CHAPTER 4

GENETIC ALGORITHMS FOR MULTIOBJECTIVE STRUCTURAL OPTIMIZATION

Abstract: Automated structural design optimization procedures rely heavily on numerical algorithms that lead the search process towards improved solutions in terms of better merit objective values. Many of the traditional optimization algorithms are problem-dependent and single-objective oriented that usually make use of gradient information to guide the search process and continuous design variables are often assumed. In contrast, population-based natural evolution-inspired genetic algorithms (GAs) are general-purpose numerical tools, with which gradients are no longer needed and discrete-valued design variables can be handled without any difficulty. More importantly, multiple conflicting objective functions can be directly and simultaneously treated by GA through the concept of non-dominance. This chapter discusses features of different optimization algorithms with a particular focus on techniques of solving multiobjective optimization problems via GAs, followed by description of multiobjective GA schemes for the numerical examples in this study. A literature review of application of GAs to design optimization of civil structural systems is presented at the end.

4.1 Traditional structural optimization algorithms

4.1.1 Overview

Traditional optimization techniques usually apply available mathematical programming algorithms to solve structural design optimization problems (Gallagher and Zienkiewicz 1973; Arora 1989). Optimality criteria are obtained through application of Kuhn-Tucker conditions combined with Lagrange multipliers accounting for relevant design constraints. With the linear programming approach, both the objective(s) and constraints must be linear functions of design

variables. In contrast, nonlinear programming approaches deal with problems whose objective(s) and/or constraints are nonlinear differentiable functions of design variables. Note that traditional methods are usually problem-dependent in nature, that is, one method that is very efficient for a kind of problems may not be viable for another kind. It is therefore necessary for one to have knowledge of applicability of one particular optimization algorithm in order to use it properly.

Continuous design variables are often assumed in traditional optimization algorithms. To handle optimal design problems with discrete or mixed continuous-discrete design variables that are generally encountered in real-world design practice, other numerical techniques are needed, including branch and bound method, integer programming, sequential linearization, rounding-off method, etc. (Arora 2002). With traditional methods, the optimization process usually starts with a single initial design elected in the feasible design space and the subsequent improved designs evolve based on searching/updating rules that typically require sensitivity calculation. This point-by-point process may be trapped at local minima.

4.1.2 Application to multiobjective optimization

Most of traditional methods by themselves can only handle single-objective based optimization problems, for which a single final optimal solution will be obtained. In order to solve optimization problems with multiple (conflicting) objective functions, one has to convert, with nontrivial inconvenience, the original multiobjective problem into a series of equivalent single-objective optimization problems, which in total produce a set of solutions that exhibits optimal tradeoff among all competing objectives. There are two basic approaches commonly used in structural optimization.

One approach is the "$\varepsilon-$constraint method" that keeps only one objective at a time and converts the other objectives into constraints; solving these single-objective problems for

different values of constraints leads to a set of optimal tradeoff designs. The difficulty of this method lies in determination of constraint values, especially the upper and lower limits. Fu and Frangopol (1990a; 1990b) used this approach to optimize both truss and frame structures.

The other approach is the "weighted sum method" that forms a single composite objective function as a weighted sum of the original multiple objectives, using a specified set of weight coefficients. Solving this composite single-objective problem leads to one of the optimal tradeoff designs. Sarma and Adeli (2000) used this method for cost optimization of steel structures. All possible sets of weight coefficients are necessary in order to obtain a complete distribution of optimal tradeoff designs. Because traditional optimization methods usually search for the optimized solution on a point-by-point basis, multiple algorithm runs are needed, each run at best generating one of those optimized tradeoff solutions.

It needs to be pointed out that, although globally optimal tradeoff designs represent the best possible solutions for a multiobjective optimization problem, excessive computational efforts may be required to locate all of them provided local optimum is not an issue. In a design practice, however, a set of improved tradeoff designs intermediately obtained could be readily acceptable if they are satisfactorily better than the set of initial tradeoff designs. As will be discussed in Section 4.2, a numerical algorithm using evolutionary computation will be selected in this study as the optimization tool that updates the tradeoff designs generation by generation. This evolution process can be terminated when designers are satisfied with the set of tradeoff solutions at a particular generation. Compared to (global) *optimal* solutions, these intermediate results should be rigorously referred to as *optimized* solutions. In the following study, however, this distinction may not necessarily be made and the two terms are used interchangeably.

4.2 Genetic algorithms

4.2.1 Overview

A brief introduction is provided in this section to the working mechanism of genetic algorithms (GAs), which will be used to solve the posed design optimization problems of steel SMRF structures throughout this study. GA is global stochastic search and optimization heuristics that mimic natural evolution and are based on Darwin's survival-of-the-fittest principles (Holland 1975). GA belongs to the category of nature-inspired evolutionary algorithms that include as well ant-colony optimization, simulated evolution, DNA computing, and cultural algorithms (Deb 2001). Since their inception in the 1960's, GAs have been successfully used in a wide array of applications (Goldberg 1989). The growing popularity stems from GA's ease of implementation and robust performance for difficult engineering and science problems of vastly different natures.

GA usually operates on solutions that are encoded as *genotypic* representations (e.g., binary strings), called *chromosomes*, from their original *phenotypic* representations (i.e., actual data values), although there also exist real-coded GAs that work directly on original data with no coding operation (Coello et al. 2002). GAs start with a set of initial solutions (*population*) that is randomly generated in the search space. For each solution in the current population, objective functions defining the optimization problem are evaluated and a *fitness* value is assigned to reflect its (relative) merit standing in the population.

Based on the fitness values, GA performs a *selection* operation that reproduces a set of solutions that have higher fitness values from the previous *generation* to fill a mating pool. A *crossover* operation is then pursued with which two *parent* solutions in the mating pool are randomly selected and interchange, with a prescribed crossover probability, their respective

string components at randomly selected bit locations referred to as *cross sites*. The resulting new solutions are called *children* or *offspring*. This step is meant to hopefully combine better attributes from the parent solutions so that child solutions with improved merits could be created. The next operator in GA is *mutation* that changes the genotype value at one or more randomly selected bit locations in a child solution with a pre-defined mutation probability. This operation serves to possibly recover useful information that is inaccessible through selection and crossover operations and therefore encourages search into the completely new solution space. After these three basic operations, a new generation is created. The search process continues until prescribed stopping criteria are met, for example, the maximum generation number is reached or negligible improvement of the optimized solutions is found.

Compared to most of the traditional optimization methods, GA has the following distinct advantages (Goldberg 1989): (1) GA works with a population of solutions instead of one solution at each iteration. By starting with a random set of solutions, the current population evolves to an offspring population at each iteration. Working with a number of solutions provides GAs with the ability of capturing multiple optimized tradeoff solutions in a single algorithm run. (2) Gradients are not required in guiding the genetic search, which makes GA insusceptible to pitfalls of traditional gradient-based hill-climbing searching procedures. This feature is especially useful for nonlinear structures where either objectives or constraints or both are non-differentiable with respect to discrete-valued design variables. One such example is the steel frame design problem where possible steel member sections are usually selected from commercially available standard discrete wide-flange shape databases. (3) GA is a problem-independent universal search tool, which makes it more flexible as well as robust for application to problems of different natures; the requirement on users' capability is greatly reduced.

The most significant concern about GA is its extensive computational expenses. A sufficiently large population of solutions needs to be maintained in order to override the impact of errors due to stochastic transition operators. This entails a tremendous amount of computational time for evaluating objective functions as well as for performing genetic operations. Consequently, it is helpful to decide, before a GA based procedure is formally launched, if the benefits gained from GAs are worth the extra computational efforts spent. With the availability of high-speed computer facilities, however, computational demands might not be a very big obstacle for problems of small to moderate sizes.

4.2.2 Multiobjective optimization via genetic algorithms

General remarks

Multiobjective GAs have been comprehensively studied and fruitfully developed in the last decade (Coello 2003). Unlike problems with single objective functions for which one optimized solution is sought after, there is no unique optimized solution with respect to all conflicting objective functions for multiobjective optimization problems (unless all these objectives are strictly equivalent to one another). Instead, a set of optimized tradeoff solutions is expected among conflicting objectives. GA searches for these multiple optimized tradeoff solutions based on the concept of dominance.

A solution x_i is said to *dominate* another solution x_j, if two conditions are both met (Deb 2001): (1) x_i is no worse than x_j in all objectives; (2) x_i is strictly better than x_j in at least one objective. If either of these two conditions is violated, x_i does not dominate x_j. Among a set of solutions P, the *nondominated set* P' contains those that are not dominated by any individual of the set P. When the set P is the entire search space, the resulting nondominated set is called the

(global) Pareto optimal set (or *front*). Note that the generation-wise nondominated set is referred to as the *optimized tradeoff set* throughout this study.

A successful multiobjective optimization algorithm must have the ability: (1) to obtain a nondominated set of solutions close to the global Pareto optimal front, and (2) to have this solution set as diverse as possible, that is, to prevent solution clustering from occurring (Deb 2001). Note that the selection operation is based on the relative fitness measures of solutions. Unlike single-objective problems where the objective function itself may be used as the fitness measure (possibly with scaling and constraint-handling treatment), a multiobjective optimization algorithm needs a single fitness measure that reflects the overall merit of multiple objectives. A qualified fitness assignment strategy should fulfill the two-fold goals as stated above. Another important issue is how to handle constraints in an efficient way such that useful information from infeasible (i.e., constraint-violating) solutions can be appropriately integrated into the search process towards the Pareto optimal set. These will be discussed in the following text.

Fitness assignment strategies

A useful fitness measure should encourage the nondominated set to converge to the true global Pareto optimal front as well as to keep diversity among solutions, i.e., to spread out evenly over objective regions of interest. The key to this problem is that fitness values should not only reflect the solutions' relative distances to the universal Pareto front, but also penalize those solutions that tend to cluster. There exist two general types of strategies to assign fitness values: the rank-based and the non-rank-based.

Rank-based strategies

Goldberg (1989) proposed a nondominated sorting technique to rank solutions in a population based on the concept of domination. For the population of one generation, a nondominated subset is first identified according to the definition of Pareto optimality: a solution is dominated if there exists another solution in the generation that is better in at least one objective and no worse in all other objectives. All solutions in this subset are assigned a rank of one and are then temporarily deleted from the population. The nondominated subset of the remaining solutions is identified and assigned a rank of two. This procedure continues until all solutions in the population have been assigned ranks. A solution in a lower-ranked front will be assigned a higher fitness than that of a solution in a higher-ranked front. Therefore, solutions that are closer to the global Pareto optimal front have higher fitness values. As an illustration, consider a generic problem that has two objectives to be minimized. Figure 4.1 indicates, in the solution space, ten solutions that are classified into four fronts with varied ranks.

For solutions of the same rank, criteria must be made to identify the fitter solutions. Goldberg (1989) suggested a *niching* strategy that penalizes those clustering solutions by reducing their fitness values, that is, preferences are given to solutions that are less crowded by other solutions (usually of the same rank only). Because solutions with higher fitness measures will have larger probabilities of being selected at the reproduction stage, the nondominated solutions are now encouraged to spread over a wider region so that an evenly distributed tradeoff solution set is obtained that is close to the true Pareto optimal front.

Based on Goldberg's idea of nondominated ranking and niching techniques, several versions of multiobjective genetic algorithms were developed that differ mainly in the specific way fitness is assigned to each solution. Different niching strategies defined either in the design variable

space or in the objective space are used to maintain diversity among solutions (Deb 2001). Some representative studies are briefly described as follows.

In the work by Fonesca and Fleming (1995), rank is assigned solution-wise: solutions in the best front are assigned with rank equal to one; rank of any of other solutions in the population is equal to one plus the number of solutions that dominate it. For solutions with the same rank, niching is defined in terms of objective function values. The selection operation is performed directly using rank information of solutions.

Srinivas and Deb (1994) discriminated solutions with the same rank using a "niche count", which is computed as the number of solutions in the population within a predefined distance σ_{share} from a particular solution. The original fitness of a solution is shared evenly by its neighboring solutions within σ_{share}; a solution with a smaller niche count is assigned a higher fitness value. The choice of σ_{share}, however, is not very well defined and largely based on experience.

To circumvent this inconvenience, Deb et al. (2002) proposed a parameter-free niching strategy in which relative fitness of solutions with the same rank are determined by a "crowding distance" measure, which is taken as an average distance of the two solution points on either side of a particular solution point along each of the objectives and thus serves as an estimate of the density of solutions surrounding a particular solution in the population. As illustrated in Figure 4.2, the crowding distance of the *i*-th solution in its own front (marked with dark circles) is the average side-length of the cuboid (shown with a gray box). Higher fitness values will then be assigned to solutions that are more apart from their neighboring solutions in terms of crowding distances.

Non-rank-based strategies

The above rank-based strategies assign fitness with two basic steps: (1) a sequence of ranks in the population is identified and then (2) solutions with the same rank are discriminated. There exist other strategies that assign fitness directly based on the definition of Pareto optimality. For example, Balling (2000) proposed a "maximin fitness function" that takes the form, assuming all objectives are subject to minimization:

$$fitness_i = 1 - \max_{j \neq i} \left(\min_k \left(f_{ki} - f_{kj} \right) \right) \tag{4.1}$$

where f_{ki} represents the *k*-th normalized objective function value of the *i*-th solution in a population. Solutions with calculated fitness values greater than 1.0 belong to the nondominated set. It was claimed (Balling 2000) that Equation 4.1 produces decreased fitness values for solutions tend to cluster and consequently a well-distributed population evolves.

A closer look at this fitness definition reveals some interesting insights. The term $\min_k \left(f_{ki} - f_{kj} \right)$ identifies the most promising non-dominant objective value of the *i*-th solution against all objective values of the *j*-th solution; $\max_{j \neq i} (\cdot)$ then pinpoints the solution that makes the non-dominance property of the *i*-th solution suffer the most. In other words, the fitness value of a nondominated solution depends on the solution that is the closest to it, in terms of the smallest difference among all objective functions; the fitness value of a dominated solution depends on its distance, in terms of the individual objectives, to the dominating solutions in the nondominated set. As a result, the calculated fitness values properly address the relative merits of solutions that form the (best) nondominated set. For solutions that are not on the nondominated front, the calculated fitness values roughly reflect the distance of a dominated solution to the dominating

solution(s) in the nondominated set, which is somewhat similar to rank information. Clustering issues among dominated solutions are not explicitly addressed.

Comparison

In summary, rank-based fitness assignment strategies treat solutions rank-wise; within the same rank, solutions occupying sparser space are generally preferred. In the Balling's fitness assignment strategy, however, no ranks are explicitly identified except for the nondominated set, i.e., rank of one; fitness values of dominated solutions are roughly determined by their relative distances to the nondominated set; clustering among dominated solutions is not explicitly penalized and therefore it has no direct effects on fitness values of dominated solutions.

Divergence-preserving for dominated solutions, however, is not as crucial as for nondominated solution set. As pointed out by Deb (2001), a large portion of population will usually be occupied by nondominated solutions as generation number increases, which is especially the case when the number of conflicting objectives is large and continuous in nature. Therefore, the final nondominated solutions might not be affected appreciably by ignoring divergence among dominated solutions. The distinctive feature of Balling's strategy is that the calculated fitness values directly reflect distance of dominated solutions to the nondominated set, which might help to improve the quality of nondominated solutions. With the rank-based strategies, however, solutions of the same rank may not necessarily have the similar distances to the nondominated set.

Constraint-handling strategies

Constraints define the feasible solution space for an optimization problem. For civil structural designs, constraints may come from code provisions, additional performance

requirements, construction considerations, and so on. All optimized tradeoff designs resulting from a multiobjective optimization procedure have to be valid designs, i.e., they do not violate any established constraints. During the searching process, constraint-violating designs may occasionally appear in the population due to crossover and/or mutation operations. The simplest way to handle invalid designs is to discard such design solutions once detected and keep only feasible designs alive in the population. The obvious drawback of this "death penalty" approach is that useful genetic information possessed by these infeasible designs is completely lost. As a result, the nondominated designs are pushed to the true optimal designs with the knowledge from the feasible space only. To appropriately make use of information from both feasible and infeasible spaces, more efficient constraint-handling techniques are necessary.

For GA based multiobjective optimization problems, constraint-violating designs may be assigned with dummy fitness values, which are always less than those of valid solutions in the population. These dummy fitness values are defined in terms of degree of violations for all relevant constraints. Thus the original constrained optimization problems are essentially converted into unconstrained problems. In an alternative way, constraint-violation may be considered by modifying genetic operators instead of assigning fictitious fitness values to invalid solutions.

Deb (2001) proposed a "constrained binary tournament selection" scheme that, when two solutions are randomly selected from the population, determines the better (fitter) solution based on three rules: (1) if both solutions are feasible, the one with higher fitness wins; (2) if one solution is feasible and the other is infeasible, the feasible solution always wins; (3) if two solutions are both infeasible, the one with less constraint violations wins. A comprehensive survey on constraint-handling techniques for GAs can be found in Coello (2002).

Elitist strategies

The term *elitists* in GAs denotes for the best possible solution(s) in a generation. In multiobjective GAs, elitists are usually the generation-wise nondominated solutions, i.e., those with rank of one. It is beneficial to favorably retain elitist solutions in the subsequent generations for evolution operations due to their excellent genetic properties. The stochastic nature of GAs, however, may disturb this ideal situation especially at early generations when the number of elitists is much smaller than the population size. To solve this problem, the elitist strategy may be used: nondominated solutions (elitists) from the last generation are forcibly inserted back to the present population that is generated by genetic operations (selection, cross, and mutation). A new set of elitists is then identified based on the updated population containing the previous elitists.

4.3 Genetic algorithm for the present study

A steel SMRF design is encoded in the present GA as a two-portion string of pointers into two tables of commercially available standard hot-rolled steel wide-flange sections commonly used for column and beam members, respectively. Appendix B provides the database that contains all these standard steel section types, which constitutes a subset of member sections listed in AISC (1994) specifications. Figure 4.3 illustrates the construction of the GA string representation for the example SMRF described in Section 3.2. There are in total 11 design variables (6 for columns and 5 for beams) for this present design problem.

GA starts with a set of initial structural designs that are randomly generated in the design space from an exhaustive combination of different section types, one from the column section table and the other from the beam section table. Note that many of these $80 \times 154 = 12{,}320$ different initial trial designs may not satisfy code criteria.

Many GAs work with fixed a population size. As generations evolve, the nondominated solutions fill the majority of solution slots in a population, and it may become very difficult for dominated designs to enter the population for genetic operations. As a result, diversity of nondominated designs in the subsequent generations may not be fully explored due to lack of information from valid yet dominated alternative designs (Deb 2001). In this study, a variable sized population is used, comprising 1000 solutions plus the nondominated set from the previous generation (elitism).

In addition, fitness is determined according to Goldberg's nondominated sorting plus Deb's crowding distance measure; Deb's constrained binary tournament selection scheme is adopted; a two-point crossover is applied with a probability of 50%, one crossover being in each portion of the string representation; mutation is performed with a probability of 30% by randomly selecting and perturbing one pointer anywhere in the string representation. Although this is a relatively high rate of mutation, by using elitism to preserve the nondominated designs at each generation, mutation tends not to be very disruptive; sometimes a high level of mutation is used to avoid premature convergence.

4.4 Structural design optimization via genetic algorithms

4.4.1 Overview

GAs have been powerfully applied to a broad range of research areas in structural engineering due to their salient convenience over traditional optimization/search methods as previously stated: problem-independency that requires no gradient information to guide the search process, the global perspective, and easy treatment of discrete design variables. The first structural application of GA was made by Goldberg and Samtani (1986) who studied structural optimization with a ten-bar truss. Afterwards, researchers around the world applied GAs to many

different structural systems. For example, Rajeev and Krishnamoorthy (1992) investigated GA based optimal design of a 160-bar transmission tower system; Adeli and Cheng (1993) used GAs to optimize large high-rise framing systems. GAs were also developed to optimize trusses by simultaneously considering size, shape, and topology (Rajan 1995; Shrestha and Ghaboussi 1998). In parallel to application, comparison studies were performed that recommended more efficient GA operators especially suited for structural optimization applications; for example, Leite and Topping (1998) provided an extensive discussion on a wide range of GA applications as well as performance comparisons with other optimization methods. Ghaboussi (2001) reviewed structural mechanics and engineering applications of biologically inspired soft computing methods (GAs and artificial neural network).

Applications of GA to steel moment frame designs have also received research attention. Member sizes of a practical steel frame can usually be selected from a catalog of commercially available sections only, which leads to optimization problems that are discrete and thus highly non-continuous, making gradient information very difficult to compute. An intuitive remedy is to first use continuous design variable assumptions and then round the design variables of the resulting solution to the nearest discrete values, respectively. This approximate approach, however, by no means guarantees a valid solution, let alone an optimized one. In contrast, GAs have no difficulty in solving these discrete optimization problems.

Camp et al. (1998) developed an automated algorithm that integrates GA, a finite element analysis program, and a complete database of AISC sections for the design of steel frames according to AISC-ASD provisions. Pezeshk et al. (2000) incorporated into GAs a group selection scheme, an adaptive crossover scheme, and a finite element analysis to design frames according to AISC-LRFD as well as AISC-ASD. Foley and Schinler (2003) used an advanced

analysis and object-oriented evolutionary computation for optimizing both fully and partially restrained steel frame designs.

4.4.2 Genetic algorithm based multiobjective structural design

Applications of GA to structural design problems considering multiple merit objective functions have appeared only very recently in literature. Cheng and Li (1997) developed a GA based multiobjective procedure with an elitist strategy through niching and a "Pareto-set filter" technique that alleviate the effects of "genetic drift" or loss of diversity among final solutions; dominance characteristics and constraint violations were handled by a fuzzy-logic penalty function; three truss design examples were provided. Cheng et al. (1999) discussed the possibility of applying GAs to multiobjective optimization for life cycle cost analysis of building systems. Using the concept of min-max optimum, Coello and Christiansen (2000) proposed their GA-based multiobjective optimization technique and applied it to two truss design problems. Greiner et al. (2001) used a nondominated sorting GA with an elitist strategy for a multiobjective problem that minimized both the number of different cross-section types (the largest number being 4) and the mass for design of frame structures. Cheng (2002) summarized his previous research on multiobjective structural optimization emphasizing GAs as well as game theory; objective functions used in the numerical examples included structural weight, control performance index, seismic input energy, potential energy, and construction cost.

To the best of author's knowledge, GA based multiobjective optimization for seismic design of steel moment frame structures has not been studied comprehensively in the literature. This is largely because that (1) multiobjective GAs have been fruitfully developed only recently and (2) practical merit objectives other than the commonly cited structural material weight (cost) have

not yet been formulated/applied in a design optimization environment for the emerging performance-based as well as life cycle cost oriented seismic structural design problems.

The main task of this Ph.D. study is to investigate application of the most recently developed GA techniques to optimized seismic design of steel moment frame structures while considering multiple conflicting objective functions. Three numerical examples will be provided in the following three chapters to demonstrate the significance of the present research.

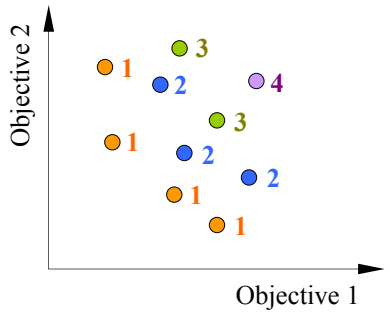


Figure 4.1 Population ranking based on nondominated sorting

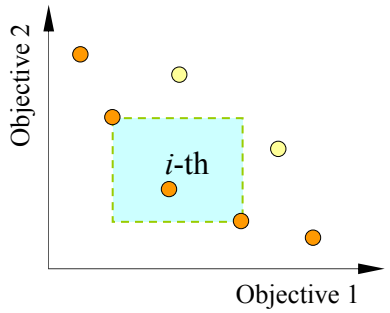
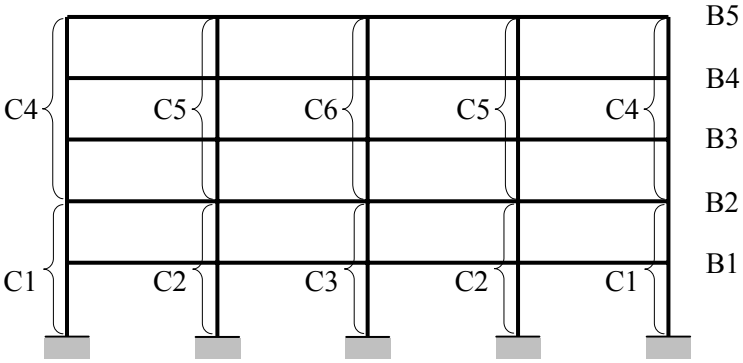


Figure 4.2 Crowding distance calculation

Original Design



Pointer Index (PI_C)	Column (C)
1	W6X9
…	…
69	W14X311
…	…
80	W14X730

Pointer Index (PI_B)	Beam (B)
1	W6X9
…	…
124	W33X130
…	…
154	W36X300

Encoded Design in GA

PI_{C1}	PI_{C2}	PI_{C3}	PI_{C4}	PI_{C5}	PI_{C6}	PI_{B1}	PI_{B2}	PI_{B3}	PI_{B4}	PI_{B5}

Figure 4.3 String representation of the example SMRF design in the present GA