.C. and Beasley, J.E.: A Genetic Algorithm for Multidimentional Knapsack Problem, Journal of Heuristics, Vol. 4. Kluwer Academic Publisher (1998) 63-86

[8] Dallaali, M.A. and Premaratne, M.: Configuration of optical network using Genetic Algorithm, Proceeding of ONDM2004, (2004) 279-292