한국해양공학회 2004년도 추계학술대회 논문집 PP.35~39

Shipyard Spatial Scheduling Solution using Genetic Algorithms

Prof. (Dr.) Yoon, Duck Young**; Ranjan Varghese*

*Dept. of Naval Architecture And Ocean Engineering Graduate School of Chosun University Gwangju

**Dept. of Naval Architecture & Ocean Engineering Chosun University Gwangju

KEY WORDS: Genetic Algorithms, Optimisation Spatial layout,

ABSTRACT: *In a shipyard, there exist various critical decision making components pertaining to various production hindrances. The most prominent one is best-fit spatial arrangement for the minimal spatial occupancy with better pick-ability for the erection of the ship in the dock. During the present research, a concept have been conceived to evade the gap between the identification of inter-relationships among a set of blocks to be included on a pre-erection area, and a detailed graphical layout of their positions, is called an Optimal Block Relationship Diagram. A research has been performed on generation of optimal (or near Optimal) that is, with minimal scrap area. An effort has been made in the generation of optimal (or near-optimal) Optimal Block Relationship Diagram with the Goldberg's Genetic Algorithms with a representation and a set of operators are "trained" specifically for this application. The expected result to date predicts very good solutions to test problems involving innumerable different blocks to place. The suggested algorithm could accept input from an erection sequence generator program which assists the user in defining the nature and strength of the relationships among blocks, and could produce input suitable for use in a detailed layout stage.*

1. Introduction

In the industrial setup like that of shipbuilding is faced with the bundles of constraints and the operational hurdles of various natures. The natural phenomenon and consideration with geometrical consideration is one of the serious issues of concern.

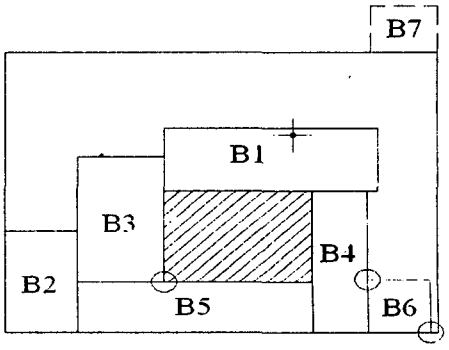
2. Problem Description

In the present day, digital world where day-to-day activities digital technology or in other words specifically in the shipbuilding industry, a new concept called digital shipbuilding has taken birth and blooming. The basic idea is improving production speed and efficiency. World over the businesses are in the transient stage where big industrial giants are relocating to varied global locations to tap Global position, local advantage to expedite growth. Ironically, to practice this kind of business relocation aspect of the core business may turn out to be a tyrant task at the corporate level or in the market domain, compromising the client satisfaction and other deadline related aspects. The developed algorithm will be incorporated into a computer program with the help of Genetic Algorithm. concepts. The objective of paper is to arrive at an optimal spatial occupation schedule capable for dynamic block movement, i.e. replacement of the blocks while the shipbuilding erection process is ongoing.

The solution of this problem lies in generating optimal spatial occupation pattern, this keeps the schedule, details with respect to its corresponding ranking or the date of erection of block intact. The use of genetic algorithm refines the solution characteristics by making the program self reliant and self-thinkable based on system training.

2.1 Heuristical Background

A natural way to improve the bottom left or BL heuristic is to apply BL to other permutation at the expense of more time and more orders. One standard technique would be random-repeat: permutations are repeatedly validated uniformly at random, and the best solution found within the desired time bound is used. Random permutations are however, known to perform poorly.



Author Ranjan Faculty of Naval Architectural Engineering Chosun University in Kwang-Ju

062-230-7881 vran77@yahoo.co.in

with the help of Genetic Algorithm.

Fig. 1 (a) BLF heuristic of Block on Hold

The basic bottom left (BL-routine) usually tends to generate layouts with relatively large empty areas, a second more sophisticated BL heuristic is considered for hybridization with meta-heuristics. The strategy here consists of placing a rectangle into the lowest available position of the object and left justifying it.

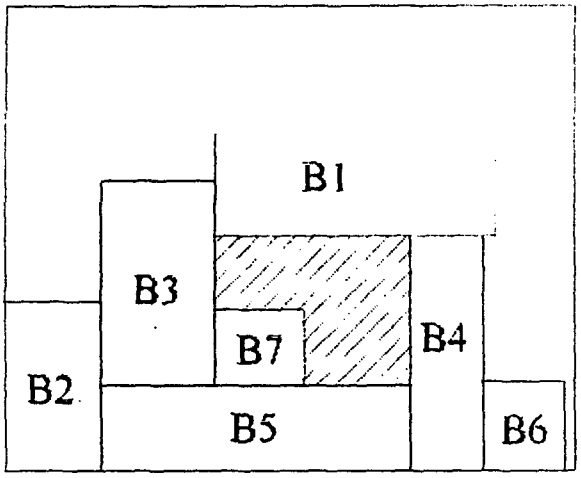


Fig. 1 (b). BLF heuristic of Block placement.

The evolved layout is based on allocation of sufficiently large region in the partial layout rather than on a series of BL moves as it is capable of filling existing gaps in packing pattern. In order to distinguish it from the BL-algorithm its referred to as the BL-Fill (BLF) heuristic. When compared to BL-routine this method results in denser packing patterns.

3. Genetic Algorithm

In preparing to use the conventional genetic algorithm operating on fixed-length character strings (chromosomes) to solve a problem, the user must

(1) Determine the representation scheme

(2) Determine the fitness measure

(3) Determine the parameters and variables for controlling the algorithm, and

(4) Determine a way of designating the result and a criterion for terminating a run

3.1 Implemented GA

Once the four preparatory steps for setting up the genetic algorithm is completed for the genetic algorithm run.

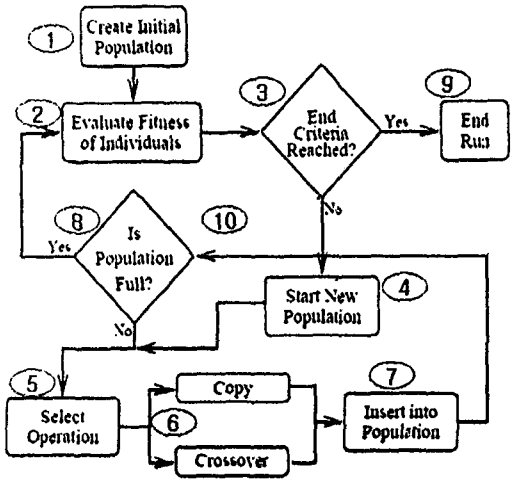


Fig. 2 Flow Diagram of adopted Genetic Algorithm.

The evolutionary process described above indicates how a globally optimum combination of alleles (gene values) within a fixed-size chromosome.

The three steps in executing the genetic algorithm operating on fixed-length character strings are as follows:

(1) Randomly create an initial population of individual fixed length character strings.

(2) Iteratively perform the following sub-steps on the population of strings until the termination criterion has been satisfied:

(A) Assign a fitness value to each individual in the population using the fitness measure.

(B) Create a new population of strings by applying the following three genetic operations. The genetic operations are applied to individual string(s) in the population chosen with a probability based on fitness.

(i) Reproduce an existing individual string by copying it into the new population.

(ii) Create two new strings from two existing strings by genetically recombining substrings using the crossover operation (described below) at a randomly chosen crossover point.

(iii) Create a new string from an existing string by randomly mutating the character at one randomly chosen position in the string.

(3) The string identified by method of result designation (e.g., the best-so-far individual) is because of genetic algorithm for the run. This result may represent a solution (or an approximate solution) to the problem.

3.2 Detailed Implementation Decision Operators

3.2.1 Fitness

Fitness evaluation involves defining an objective or fitness function against which each chromosome is tested for suitability for the environment under consideration. As the algorithm

proceeds we would expect the individual fitness of the "best" chromosome to increase as well as the total fitness of the population as a whole.

Fitness evaluation involves defining an objective or fitness function against which each chromosome is tested for suitability for the environment under consideration. As the algorithm proceeds we would expect the individual fitness of the "best" chromosome to increase as well as the total fitness of the population as a whole.

Fitness Parameter for this problem,

F(A) = Area(Scrap (A))
= Σ_{ai} |x1 (i) - x2(i)|*(Limit Level (PE) - y(i)) ···. (1)

With,
ai = {(x1(i) - y(i))}, {(x2(i) - y(i))}

3.2.2 Explanation of Sigma Scaling

Fitness proportionate selection suffers from premature convergence at times. Therefore, a backup system developed in order to do the exploration. More emphasis is on exploitation as apposed to exploration. Sigma Scaling addresses this problem

This keeps the selection pressure relatively constant over a run.
ExpVal(i,t) = 1+ (fitness(i)
-mean(t))/2*SD(t) ···. (2)
SD is Standard Deviation.
If SD(t) <>0 otherwise ExpVal(i,t)=1.0;

This, for example, gives an individual whose fitness is one SD above the mean 1.5 offspring out of N

3.2.3 Special Penalty Function

Genetic algorithms (GAs) are successfully applied to numerical optimization problems. Since GAs are usually designed for unconstrained optimization, they have to be adapted to tackle the constrained cases, i.e. those in which not all representable solutions are valid.

3.2.4 Crossover

The selected chromosome copied into new population. This copying facilitates the addition of the populations in to agile group to improve the system to develop maximum possible random combinations (Fig.4).

In this step, new offspring created by system for the new population by recombining randomly chosen parts of two selected genes. Thus, enormous samples are created.

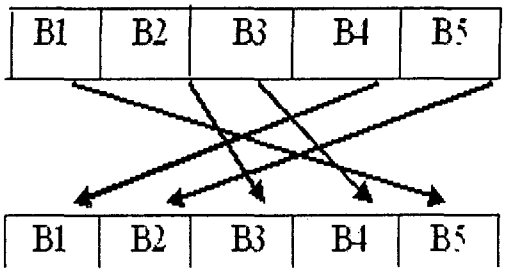


Fig. 3 A typical Crossover operation

The newborn offsprings designated by individual operation as a result. That is, individual with the best fitness as run is result.

```
int OurCrossover(const GAGenome& p1,
        const GAGenome& p2,
    GAGenome* c1, GAGenome* c2){

        GA1DBinaryStringGenome
&mom=(GA1DBinaryStringGenome &)p1;
        GA1DBinaryStringGenome
&dad=(GA1DBinaryStringGenome &)p2;
            int n=0;
    unsigned int site = GARandomInt(0,
            mom.length());
    unsigned int len = mom.length() - site;
            if(c1){
        GA1DBinaryStringGenome
&sis=(GA1DBinaryStringGenome &)*c1;
        sis.copy(mom, 0, 0, site);
    sis.copy(dad, site, site, len); n++; }
            if(c2){
        GA1DBinaryStringGenome
&bro=(GA1DBinaryStringGenome &)*c2;
        bro.copy(dad, 0, 0, site);
    bro.copy(mom, site, site, len);
            n++;
            }
        return n;
            }
```

Fig. 4 The Sample code of Genetic Crossover

3.2.5 Time specific relation

An instance of the on-line scheduling problem is given by n Blocks Bj (j = 1····· n) and m material handling devices Mi (i = 1····· m). Each Block Bj (j =1····· n) has a given processing time pj (Dynamic time) and release time rj (idle time at PE). A material handling devices can process only one Block at a time and a Block can be processed by only one machine at a time. The processing of a Block cannot start before its release time. If preemption is not allowed, then a Block must be processed without

interruption on one material handling devices. If preemption is allowed we may interrupt the processing of a Block and continue it at the same time on another machine or at a later moment on any machine. The first moment at which Block Bj is processed is referred to as its starting time Sj , and the time at which a Block is completed is referred to as its completion time Cj . We say that a machine is idle at time t if it is not processing any Block during an open interval containing t. The problems that we study here concern finding non-preemptive schedules for which the sum, taken over all Blocks, of completion times is minimal. The sum of completion times called the latency.

The algorithm is to construct a feasible schedule on-line, meaning that for any t >0, the schedule restricted to the interval [0, t] must be constructed without knowledge of the Blocks that are not released before time t, i.e., those Blocks with rj >t. The idea behind our algorithm is to delay the time at which Blocks become available for processing to the algorithm, by increasing the release times and to apply the SPT rule to the available Blocks. Since the release times are shifted before applying SPT, we call this algorithm Shifted SPT (SSPT). We generalize their idea and show that the release time of any Block may be delayed, independently of any other Block, until any moment between these two values without reducing the competitive ratio of the algorithm.

3.2.6 Programming Assignment Details

The GA in this assignment is capable of evaluating two functions. The first is the 2-dimensional Rosenbrock valley function, a function which often causes trouble for many optimizers. This function is defined as:

$$F(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \quad \cdots. (3)$$

over the range [0,2] for both x1 and x2. It is to be evaluated for both a resolution of 6 and 10 bits per pattern variable. The second function is a multimodal function, and could be described as an egg crate. There are many valleys that an optimization method like Powell's method or the like could get trapped in. This function is:

$$F(x_1, x_2) = 25(\sin^2(x_1) + \sin^2(x_2)) + x_1^{\,2} + x_2^2 \quad \cdots. (4)$$

over the range [-6,6]. It will be evaluated for a resolution of 8 and 16 bits per pattern variable.

3.2.7 Convergence Checking Operator

The premature convergence is very critical problem in any kind of genetic algorithm codes. This problem has to be dealt with utmost care, since a premature convergence to a result leads to a very unsatisfactory result and might be a scandalous result. In order to counter such a mishap, we introduce a chromosome training technique using a predominant optimisation technique called penalty function method. This method penalises the parameter depending on the desired function and here it acts as a training operator.

Penalty methods use a mathematical function that will increase the objective for any given constrained violation. General transformation of constrained problem into an unconstrained problem is undertaken for this problem.

$$\min T(x) = f(x) + rkP(x) \quad \cdots. (5)$$

where,

f(x) is the objective function of the constrained problem

(rk) is a scalar denoted as the penalty or controlling parameter

P(x) is a function which imposes penalties for infeasibility (note that P(x) is controlled by (rk))

T(x) is the (pseudo) transformed objective

Two approaches exist in the choice of transformation algorithms:

1. Sequential penalty transformations
2. Exact penalty transformations

3.2.8 Termination Conditions

The terminations conditions have to be more specific. Since these cross validation operators signals the termination of the Genetic Algorithm cycle. In our problem, we define the first conditions as 'n!' iteration being conducted; secondly did the least possible scrap area formulation is achieved and thirdly did the minimum time consumed chromosome identified.

All these condition stands fixed in each run of the GA to provide appropriate solution.

4. Conclusion

The developed spatial scheduler program is capable of handling any desired number of blocks required for shipbuilding. The algorithm is claimed to be robust with inter file handling systems, database modifications facilities for customising and producing the network related solutions.The algorithm induced assures the capability foreseen in a dynamic problem with the consistent movement of the blocks into and out of the pre-erection area. Here enormous usage of data structures to handle the huge working database and random operator's workspace. This could produce self-explained graphical colouroutput of the spatial layout

of pre-erection area, which helps in reconsidering and giving proper MIS to take concerned corrective actions. The main advantage of such efforts is that without any additional investment in man and machine, an eventually superfluous working strategy is evolved

Aknowledgements

References

1. Ivor Horton (2002); Beginning Visual C++ 6; Wrox Press.
2. Kyoung, Jun Lee (1995); Development of Spatial Scheduling Expert Systems; Doctoral Thesis.
3. Mitsuo Gen, Runwei Cheng. A (1996); Genetic Algorithms and engineering Design, Wiley-Interscience Publication, John Wiley & Sons, INC.
4. Yoon, Duck Young; Ranjan Varghese; Tae Kyu, Bae; Chung Kon, Koo; Erection Sequence Generator for Ship Building; 2004, 25-27 March; 33rd International conference on computers and industrial engineering, Jeju, Korea.
5. Yoon, Duck Young; Ranjan Varghese; Chung Kon, Koo; (2004) Proceedings of
6. Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, & Frank D. Francone (1998); Genetic Programming ~ An Introduction; Morgan Kaufmann Publishers, Inc. San Francisco, California.
7. D. Todd; P. Sen; Multiple Criteria Scheduling unsing Genetic Algorithm in a Shipyard Environment; ICCAS '97, 9th International Conference on Computer Applications in Shipbuilding;.
8. John R. Koza,; Martin A. Keane; Mathew A. Keane, Mathew J. Streeter; William Mydlowec; Jessen Yu; Guido Lanza; Genetic Programming IV- Routine Human- Competitive Machine Intelliegence; Kluwer Acedemic Publishers; 2003.
9. Sadiq M. Sait; Habib Youssef; Iterative Computer Algorithms with Applications in Engineering; Solving Combinatorial Optimisation Problems; IEEE Computer Society; 1999.