Evolving Design Genes in Space Layout Planning Problems

John S. Gero and Vladimir A. Kazakov
Key Centre of Design Computing,
Department of Architectural and Design Science,
The University of Sydney, NSW 2006 Australia.
e-mail:{john,kaz}@arch.su.edu.au

ABSTRACT

The paper describes the application of a genetic engineering based extension to genetic algorithms to the layout planning problem. We study the gene evolution which takes place when an algorithm of this type is running and demonstrate that in many cases it effectively leads to the partial decomposition of the layout problem by grouping some activities together and optimally placing these groups during the first stage of the computation. At a second stage it optimally places activities within these groups. We show that the algorithm finds the solution faster than standard evolutionary methods and that evolved genes represent design features that can be re-used later in a range of similar problems.

Keywords: genetic algorithms, genetic engineering, evolved genes, layout planning

1. Introduction

The space layout planning problem is one of the most difficult in architectural design. It is practically important in architectural design because it is the basis of the development of most designs. It is important in a wider context because it maps onto a large class of location-allocation problems including VLSI floorplanning, process layouts and facilities layout problems. We will use the formalization of the space layout problem as a particular case of a combinatorial optimization problem - the quadratic assignment problem [1].

As such it is NP-complete and presents all the difficulties associated with this class of problems. Over the years a number of approximate algorithms based on combinations of global and local search techniques and heuristics have been developed specifically for this class of problems. Although they are reasonably efficient for small-scale problems, the computational cost is still too high for large-scale problems. Another shortcoming of the majority of these algorithms is the difficulty of ensuring (at least with some high probability) that the solution found is close to the global optimum, that the restart of the computation from another initial point would not result in a better solution. That is why attempts to develop more effective tools to solve the space layout problem continue. Since stochastic search methods such as simulated annealing [2] and genetic algorithms [3] recently have proven to be successful in solving combinatorial optimization problems they have also been extensively used in design and in particular for the space layout problem, for example [4, 5]. Among other advantages is that they offer some theoretical estimates of the global optimality of the solution found.

So far the major emphasis in design applications has been on genetic algorithms due to their special appeal to the design computing community - ease of interpretation with possible analogies with natural systems. We use the extension of genetic algorithms [6] which is based on the concepts of genetic engineering. It enhances the computational process by explicitly evolving and using a hierarchy of evolved genes (genetic patterns) which are responsible for the useful and harmful features of designs. Gene evolution (development of this hierarchy from the original set of elementary genes) proceeds in parallel with the evolution of designs themselves.

2. Space layout planning problem

The space layout problem is concerned with finding the optimal locations for a set of interrelated objects. We use the following canonical form of this problem [1] to find a one-to-one mapping ρ

$$\rho : \{M \rightarrow N\}, \quad j = \rho(i), \quad i \in M, \quad j \in N$$

of the discrete set M with m elements (set of activities, for example office facilities) onto another discrete set N of n elements (set of locations, for example floors of the building where these facilities should be placed), $m \leq n$, such that the overall cost of the layout I is minimal

$$I = \sum_i f_{i\rho(i)} + \sum_i \sum_j q_{ij} c_{\rho(i)\rho(j)} \rightarrow \min$$

where $f_{i,j}$ is the fixed cost of assigning element $i \in M$ to element $j \in N$; q_{ij} is measure of interaction of elements $i, j \in M$; and c_{ij} is measure of distance between elements $i, j \in N$.

Some constraints are usually present in space layout problems which either prohibit some of the placements or impose some extra requirements on the feasible placements. They are usually handled by penalties.

3. Genetic engineering extension of genetic algorithms

Genetic algorithms (GAs) are a family of search methods which can be viewed as computational models of Darwinian evolutionary theory. The search space consists of character strings (chromosomes or genotypes) composed of the elements of a given alphabet (alleles). GAs search for the points in the search space which have optimal fitness. Typically they start by generating a random population of states in the search space (initial generation) then the next generation is reproduced from the initial one using the analogs of evolutionary operators (usually crossover - the swap of the genotype's substrings and mutation - random change of a small fraction of the genes). Then the process proceeds iteratively, each next generation is reproduced from the previous one. The reproduction is done stochastically according to the fitness of the individuals in the current population ("survival of the fittest").

One can also study the dynamic changes in the genotypic structure of the population which take place during the evolution. The corresponding process of genetic evolution can be interpreted as a competition between different combinations of genes (clusters) for dominance in the genetic pool - new combinations of elementary alleles fight their way into genetic pool of the current population and some of the old ones are forced out of it. Some gene combinations become very popular and make up a significant fraction of the genes used to built genotypes of the current population. Some combinations become extremely unpopular and make up a much lower fraction of the genes than the ones in a randomly generated genetic pool. These changes of genetic material during evolution can be viewed as an evolution of the basic genetic primitives, the genetic alphabet, which is used to build the genotypes of the current population. This process of genetic evolution starts with the initial population of genes (genetic alphabet of original alleles) then the next population of genes is bred from the original one where each entity (complex or evolved gene) is a compound of the original alleles (elementary genes), etc. The laws of the evolved genes reproduction are defined implicitly via the evolution of the actual genotypes but it is clear, that reproduction proceeds stochastically proportionally to the ability of evolved genes to generate fit phenotypes (again "survival of the fittest" principle). In this interpretation two interrelated evolutionary processes occur in parallel. The first process replaces the set of original alleles with the emerged sets of more and more complex (in the general case) and more specialized gene structures - the set of evolved genes. The second process replaces the population of individuals whose genotypes are built using the elementary alleles with these new populations whose genotypes are built using the current set of evolved genes. The first process of gene evolution is not as directly controllable in the standard GA as well as in particular GAs which emulate artificial evolution (breeding processes) [7].

The modern practice of genetic engineering provides the means to study and control directly this underlying process of gene evolution in order to short cut the evolutionary path and to manufacture the population with desired properties faster than can be done during the standard evolution process. Essentially, the genetic engineering practice consists of three steps:

(1) the analysis of the current genetic pool in order to identify the beneficial genetic material ("superior" evolved genes);
(2) extra processing of the genetic material of the population which makes it richer in the "superior" evolved genes before the reproduction stage;
(3) it is known that the reproduction operators can disrupt the useful genetic material [3]; hence the reproduction operators have to be modified in order to prevent this from happening or at least make the damage of the "superior" evolved genes which happen to be present in the current pool less likely than it is for the standard evolutionary operators.

Step (1) is usually executed in the following manner, Figure 1: the "super" group (extremely fit, for example, the fittest 10% of the population) and the "sub" group (extremely unfit, for example, 10% of the population which are most unfit) of the individuals are separated from the current population. Then an attempt is made to find the genotypic features (characteristic substring, patterns, etc.) which distinguish the former and the latter groups. This is done using the apparatus of sequential analysis, which has been developed in the areas of genetic engineering, speech recognition and computer science [8, 9, 10, 11, 12, 13, 14]. Assume that the characteristic feature of a highly fit genotype in some problem is a contiguous substring (genetic word) arbitrarily positioned within the genotype, Figure 1. Here genetic analysis includes first making a list of all genetic words which are used in majority of the “super” genotypes. Then those words which are also used in ”sub” genotypes are eliminated from this list. This procedure yields the new “superior” evolved genes. A list of all words which are used some given number of times in a group of sequences can be constructed using suffix trees in linear time on the combined length of all the sequences considered [14]. The characteristic genetic patterns of the “super” group are declared the “superior” evolved genes.

This conception of genetic engineering bears some relation to the concept of ”gene linkage” [16]. However, there are important differences: we seek to locate and articulate those gene sequences which produce the beneficial fitnesses of the phenotypes; and we replace those gene sequences by a new evolved gene, thus changing the alphabet with which the GA works.

Genetic processing from step (2) should be aimed at enriching the genetic pool with the “superior” evolved genes with minimal changes to this pool. It can be done by using techniques derived from genetic engineering:
(1) *gene surgery* - finding the pieces in the genotypes which differ from the “superior” evolved genes only slightly (such that they could be converted into these “superior” evolved genes using not more than some preset number of gene swaps, for example) and replacing them with corresponding “superior” evolved genes.
(2) *gene therapy* - random choice of a “superior” evolved gene and a position within the genotype and execution of the pair-wise gene swappings required to put this evolved gene in this position.

Step (3) consists of a modification of the reproduction operators which explicitly prohibits the damage of “superior” evolved genes. For example, the crossover point can be chosen only within the non-“superior” parts of the parent’s genotypes, the mutation of the “superior” genes is not allowed, etc.

The genetic engineering based GA has the following structure:

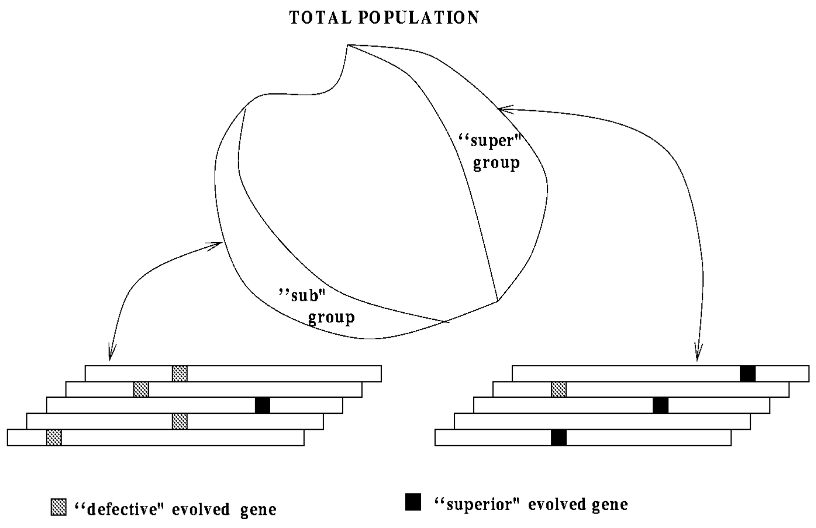


Fig. 1: The identification of the "superior" evolved genes.

(1) Initialization of the population (randomly) and the library (usually empty initially) of the "superior" evolved genes.
(2) (A) Extraction of the "super" and "sub" groups of individuals (extremely fit and extremely unfit) from the current population. For example, it could even be the most fit 10% and the least fit 10% of the population.
(B) Identification of the "superior" evolved genes which distinguish these groups from each other at the genetic level.
(C) Check to determine if the evolved genes which have already been evolved actually distinguish the "super" and "sub" groups from each other.
(D) Update the evolved genes' library by adding the newly evolved ones and eliminating (some) of the ones which test negatively.
(3) Pre-reproduction processing step, defined by the type of the genetic engineering technique employed.
(4) Reproduction using the evolved genes.
(5) If the stop conditions (for example, the given number of generations have been produced or the population has converged, etc.) are not met go to step **(1)**.

The genetic engineering based modification of a GA is a computational model of genetic engineering and serves as an extension of the GA just as genetic engineering can be viewed as an extension of the natural evolution process. Likewise, since this modification provides a direct control of the gene evolution, the overall efficiency of the evolution process at the phenotype

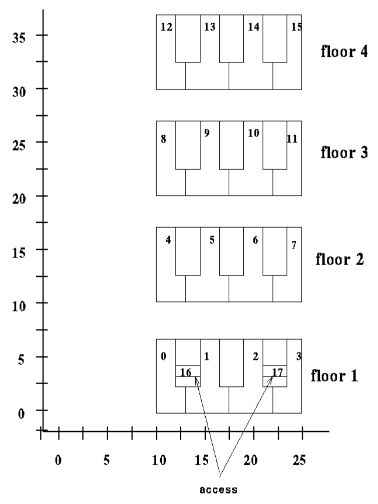


Fig. 2: Zone definition - graphical representation.

level is higher. The ability of the breeder (designer) to produce improved individuals is also higher.

Note, that evolved genes have been described in very general terms as some unspecified characteristic feature of a group of genotypes. Depending on the particular type of genetic encoding it could be a particular genetic substring in a given position (building block in standard GA theory [3]), or some substring in an arbitrary position within the genotype or some structure found within a substring. It could even be a group of genes gathered together, irrespective of the order of these genes within the group (which, as we will demonstrate later, happens to be the case in many layout planning problems).

4. Examples of space layout planning problems

Example 1: Office layout

As the first test example we use the problem of the placement of a set of office departments into a four-floor building [1]. The areas of the 19 activities to be placed (office departments, numbered 0,. . . ,18) are defined in Table 1 in terms of elementary square modules. There is one further activity (number 19) whose location is fixed. The objective of the problem does not have a non-interactive cost term, i.e. $f_{ij} = 0$. The interaction matrix q_{ij}, $i, j = 0, 19$ is given in Table 2. The set of feasible placements is divided into 18 zones numbered from 0 to 17, Figure 2. The areas of these zones are defined in Table 3. The activity number 19 is an access area which has a fixed location - zones numbers 16 and 17, Figure 2. Since the layout cost does

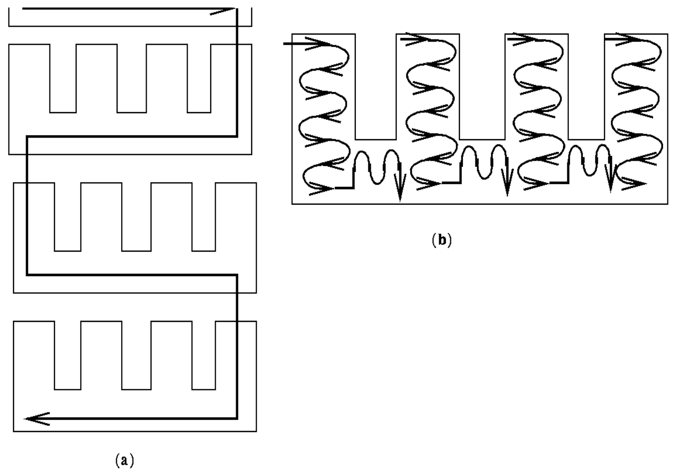


Fig. 3: The pattern used to map the genotype onto the phenotype: (a) The order of between floor mapping and (b) of the mapping within the floor.

not depend on the areas of the zones numbered 16 and 17 or on the area of the activity number 19 they are not shown in Table 3. The matrix showing travel distances between all zones is defined in Table 4. We use the same genetic representation as was used in [4] - the genotype of the problem is a sequence which determines the order in which zones are filled with activity modules. Each zone is filled line by line starting from its highest one and from the leftmost position within it. For example, the genetic sequence $g = \{13, 14, 0, 2, 6, 9, 16, 11, 3, 4, 18, 15, 17, 1, 12, 10, 8, 7, 5\}$ generates a layout plan in the following manner: first, all 18 elementary modules of activity 13, i.e. department 0800 are placed along the path shown in Figure 3, followed by 31 modules of activity 14, i.e. department 0900. Thus, the original gene alphabet of the problem consists of 19 alleles and each of these alleles must be used only once in each valid genotype.

Activity Number	Activity Name	Number of modules
0	Dept. 0210	2
1	Dept. 0211	2
2	Dept. 0220	8
3	Dept. 0230	8
4	Dept. 0240	15
5	Dept. 6815	13
6	Dept. 0300	15
7	Dept. 0400	7
8	Dept. 0500	6
9	Dept. 0600	12
10	Dept. 0700	53
11	Dept. 6300	10
12	Dept. 6881	16
13	Dept. 0800	18
14	Dept. 0900	31
15	Dept. 1000	61
16	Extra module	1
17	Extra module	1
18	Extra module	1

Table 1. The definition of the activities ([1]).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		3	3	2	2	2			3							2				3
1			3	2	2	2			3							2				
2				2	2			3							2					
3					3	2			3							2				
4						2			3							2				3
5									3							2				
6														3		2				
7														3		2				
8																2				3
9															2					
10															3	2				
11															3	2				
12															3	2				3
13																2				
14																2				
15																2				3
16																				
17																				
18																				
19																				

Table 2. Activity interactions matrix ([1]).

zone	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
N Modules	20	22	20	20	18	20	18	18	16	18	16	16	14	16	14	14

Table 3. Zone definition (measured in number of modules) used by Liggett ([1]).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0		4	9	21	27	9	10	23	28	10	11	24	29	11	12	25	30	6	24
1			5	17	23	5	6	19	24	6	7	20	25	7	8	21	26	2	20
2				13	18	6	5	18	23	7	6	19	24	8	7	20	25	2	15
3					5	23	18	5	6	24	19	6	7	25	20	7	8	15	2
4						24	19	6	5	25	20	7	6	26	21	8	7	20	2
5							5	18	23	5	6	19	24	6	7	20	25	3	21
6								13	18	6	5	18	23	7	6	19	24	3	16
7									9	23	18	5	6	24	19	6	7	16	3
8										24	19	6	5	25	20	7	6	21	3
9											5	18	23	5	6	19	24	4	22
10												13	18	6	5	18	23	4	17
11													5	23	18	5	6	22	4
12														24	19	6	5	17	4
13															5	18	23	5	23
14																13	18	5	18
15																	5	18	5
16																		23	5
17																			18
18																			

Table 4. Distance matrix ([1]).

Example 2: Hospital layout

Our second test problem is concerned with finding the optimal assignment of 19 activities (defined in Table 5, labelled $0, \ldots, 18$) of a hospital to 19 possible locations, labelled $1, \ldots, 19$ [15]. Unlike Example 1 this problem has a set of "concentrated" activities. Each of these activities takes only one cell on the grid. The distance and activity interaction matrices are presented in Tables 6 and 7.

activity code	activity function	activity code	activity function
0	Receiving and Recording	10	X-Ray
1	General Practitioner	11	Orthopedic
2	Pharmacy	12	Psychiatric
3	Gynecological & Obstetric	13	Squint
4	Medicine	14	Minor operations
5	Pediatric	15	Minor operations
6	Surgery	16	Dental
7	Ear.Nose & Throat	17	Dental Surgery
8	Urology	18	Dental Prosthetic
9	Laboratory		

Table 5. Activity definitions ([15]).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1		12	36	28	52	44	110	126	94	63	130	102	65	98	132	132	126	120	126
2			24	75	82	75	108	70	124	86	93	106	58	124	161	161	70	64	70
3				47	71	47	110	73	126	71	95	110	46	127	163	163	73	67	73
4					42	34	148	111	160	52	94	148	49	117	104	109	111	105	111
5						42	125	136	102	22	73	125	32	94	130	130	136	130	136
6							148	111	162	52	96	148	49	117	152	152	111	105	111
7								46	46	136	47	30	108	51	79	79	46	47	41
8									69	141	63	46	119	68	121	121	27	24	36
9										102	34	45	84	23	80	80	69	64	51
10											64	118	29	95	131	131	141	135	141
11												47	56	54	94	94	63	46	24
12													100	51	89	89	46	40	36
13														77	113	113	119	113	119
14															79	79	68	62	51
15																10	113	107	119
16																	113	107	119
17																		6	24
18																			12
19																			

Table 6. Distance matrix ([5]).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0		76687		415	545	819	135	1368	819	5630		3432	9082	1503			13732	1368	1783
1			40951	4118	5767	2055	1917	2746	1097	5712				268		1373	268		
2				3848	2524	3213	2072	4225	566			404	9372		972		13538	1368	
3					256					829	128								
4									47	1655	287		42				226		
5										926	161								
6									196	1538	196								
7											301								
8										1954	418								
9												282							
10												1686					226		
11																			
12														42					
13																			
14																99999			
15																			
16																			
17																			
18																			

Table 7. Activity interaction matrix ([5]).

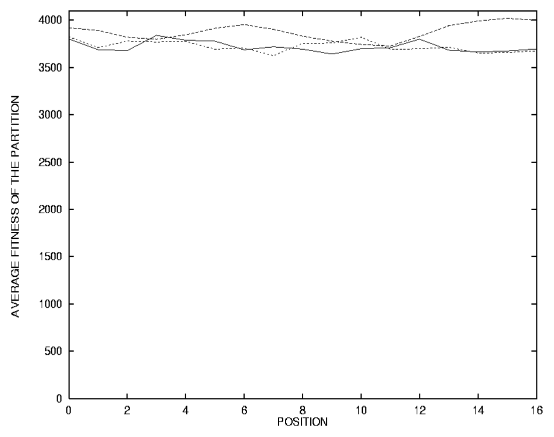


Fig. 4: Characteristic dependencies of average fitness of the defined fixed substrings (randomly generated triplets) on the positions of these substrings within the genotype.

5. Gene evolution in space layout problems

The realization of a genetic engineering based GA for the layout planning problem depends critically on the type of genetic regularities which occur in this type of problem with a chosen genetic encoding. We employ an order-based and position-dependent encoding. That is, in the general case, fitnesses, averaged over the partitions of the search space made of all the genotypes which contain some fixed substring in different positions within the genotype, differ. Therefore, at first sight it looks like the genetic regularity of this problem (its "natural" evolved gene) is a fixed substring in a fixed position within the genotype [6]. Hence, all the computational machinery should be designed based on this notion of the evolved genes as fixed-location fixed substrings, identification should be aimed at finding such fixed substrings in fixed positions, etc. Nevertheless, more detailed analysis of the problem shows that this is not precisely the case here. First, let us plot some characteristic dependencies of average fitness of populations whose genotypes contain some fixed substring (randomly generated) against the position of this substring in the genotype, Figure 4. This is the equivalent of defining a number of activities to be next to each other and treating them as a new "super-activity" which is never disaggregated. This new super-activity can be located anywhere in the layout in the same way as any other activity. We will call such gene groupings compact closed groups. The standard deviation of the average fitnesses of the partitions, defined by the presence of the fixed substring in their genotypes against the fitness of this substring is shown in Figure 5. Figures 4 and 5 correspond to Example 1, but similar dependencies

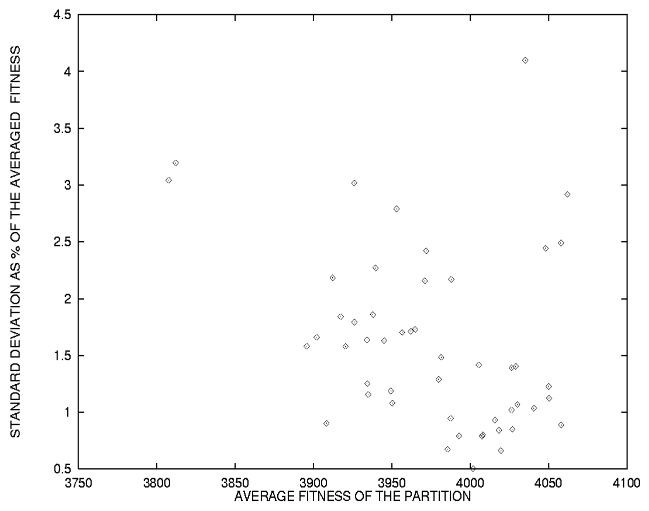


Fig. 5: The dependence of the standard deviation (measured as percentage of the averaged fitness) on the averaged fitnesses of the partitions of the population defined by the fixed substrings in the genotype. The fixed substrings were generated randomly.

are also found for Example 2. These computations as well as all further computations of the averaged variables were performed using the Monte-Carlo method. From these plots it follows that a dependence of the average fitness of the genotypes, which contain a fixed substring, on the position of this substring is almost flat and can be neglected as a first approximation.

An even stronger statement can be made - the characteristic feature of suboptimal genotypes in many layout planning problems is the presence of some genes as compact closed groups. The actual order of the genes within these groups is less significant for the cost of a layout than the presence of such groups in compact forms in genotypes. This phenomenon can be understood if we analyse the part of the overall cost function of Example 1 due to interaction of the activities from such a group. For example, consider the group $\{8,0,1,2,3,4\}$ (which happens to be a "superior" evolved gene of Example 1) and compute the reduced cost function

$$\textbf{reduced cost} = \sum_{i,j=8,0,1,2,3,4} q_{ij} c_{\rho(i)\rho(j)}$$

and the complementary cost

$$\textbf{complementary cost} = I - \textbf{reduced cost}$$

averaged over the genotypes which contain this group of genes in a scattered form, ie not contiguous, as well as over the genotypes which contain this group in a compact form for 6 different random orderings of these genes. The results of these computations are shown in Table 8. Figure 6 shows one such scattered genotype and one compact genotype with their corresponding layouts and costs.

gene group	reduced cost	complementary cost
scattered	1539	2539
$\{8,0,1,2,3,4\}$	607	2540
$\{0,1,2,8,3,4\}$	637	2536
$\{0,1,2,3,8,4\}$	671	2536
$\{4,1,2,8,0,3\}$	729	2510
$\{8,2,1,0,3,4\}$	628	2535
$\{4,2,1,8,0,3\}$	687	2509

Table 8. Average reduced cost of layouts whose genotypes contain gene group $\{2,1,0,8,3\}$ in a compact form with different orderings and in a scattered form for Example 1.

gene group	reduced cost	complementary cost
scattered	27921925	31159923
$\{0,1,2\}$	21682872	31339432
$\{1,0,2\}$	23398516	31335705
$\{2,0,1\}$	21709788	32849239

Table 9. Average reduced cost of layouts whose genotypes contain gene group $\{0,1,2\}$ in a compact form with different orderings and in a scattered form in Example 2.

The results of similar computations for Example 2 and evolved gene $\{0,1,2\}$ are presented in Table 9. Here the reduced cost is defined as

$$\textbf{reduced cost} = \sum_{i,j=0,1,2} q_{ij} c_{\rho(i)\rho(j)}$$

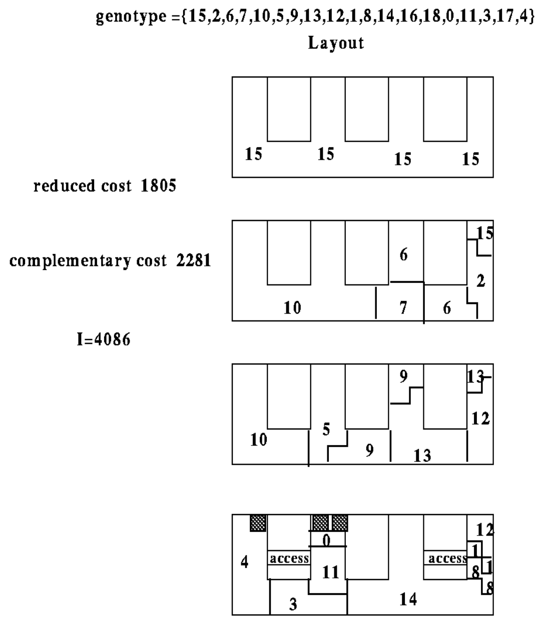
From these results it can be seen that the presence of these groups of genes reduces and possibly minimizes the reduced cost function and that the order of the activities within these groups is a less significant factor. It is clear that the complementary costs of the layouts with scattered and compact forms of these gene groups are approximately the same (that is, the dependence of the complementary part of the cost on the positions of the genes from this group is very weak). In other words, if one wants to find a layout with a smaller reduced cost and approximately the same complementary cost, one should search among layouts whose genotypes contain such genetic groups and which contain the corresponding group of activities placed in a compact spatial group (since our genetic representation is spatially continuous - if corresponding genes are located one after another then corresponding activities are located one after another along the mapping path, Figure 3).

Essentially, this possibility to single out the reduced cost (effectively to decompose the problem partially) is provided by the relatively high level of variation of the components of the matrix product $q_{i,j} \cdot c_{k,l}$ and the presence of a subset in this product whose level of variability is much lower. The weak dependence of the reduced cost function on the order of activities within the group is a consequence of this relatively low level of variations of corresponding components of the product of the matrices $\{q\}$ and $\{c\}$. Hence, in some layout planning problems "natural " evolved genes (naturally occurring genetic regularities) are not fixed substrings but rather groups of genes.

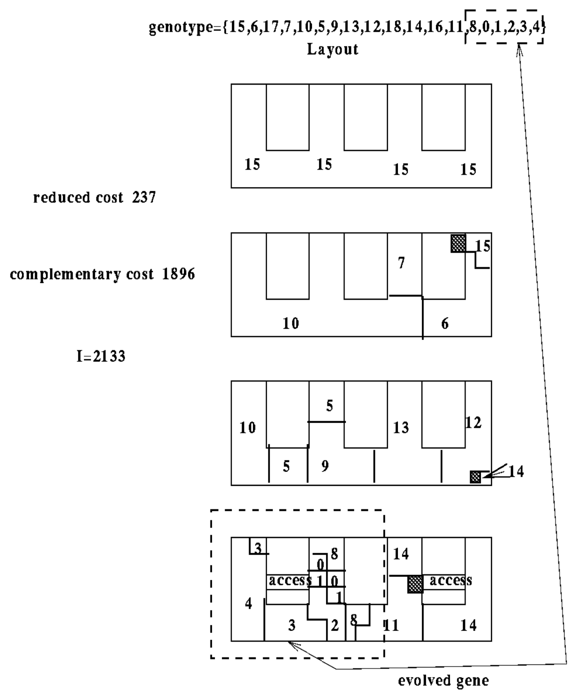
Thus, the optimization of the layout cost can be performed in two steps. First, identify these gene groups and choose their optimal positions with respect to each other (which is the major source of cost savings). This yields an optimization problem with a smaller number of parameters than the length of the initial genotype (instead of the positions of all genes in such group we have just the positions of the whole groups). Second, choose positions of the gene-components (activities) within each of these groups which improve the cost only marginally. It is clear that the standard GA actually performs both searches simultaneously. The search process of the genetic engineering based GA attempts to separate these two processes. Since the effective size of the search space in these two subsearches is smaller than the overall size of the search space of the original layout planning problem it can either be done faster or produce a better performing solution for the same computational expense. It is also clear that the local search in such large-scale spaces of "granulated" cells corresponds to the non-local search in the original state space.

Note that one does not have to establish beforehand the feasibility of evolving the "superior" evolved genes in such a form - one can simply try an algorithm based on a corresponding notion of the "group"-like "evolved genes" and a two-level optimization process. Its success would be a sufficient condition for the existence of such genes. Otherwise the genetic engineering based GA degenerates into a standard GA.

Therefore, the aim of the identification step of the genetic engineering



(a)



(b)

Fig. 6: Genotypes and corresponding layouts with a evolved gene in scattered (a) and compact (b) forms. Filled squares denote dummy activities 16,17 and 18.

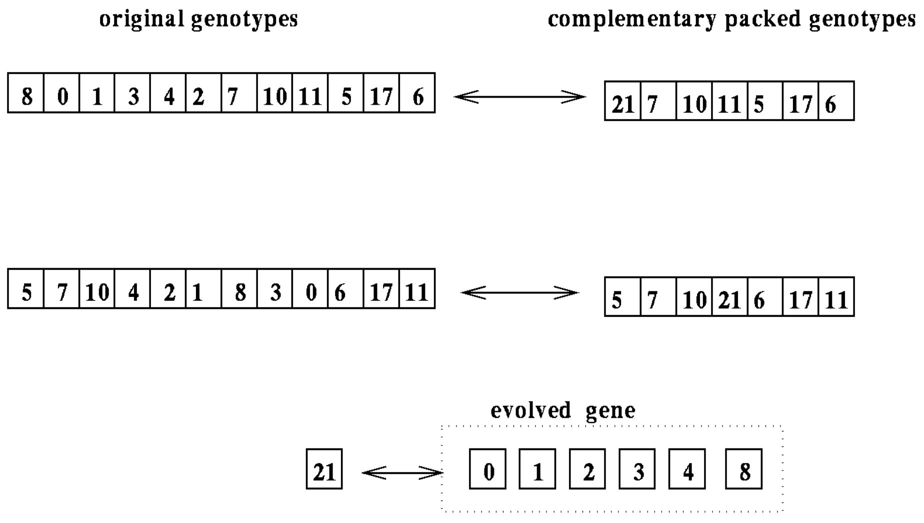


Fig. 7: Transformation of the original genotype into a "packed" form.

algorithm in this problem is to find particular groups of genes (irrespective of their locations in the genotypes and their actual order within the group) whose presence distinguishes the "sub" group. The simplest way to do this is to compute empirical frequencies of the appearance of all compact genetic groups (doublets, triplets, etc.) in the genotypes of the "super" and "sub" groups. Then, such gene groups for which empirical frequencies are higher than the probabilities of the corresponding genetic groups to appear in a random sequences are filtered. Finally, the groups found exclusively in the "super" group are declared as newly evolved "superior" evolved genes. This gives the current set of evolved genes. It is clear that no evolved gene can be present more than once in a valid genotype for this class of problems. It is also possible that one evolved gene can contain a previously evolved gene as its component. This generates a hierarchy of evolved genes.

Since evolved genes in this problem are not simply contiguous pieces of the genotypes we cannot apply the fast identification procedure used in Section 3. Instead we designed a specialized algorithm. It calculates average distances between the positions of all the genes in all the genotypes from the analyzed group. Those genes which are on average close to each in the "super" group and are not close to each other in the "sub" group are declared as new evolved genes.

For each individual layout we introduce a second complementary genotype where the corresponding compact groups of genes are replaced with the new single evolved genes, Figure 7. This yields what we shall call "packed" genotypes. We can view this transformation of the genotype as the replacement of the decomposition of the genotypes on the elementary alleles with its partial decomposition on the evolved genes. Note that the information about particular ordering of the components within any instance of such an evolved gene is still kept by the genetic engineering based GA in the original genotype.

6. Genetic engineering based GA for layout planning problems

The genetic engineering based GA now proceeds in three stages :
Stage 1 - simultaneous evolution of the "superior" evolved genes and minimization of the cost of layout built from these evolved genes.
(1) Fixed-sized population of the original genotypes is initialized randomly. The corresponding population of packed genotypes is produced by just copying the original genotypes.
(2) Standard GA is run. Genetic operators are applied to the packed genotypes (this guarantees the survival of the evolved genes which are currently present in the population). Then the corresponding unpacked genotypes are reconstructed from the packed ones by replacing the evolved genes with their particular realization (particular ordering of their components) in the corresponding parent.
(3) After a fixed number of generations (or after some fixed reduction of cost is achieved) the "super" group of the best 10% of packed genotypes of the population and the "sub" group of the worst 10% of the population are singled out and analysed. The compact gene groups found almost exclusively in the "super" group and almost never in the "sub" group are declared new "superior" evolved genes. The packed form of the current genotypes is reproduced by scanning unpacked genotypes and replacing all the instances of the evolved genes with the single new evolved genes.
(4) Optional step - advanced genetic engineering operation. We use some hybrid of "gene surgery" and "gene therapy" - all the genotypes which do not belong to the "super" subpopulation are updated by moving the components of the corresponding evolved genes together. This is done in a way which minimizes the changes to the genetic material, that is, which can be done using the minimal number of pairwise swappings of the genes. It is clear, that it can also be done by re-initializing the non-"super" part of the population with the randomly generated packed genotypes and then reconstructing evolved genes using random ordering of their components. (5) Repeat steps (1), (2) and (3) until the population converges (or until there

is no progress during some predefined number of generations).
Stage 2 - checking the optimality of the evolved set of "superior" evolved genes.
(1) The goal of this step is to relax the restriction on the genotypes of the population imposed by the current set of evolved genes and to check that the evolution without evolved genes is not productive. This is achieved by replacing the genotypes of the population except its "super" subgroup with the random sequences of the initial alleles and by forming the corresponding set of the packed genotypes by copying the original genotypes. Then a predefined number (chosen empirically) of evolution steps are executed. If there is no improvement then we assume that the evolved set of genes is the potentially optimal one. Otherwise we check what part of the evolved genes are still the "superior" evolved genes of the population. If we end up with the same evolved genes then again we assume that the current set of the evolved genes is the optimal one. Otherwise we return to the first stage with the current set of "superior" evolved genes and continue evolution of the layouts and further evolution of the evolved genes.
Stage 3 - evolving optimal ordering of the facilities within "superior" evolved genes.
(1) At this stage the population consists of the copies of the same packed genotype. Crossover and mutation operate here on the corresponding "superior" evolved genes. That is, after two parents are chosen then instances of the same evolved genes which are present in these two parents are crossed with each other with some probability and the resulting instances of the evolved genes are mutated. In other words when we cross two genotypes we actually cross and mutate the corresponding evolved genes.
(2) The evolution proceeds until some predefined number of steps is performed or until the population converges.

We use the standard one-point crossover operator - two parental genotypes are picked randomly based on their relative level of fitness from the current population then a crossover point is chosen at random in both genotypes where they are cut and finally genotypes of two children are produced by combining the first part of the first parent's genotype and the second part of the second parent's genotype and vice versa.

7. Numerical results

Example 1: Office layout

The result of the run of the standard GA which converges to the best

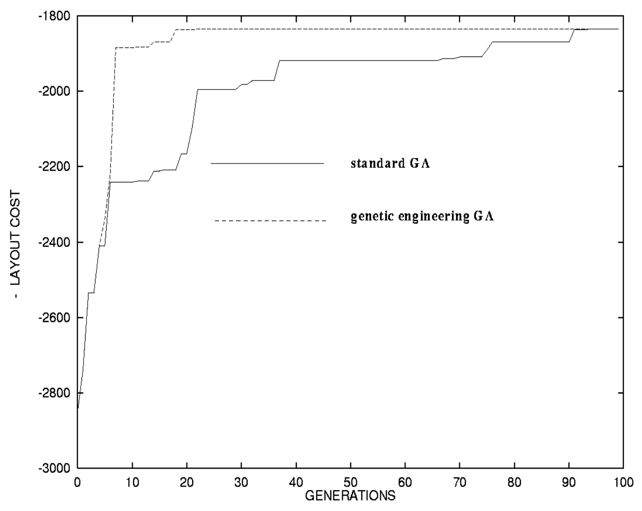


Fig. 8: The best fitness vs. generation number for the standard GA and genetic engineering based GA in Example 1. The results are averaged over 10 runs with different initial random seeds, which converge to the best solution found.

solution for Example 1 (averaged over 10 such runs from different initial random seeds) is presented in Figure 8 (bold line). The population size was 200, probability of crossover 0.6 and the probability of mutation 0.01. We use elitist GA with a generation gap of 3. On average, it was necessary to run the GA 13 times in order to get the solution $\{7, 6, 13, 17, 14, 12, 15, 5, 18, 11, 10, 9, 0, 1, 2, 8, 3, 16, 4\}$ with the cost of the corresponding layout of 1834. Although we use elitist GA with different parameters and a different realization of the crossover operation the results are essentially the same as the output of the EDGE system [4]. The analysis of the evolutionary path of the GA shows that the search process consists of a first stage (5-10 generations) when the crossover serves as the major constructive tool and when non-local search takes place and of a second stage when pair-wise swapping of the genes is the driving force of a search. During the first stage the algorithm finds the point which belongs to the basin of attraction of one of the minima. During the second stage (about 90 generations) crossover does not make any contribution to the search, the actual improvement is due to mutations only and the algorithm actually searches locally within the basin of attraction just found.

The results of the genetic engineering based GA are presented in Figure 8 with the dotted lines. Two "superior" evolved genes were identified after 5 generations: $\{0, 1, 2, 8, 3, 4\}$ and $\{12, 14\}$. After generation 10 two further genes evolved, $\{6, 13\}$ and $\{15, 12, 14\}$, which includes the previously evolved

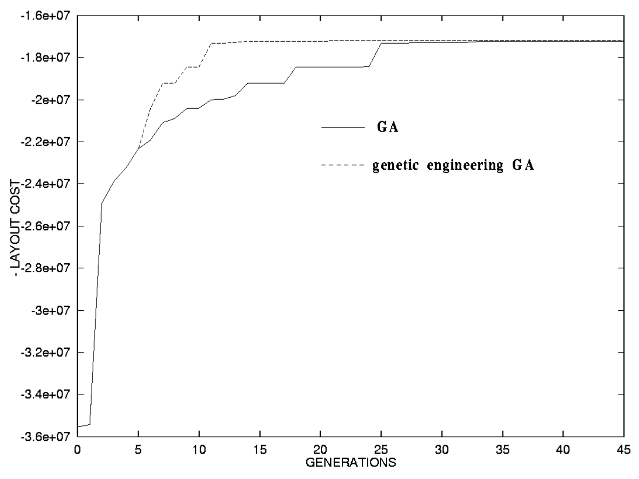


Fig. 9: The best fitness vs. generation number for the standard GA and genetic engineering based GA in Example 2. The results are averaged over 10 runs with different initial random seeds, which converge to the best solution found.

gene $\{12, 14\}$. On average just two runs were required with different initial seeds to find the layout with the cost of 1834. It turns out that both versions of the advanced genetic engineering operations described in stage (3) of the algorithm produce essentially the same computational results. The computational cost of genetic engineering analysis was approximately one tenth of the computational cost of one generation of the GA (probably because the genotype-layout mapping is very expensive computationally here).

Let us note that our results (as well as the results presented in [4]) are not directly comparable with the results of [1], since we omitted the penalty associated with splitting activities between floors.

Example 2: Hospital layout

The results of the numerical simulations of the second test problem are presented in Figure 9. The bold line denotes the results of using the standard GA. The dotted line denotes the results of the genetic engineering based GA. On average it took 26 runs of the standard GA from different initial random seeds to find the point $\{8, 9, 6, 18, 13, 17, 12, 16, 5, 10, 3, 4, 11, 7, 14, 15, 0, 1, 2\}$ with the cost 17212548. The genetic engineering based GA needed on average only three runs with different initial seeds in order to find this layout. The corresponding dependence of the best layout cost on the generation number, averaged over 10 successful runs, is shown in Figure 9 with the dotted line. Three genes evolved: $\{0, 1, 2\}$ after generation 2, $\{14, 15\}$ after generation 7 and $\{8, 9, 6\}$ after generation 4. The computational cost of genetic analysis

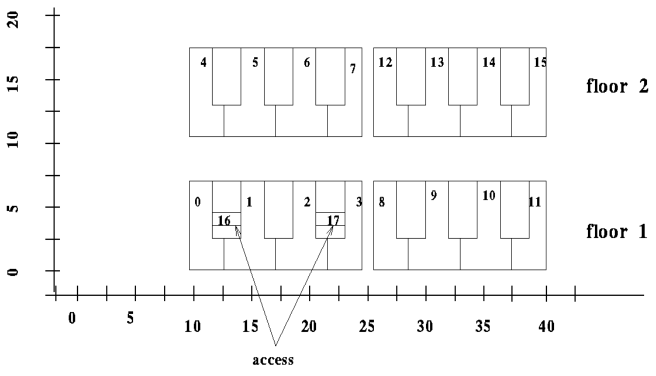


Fig. 10: Modified floor locations from Example 1 - graphical representation.

here was approximately one half of the cost of production of one generation by the GA (because the genotype here is actually a layout).

Genetic engineering based GAs allow larger areas of the entire search space to be explored than could be done by a standard GA subject to the same computational expense. The computational savings in terms of generations are of the order of 90% (although some extra computations are needed for additional gene analysis and processing these are a relatively small persentage of the computational cost of a single generation of the standard GA).

8. Using evolved genes to solve families of layout planning problems

In many cases one has to solve a number of similar layout planning problems (for example, to place essentially the same set of activities into different locations, etc). Unfortunately, it is very difficult to use information about optimal layouts already designed to generate solution for a similar problem.

The major advantage of the genetic engineering based GA is the possibility to re-use the evolved genes (pure information about optimal layouts) in a family of similar problems. This possibility is based on an intuitively obvious assumption that in many layout planning problems some activities "gravitate" to each other much more strongly than they are attracted to the rest of the activities and therefore should be placed as a compact spatial group in any suboptimal layout. If the possible locations (distance matrix) is changed then one still has to place such activities in a compact spatial group (although the actual physical placement could be quite different). Even the changes to the activity's interaction matrix could preserve the attraction with respect to the ones outside it do not change.

Example 3: Modified office layout
In order to check this intuition we first modified the problem in Example 1, by changing the geometry of the system to be that shown in Figure 10. This yields the distance matrix shown in Table 10.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0		4	9	21	27	9	10	23	28	39	44	57	62	40	45	58	63	6	24
1			5	17	23	5	6	19	24	35	40	53	58	36	41	54	59	2	20
2				13	18	6	5	18	23	30	35	48	53	31	36	49	54	2	15
3					5	23	18	5	6	18	23	36	41	19	24	37	42	15	2
4						24	19	6	5	13	18	31	36	14	19	32	37	20	2
5							5	18	23	40	45	58	63	30	36	49	55	3	21
6								13	18	35	40	53	58	25	31	44	50	3	16
7									9	22	27	40	45	12	23	31	37	16	3
8										13	18	31	36	5	10	23	28	21	3
9											5	18	23	5	10	23	28	4	22
10												13	18	10	5	18	23	4	17
11													5	23	18	5	10	22	4
12														28	23	10	5	17	4
13															5	18	23	5	23
14																13	18	5	18
15																	5	18	5
16																		23	5
17																			18
18																			

Table 10. The modified distance matrix for Example 3, which corresponds to the floor locations shown in Figure 10.

The activity interaction matrix remains the same. The results of the optimization are shown in Figure 11. We run the standard GA and genetic engineering based GA beginning with the empty set of "superior" evolved genes (dotted line) and from the previously evolved ones (dotted line). We can see that the re-use of the evolved genes yields about a 20% savings in terms of the number of generations.

Example 4: Modified hospital layout
The simulations were also carried out for the second example with a modified distance matrix (the average change of its component was about 6%), Table 11. The results (standard GA, genetic engineering based GA with an empty initial set of evolved genes and genetic engineering based GA with the set of evolved genes that was evolved for the Example 2) are shown in Figure 12. Again, we can see computational savings when re-using the previously evolved genes.

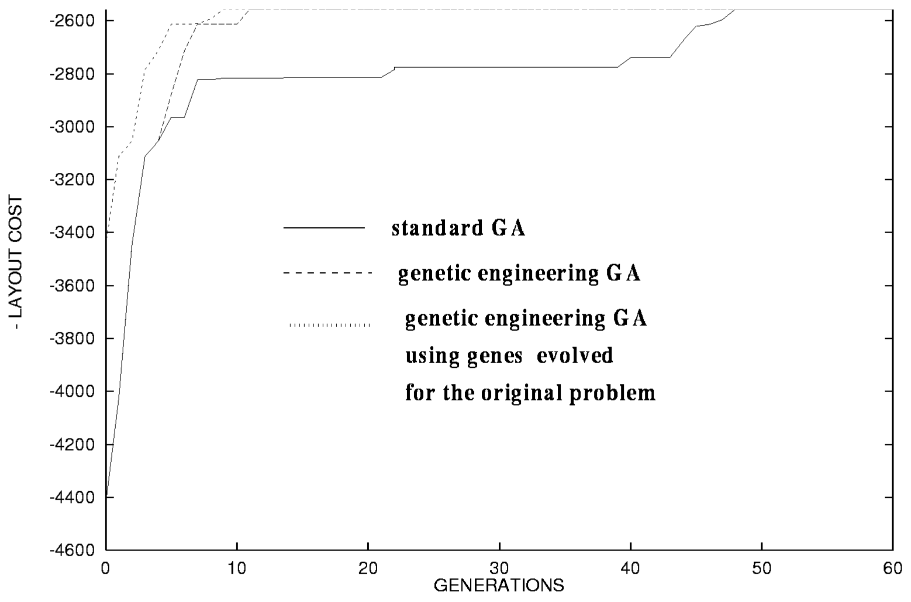


Fig. 11: The best fitness vs. generation number for the standard GA and genetic engineering based GA in Example 3. The results are averaged over 10 runs with different initial random seeds, which converge to the best solution found.

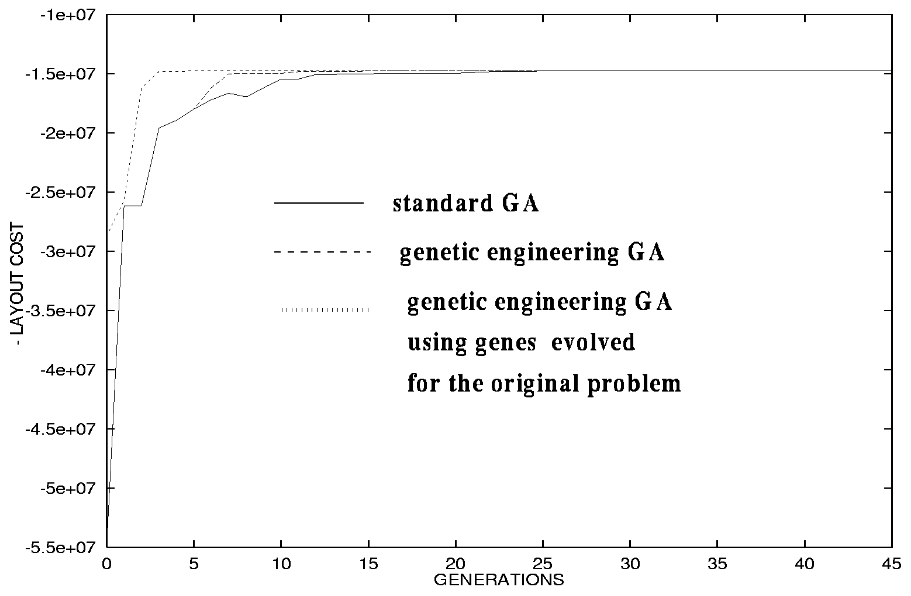


Fig. 12: The best fitness vs. generation number for the standard GA and genetic engineering based GA in Example 4. The results are averaged over 10 runs with different initial random seeds, which converge to the best solution found.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1		12	36	24	52	44	110	125	89	63	125	101	65	91	123	124	126	120	126
2			24	71	72	75	108	68	118	81	93	106	58	124	151	154	67	61	70
3				47	70	43	110	73	117	65	95	107	46	121	155	163	73	67	68
4					42	34	142	102	160	51	94	146	49	109	102	105	104	105	111
5						42	125	126	102	13	73	115	30	94	125	127	136	120	136
6							148	111	162	46	96	148	49	115	145	152	111	99	111
7								46	46	136	39	30	108	51	72	79	46	47	31
8									59	138	55	46	111	68	121	119	19	24	29
9										102	34	45	77	20	74	80	65	63	44
10											54	118	29	95	131	124	141	129	134
11												47	49	47	85	94	63	42	24
12													100	51	89	89	46	39	27
13														71	113	113	109	113	119
14															69	79	68	62	51
15																3	113	106	116
16																	109	107	119
17																		5	24
18																			12
19																			

Table 11. Modified distance matrix for Example 4.

9. Conclusions

Layout planning remains an important task in architectural design. It has been formulated as a mathematical optimization problem. However, this has been shown to be an NP-complete problem in its complexity, which explains why it is so difficult to obtain optimal or near optimal solutions. The approach presented here has some intuitive appeal for designers because of the direct mapping between the genes and their expression in a layout. It has a further appeal since the evolved genes readily map on to groups of activities which need to be located next to or near each other. Such information is hard to see intuitively in a large scale problem.

Another advantage of this approach which has not been seen in other approaches is the ability to re-use the evolved genes in related problems. The evolved genes represent problem-specific knowledge that has been learned by the system which is now externalized and available for re-use in similar problems. In conclusion, we have developed a genetic engineering based extension of GAs and applied it to layout planning problems. We demonstrated that in a number of layout planning problems the solution process can be roughly divided in two stages. During the first stage some activities are aggregated into compact groups and the objective is to find the optimal placement of such groups. During the second stage the optimal location of the activities within these groups is sought. The first stage corresponds primarily to non-local search in the original search space (since even pair-wise permutation of two aggregated activities corresponds to the macro step in the placement space of non-aggregated activities). The second stage sooner or later becomes a local search. We demonstrated significant computational advantages of such an extension. We have also shown that such optimal (natural) aggregation of activities can be used in a range of similar problems with significant computational benefits.

10. Acknowledgments

This work is directly supported by a grant from Australian Research Council, computing resources are provided through the Key Centre of Design Computing.

References

[1] Liggett, R.S.: 1985, Optimal spatial arrangement as a quadratic assignment problem, in Gero, J. S. (ed.), *Design Optimization*, Academic Press, New York: 1-40.

[2] Kirkpatrick, S., Gelatt, C. G. and Vecchi, M. 1983: Optimization by simulated annealing, *Science* **220** (4598): 671–680.

[3] Holland, J. 1975: *Adaptation in Natural and Artificial Systems*, University of Michigan, Ann Arbor.

[4] Jo, J. H. and Gero, J. S. 1995: Space layout planning using an evolutionary approach, *Architectural Science Review*, **36** (1): 37-46.

[5] Wilhelm, M. R. and Ward T.L. 1987: Solving quadratic assignement problems by simulated annealing, *IEE Transactions*, **19** (1): 107-119.

[6] Gero, J. S. and Kazakov, V. 1995: Evolving building blocks for genetic algorithms using genetic engineering. *Proceedings of the IEEE Conference on Evolutionary Computing*: 340-345.

[7] Muhlenbein, H. and Schlierkamp- Voosen, D. 1994: The science of breeding and its application to the breeder genetic algorithm BGA, *Evolutionary Computation*, **1** (4): 335-360.

[8] Sankoff, D. and Kruskal, J.B. eds. 1983: *Time Warps, String and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, MA.

[9] Collins, J.F. and Coulson, A.F.E. 1987: Molecular sequence comparison and alignment, in *Nucleic Acid and Protein Sequence Analysis: A Practical Approach* IRL Press, Washington DC: 323-358.

[10] Schuler, G.D., Altschul S.F. and Lipman D.J. 1991: A workbench for multiple alignment construction and analysis, in *PROTEINS:Structure, Function, and Genetics*, **9**: 180-190.

[11] Needleman, S.B. and Wunsch, C.D. 1970 : A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology*, **48**: 443-453.

[12] Karlin, S., Dembo, A., Kawabata, T. 1990: Methods for assessing the statistical significance of molecular sequence features by using general scor-

ing scheme, *Proceedings of the National Academy of Science U.S.A.*, **87**: 5509-5513.

[13] Karlin, S., Dembo, A., Kawabata, T. 1990: Statistical composition of the high-scoring segments from molecular sequences, *Annals of Statistics*, **18**: 571-581.

[14] Crochemore, M. 1994: *Text Algorithms*, Oxford Univesity Press, New York.

[15] Harik, G. R., Goldberg, D. E. 1996: Learning Linkage, *IlliGAL Report No. 96006.*

[16] Elshafei, A. N. 1977: Hospital layout as a quadratic layout problem, *Operations Research Quarterly*, **28**: 167-179.

[0] This is the copy of the following: Gero, J. S. and Kazakov, V. (1997). Learning and reusing information in space layout problems using genetic engineering, *Artificial Intelligence in Engineering* **11**(3): 329-334.