6 EXPERIMENT: GLOBAL SEARCH

6.1 Background

Experiments conducted in the previous chapter show that the novel object-based software design representation, when combined with associated genetic operators that make use of self-adapting mutation probabilities, provides an effective foundation for single-objective local search. At this point however, it is now necessary to build on these findings and investigate ways in which global search might facilitate collaborative designer / computer interaction. Indeed, to facilitate effective search and exploration from an interactive viewpoint, it is useful and practical to shape early lifecycle software design search into two complimentary elements. The first element is global search of the overall search space to narrow the search and focus on localised zones of interest; the second element is local search within the identified zones.

The number of classes in a software design is significant in a number of ways. Firstly, there exists an inherent tension between the number of classes in an early lifecycle software design and the coupling between the classes. On the one hand, the higher the number of classes in a design, the smaller the number of attributes and methods per class, thus the external coupling between classes tends be high. On the other hand, if the number of classes in a software design is low, the number of attributes and methods per class may be high, and so external coupling between classes is potentially lower. Indeed, taken to its logical extreme, if a software design is comprised of only one class, then the external coupling is zero. Thus the number of classes in a software design can be used as a fitness function in global multi-objective search, trading-off, for example, external coupling, or average design class cohesion of classes.

The number of classes in a design is important in another way too. Consistent with the discrete object-based representation, the number of classes in a software design is discrete. Therefore, the number of classes may be used to as the basis of localised zone formulation. For example, all designs of, say, five classes, provide the basis of a localised 'zone', wherein all designs possess five classes.

With regard to multi-objective search and exploration, the elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) proposed by Deb (2001) has inspired the global search algorithm used in this thesis. NSGA-II has been applied to a variety of multi-objective problem domains and in an overview of multi-objective evolutionary algorithms (MOEAs), Coello Coello *et al.* (2007) report that NSGA-II has been used frequently in comparison with newly designed MOEAs. Inspired by the approach of NSGA-II, an outline of the algorithm used in experimentation is as follows. An offspring population of the same size as the parent population is produced by mutation (self-adaptive mutation having been found in the previous chapter to be more explorative than crossover and computationally more efficient). Using selected fitness functions, the combined population is non-domination sorted into ordered "fronts" of equivalent optimality. The new population is filled by solutions of different non-dominated fronts, one at a time; the filling starts with the best front, and continues with the next best and so on until the population size is met or exceeded. Solutions that do not make the new population are discarded. Typically, the last front contains more solutions than there is space available. In selecting which solutions go forward, a 'crowding distance sorting' operator is employed to ensure that the most diverse range of solutions is preserved, thus helping search exploration. In the early stages of evolution, many fronts are evident in the population. However, if evolution proceeds effectively, the number of fronts decreases until eventually a small number of fronts, hopefully including the Pareto-optimal front, remains. At this point, the crowding distance sorting operator is crucial is preserving diversity of solutions. As Deb (2001) points out, the Pareto-optimal front may or may not be continuous, even for multi-objective search problems using real-valued representations, because of the nature of the problem and any constraints that might apply. As well as discontinuous Pareto-optimal front solutions, Deb describes "knee solutions" where the Pareto-optimal front presents as a right-angle. The non-dominated sorting evolutionary algorithm is illustrated in figure 6.1.

6.2 Methodology

Global search parameters have been tuned empirically for the non-dominating sorting evolutionary algorithm based on trial and error and are as follows:

- *selection:* no parent selection performed (i.e. random uniform parent selection);
- *crossover and mutation probabilities:* (0.0, 1.0) (i.e. no crossover performed);
- *offspring creation and replacement strategy:* (100 + 100) replacement (i.e. the least dominated 100 individuals out of the combined population of 200 go forward to the next generation).

With respect to the global search termination condition, the evolutionary algorithm

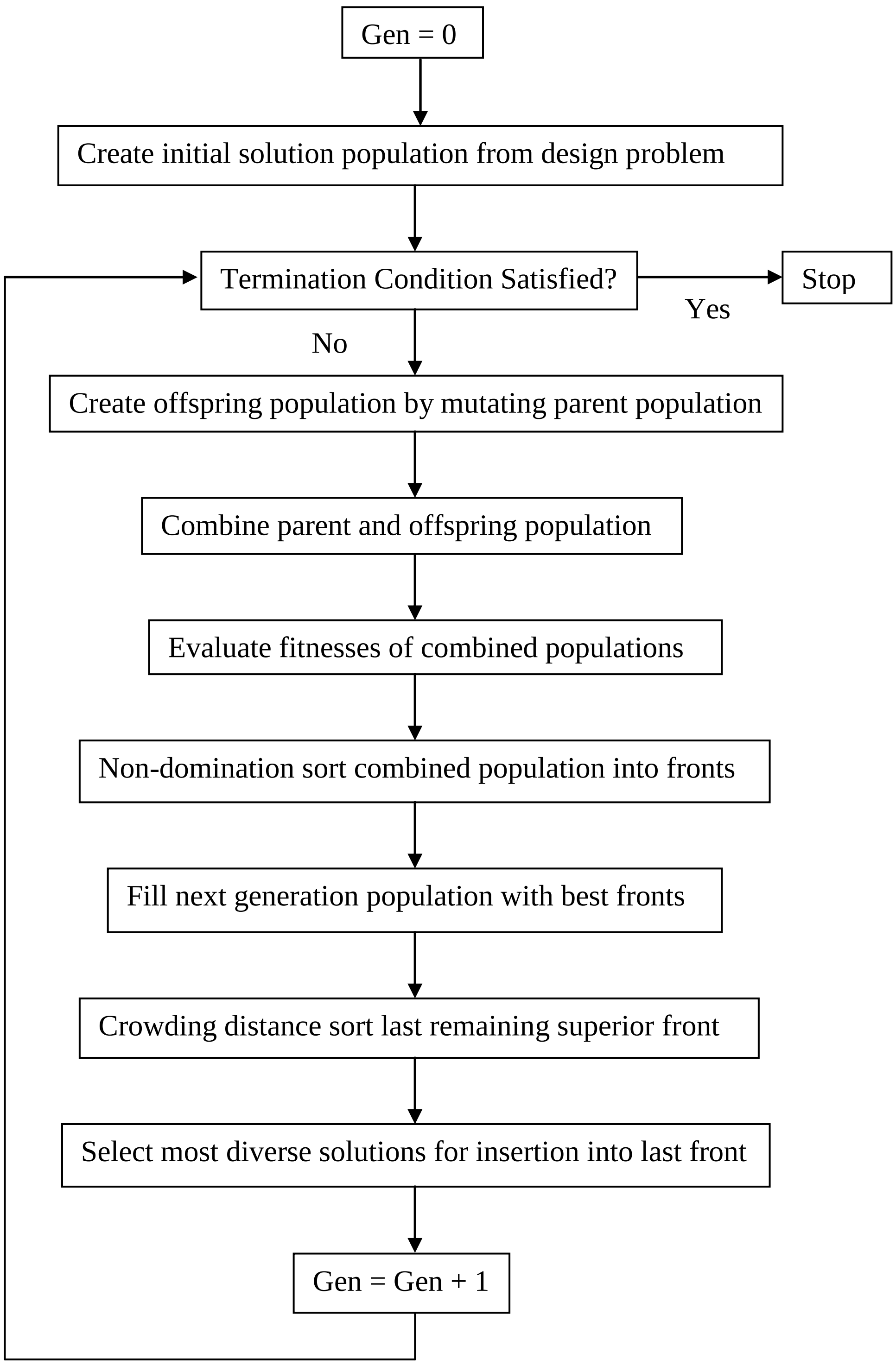
Gen = 0
Create initial solution population from design problem
Termination Condition Satisfied?
Yes
Stop
No
Create offspring population by mutating parent population
Combine parent and offspring population
Evaluate fitnesses of combined populations
Non-domination sort combined population into fronts
Fill next generation population with best fronts
Crowding distance sort last remaining superior front
Select most diverse solutions for insertion into last front
Gen = Gen + 1

Figure 6.1. Non-dominated sorting evolutionary algorithm

is allowed to proceed until it is clear that the search is no longer producing design solutions which dominate all previous design solutions. The algorithm is thus allowed to progress in order to fully observe its overall performance characteristics. With regard to fitness functions, building on findings of local search experiments, external design coupling and the number of classes are selected. The Cinema Booking System is initially selected as the example software design problem under test. Lastly, a *Global Search Agent*, whose single task is to perform the elitist non-dominated sorting algorithm, is implemented to conduct the global search.

6.3 Results

Results obtained from a typical single run of global search performed by the Global Search Agent are illustrated in figures 6.2 to 6.5.

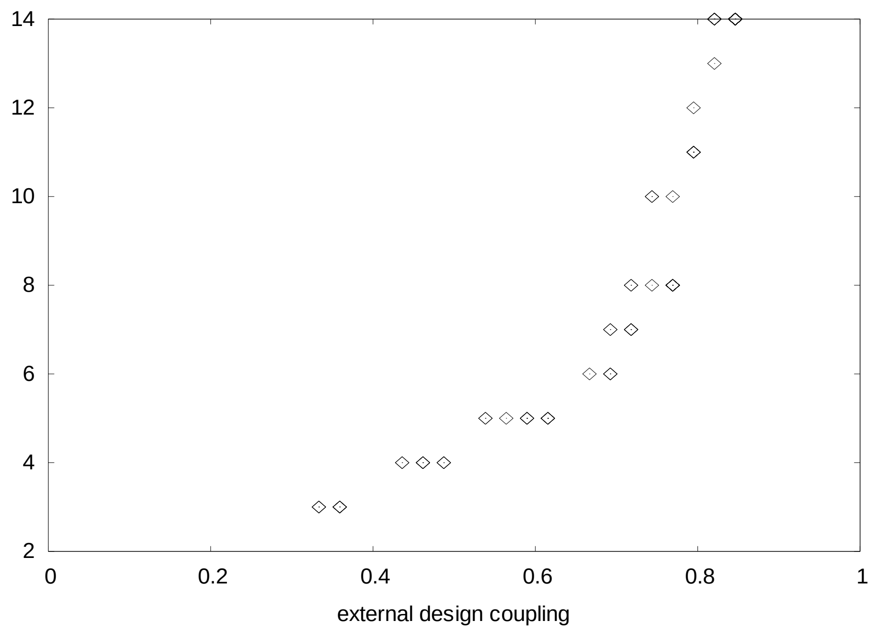

Figure 6.2. Typical Single Run Population after 50 Generations

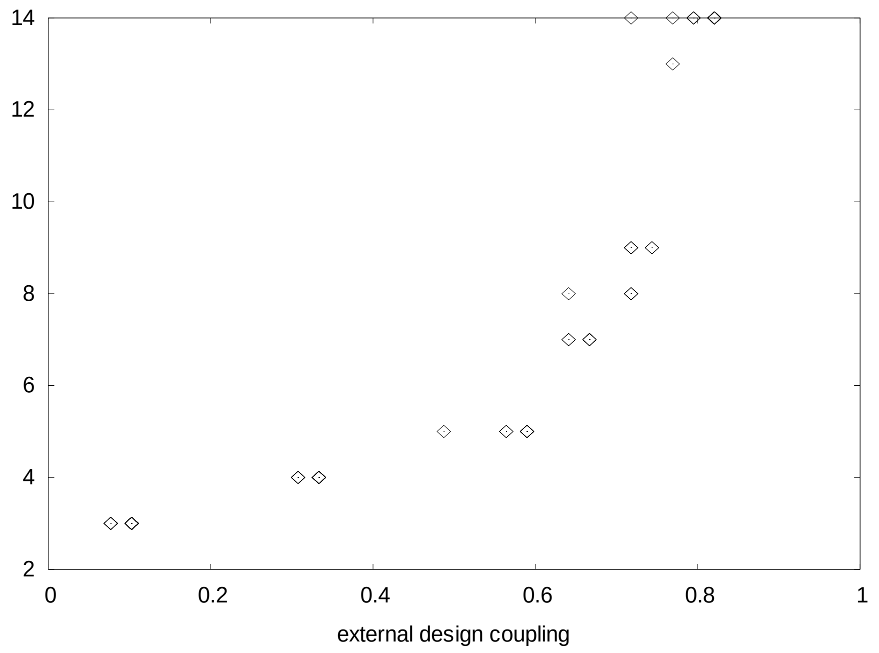

Figure 6.3. Typical Single Run Population after 100 generations

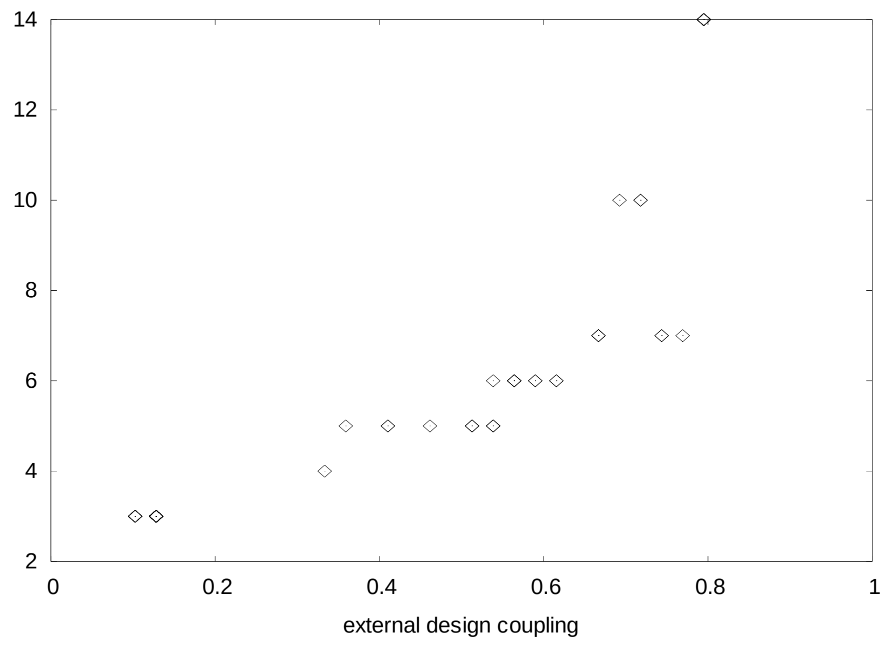

Figure 6.4. Typical Single Run Population after 200 generations

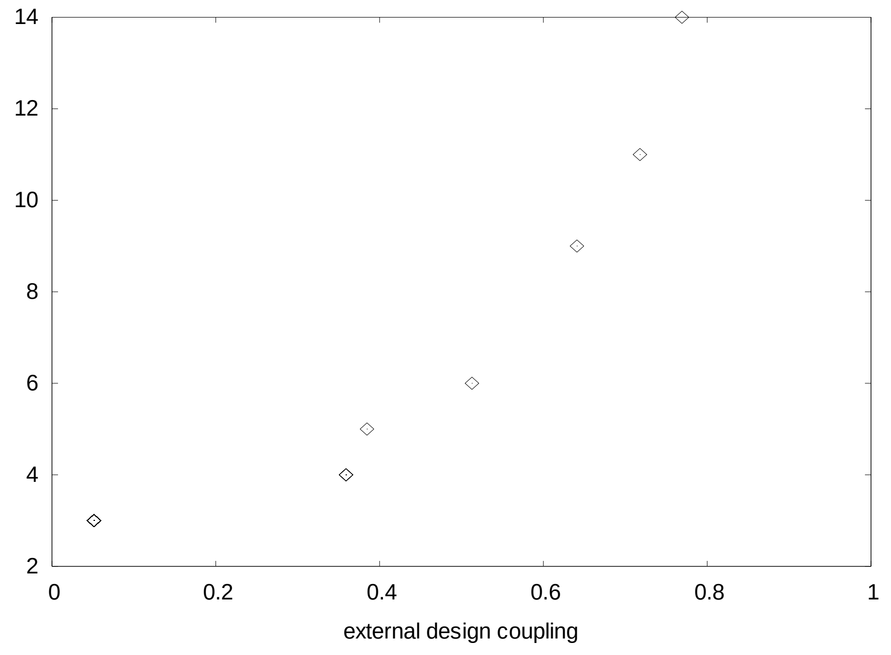

Figure 6.5. Typical Single Run Population after 300 generations

6.4 Analysis

In figures 6.2 to 6.5, the trade-off curve between design coupling and numbers of classes produced by the global search can clearly be seen. An effective increase in the population fitness with respect to the minimization of coupling is evident as the search proceeds, particularly in the first 200 generations. However, an increasing loss of population diversity is also observed as search progresses. Indeed, figures 6.4 and 6.5

reveal that although there is some increase in coupling fitness, the major effect of progressing from 200 to 300 generations is a loss of diversity in the population. (Results for 400 and 500 generations are not shown for brevity as they are very similar to results obtained for 300 generations.)

It is clear that ‘banding’ of designs occurs due to the discrete scale of the *y*-axis. Effectively, this banding phenomenon partitions the global search space into localised zones of designs that share the same number of classes. While this provides a useful mechanism to narrow the search of the global design space, a further consequence however is that diversity in fronts of designs (including the Pareto-optimal front) is effectively constrained to the range of integer values on the *y*-axis. In the search illustrated in the figures 6.2 to 6.5, the maximum number of classes is defined by the bounds on the example design problem specification. Thus diversity with respect to the number of classes is restricted to 14 localised zones. Furthermore, the non-dominated sorting mechanism for selection of individual designs for the next generation may result in individuals from particular zones being eliminated from the population, producing gaps in the fronts of non-dominated designs.

Of course, multi-objective search and optimisation techniques using multi-objective evolutionary algorithms have been successfully applied to a variety of problem situations where the goal is to find multiple trade-off optimal solutions with a wide range of continuous values for objectives (see Deb, 2001, Coello Coello *et al.*, 2007). In such applications, for example, fronts of trade-off optimal solutions for two conflicting objective fitness functions are visualized as 2D graphs where *x* and *y* axes use continuous scales, thus a diverse range of solutions of equivalent optimality may exist within the Pareto-optimal front[1]. However, for the discrete object-based representation of software designs, although the *x*-axis uses a continuous scale to reveal design coupling values, the *y*-axis uses a discrete scale to reveal the number of classes in a class design. As the evolutionary search progresses, therefore, the use of a discrete scale leads to the possibility of reduced diversity among class designs in fronts of equivalent optimality.

Therefore an important trade-off is apparent: as global search progresses through generations, overall population fitness (i.e. external coupling) improves, but at the

[1] For example, multi-objective allocation of discrete resources in constrained environments can be seen in numerous industrial scheduling problems such as production and resource scheduling, timetabling, etc. – see Coello Coello *et al.*, 2007, pp.416-214. However, while the resources being allocated are discrete, the values of the objective fitness functions are continuous.

expense of potential gaps in the fronts of non-dominated designs. It is clear that allowing the global search to progress too far before termination is undesirable as a basis for interactive search, since useful and interesting design solutions may be lost.

It is interesting to speculate on the role of the evolutionary algorithm in the loss of population diversity above. As shown in figure 6.1, after non-domination sorting of the combined population, the next generation is filled with the best fronts. Crowding distance sorting is performed on the last remaining superior front when more design solutions exist than there is space available in the next generation. This is logical but nevertheless crowding distance sorting is only performed on the last remaining superior front. It may be possible that the use of an elitist archive in an archive-based multi-objective evolutionary algorithm e.g. Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999), Pareto Archived Evolution Strategy (PAES) (Knowles and Corne, 2000) could help to preserve diversity. Having said that, it also seems possible that the performance of an archive-based approach inspired by SPEA and PAES might be in some way limited by discrete values for objective fitness functions.

Based on this analysis of the results, it is concluded that computational support would be of great benefit in monitoring this trade-off during global search. Indeed, it seems logical that a software agent could monitor the utility of the population with respect to the coupling versus diversity trade-off as evolution proceeds. When the trade-off becomes unfavourable, the agent then halts global search. Using a task-focussed agent to provide such support is also consistent with the overall research aim of facilitating efficient collaborative designer / computer interaction, in this case by shielding the designer from having to perform an otherwise potentially error-prone manual task. Therefore, a *Zone Isolator Agent* is proposed and its effectiveness assessed in the remainder of this chapter.

6.5 Zone Isolator Agent

A widely used approach to directing the task-focussed behaviour of a software agent is to assign values (or utilities) to the various environmental states that the agent is monitoring (Wooldridge, 2009). Such environmental states may be discrete or continuous. In the case of global search of software designs, the Zone Isolator Agent assesses the utility of a population as search proceeds with respect to the fitness / diversity trade-off. Population utility is measured as a continuous value and depends on such factors as population fitness, sparsity and take-over, together with a longer-term reflection of performance of the evolutionary algorithm trajectory over time. Population utility at a given generation is thus computed using the four measures below:

- *Fitness*, i.e. average population coupling;
- *Sparsity*, i.e. the count of zones that have lost all design solutions divided by the maximum number of classes in the design solution search space;
- *Take over*, i.e. the number of individual designs with one or more duplicates divided by the population size;
- *Elapsed time*, i.e. the current generation number divided by the number of generations specified for the evolutionary run.

For a maximisation function such as class cohesion, utility (*u*) of the population at a generation is computed as:

$$u = fitness - sparsity - take\ over - elapsed\ time \qquad (6.1)$$

On the other hand, for a minimisation function such as design coupling, utility (*u*) of the population at a generation is computed as:

$$u = (1 - fitness) - sparsity - take\ over - elapsed\ time \qquad (6.2)$$

Because it is not known at this point how to scale the different components of the utility calculation, experiments are conducted into the balance between the 'positive' component (fitness) versus the 'negative' components (sparsity, take over, elapsed time).

During global search, should the utility of the population be judged to be sufficiently poor, the Zone Isolator Agent communicates with the Global Search Agent to terminate search. Thus with regard to the Zone Isolator Agent halting global search, three tactics have been formulated for trial as follows:

- when the population utility becomes negative, or
- when a single fall in utility is observed, or
- when two sequential falls in utility are observed.

6.6 Methodology

For consistency with the previous experiment in this chapter, global search parameters are unchanged, i.e.

- *selection:* no parent selection performed (i.e. random uniform parent selection);
- *crossover and mutation probabilities:* (0.0, 1.0) (i.e. no crossover performed);

- *offspring creation and replacement strategy:* (100 + 100) replacement (i.e. the least dominated 100 individuals out of the combined population of 200 go forward to the next generation).

The experimental approach used to investigate the performance of the Zone Isolator Agent is as follows:

- firstly, global search is evolved to 500 generations with various scaling factors to determine a balanced scaling of the components of population utility, then
- secondly, trial the three halting tactics to determine which is the most effective at achieving a judicious termination of global search as the basis for interactive search.

The Cinema Booking System and Select Cruises are used as example design problems. Global search is conducted over 50 runs; average population utility and standard deviation are calculated.

6.7 Findings and Analysis

Figure 6.6 shows the results of average utility of a population of class designs and standard deviation for the Cinema Booking System example problem; figure 6.7 shows average population utility and standard deviation for Select Cruises. All average population utility values are taken over 50 global multi-objective search runs. Best fit linear regression lines have been calculated for each of the utility graphs; slope and intercept values for un-weighted, 2 x fitness and 3 x fitness values are shown in table 6.1.

With respect to results for the Cinema Booking System in figure 6.6, the lowest line shows the results of un-scaled (i.e. un-weighted) utility. It can be seen that the utility of the global search population reaches 0.189 at 20 generations but thereafter declines, becoming negative at 90 generations. Thereafter, as global search proceeds, population utility falls steadily to -1.284 at 500 generations, revealing the combined impact of the three ‘negative’ factors (i.e. sparsity, take-over, elapsed time) despite improving fitness of the population. In terms of halting the search, it is conjectured that a peak utility at 20 generations occurs a little too early in the search. Thus taking account of these initial results, the fitness factor in the utility calculation has been weighted by 2 and 3 to balance the scaling against the three ‘negative’ factors. The resulting average population utilities are also shown for comparison in figure 6.6. It is

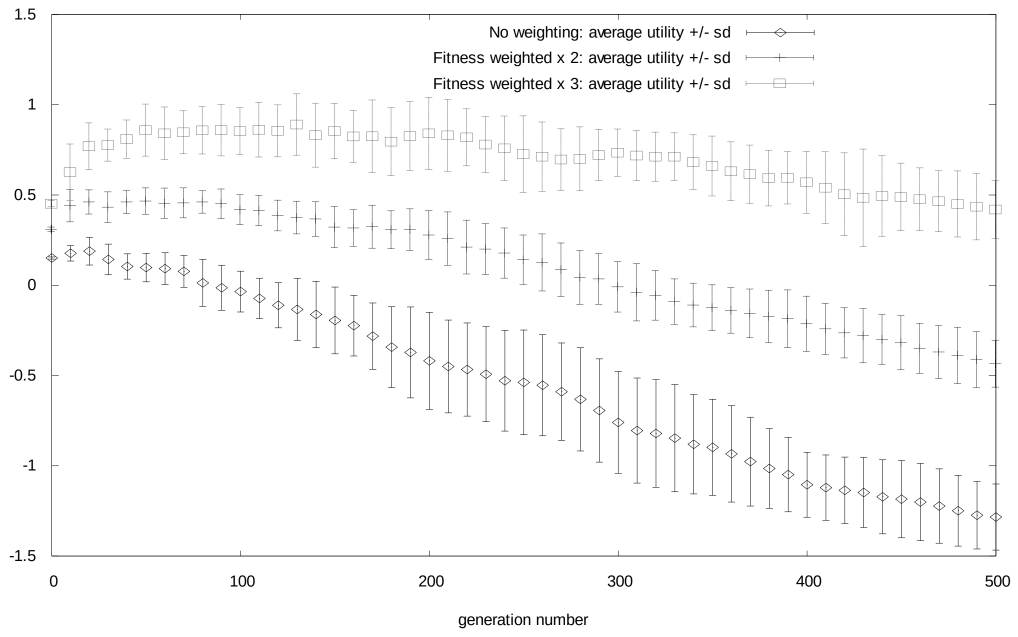

Figure 6.6. Average Population Utility for Cinema Booking System

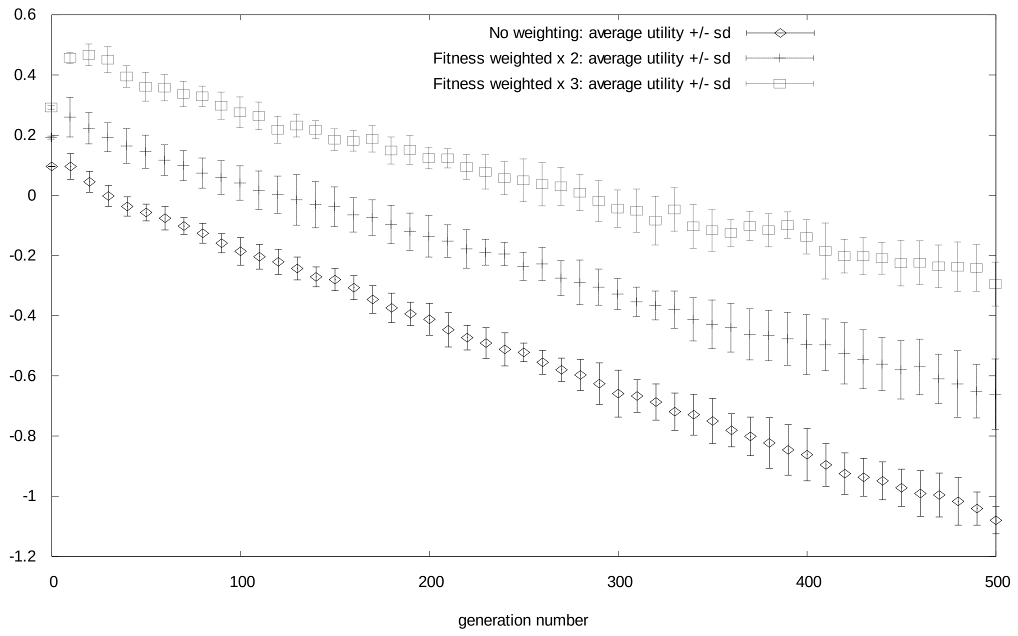

Figure 6.7. Average Population Utility for Select Cruises

Table 6.1. Slope and Intercept for Utility Values

	Cinema Booking System			Select Cruises		
	Un-weighted	2 x fitness	3 x fitness	Un-weighted	2 x fitness	3 x fitness
Slope	-0.0032314	-0.0019586	-0.0007420	-0.0023115	-0.0018068	-0.0013287
Intercept	0.2519449	0.5812292	0.8819186	0.0584487	0.2247239	0.4025558

evident that weighting fitness values results in higher utility values. For fitness weighted x 2, average population utility reaches 0.463 after 50 generations and although thereafter declining, remains positive until 290 generations. For weighted fitness x 3, utility reaches 0.887 at 130 generations, and remains positive to 500 generations. Slope values of linear regression best fit lines in table 1 reveal that slopes become closer to zero with increased weighting, whereas intercept values increase with increased weighting. Overall, the weighted calculations of utility appears a more balanced and useful expression of utility; their slower decline in utility allows the search to progress and thus improve population fitness overall. Of the two weighted calculations, 2 x weighting appears the more useful because zero utility is arrived at 300 generations unlike 3 x weighting where zero utility is not arrived at.

A broadly similar pattern of results is seen in figure 6.7 for the Select Cruises design problem. Once again, utility is calculated for un-weighted fitness, fitness weighted x 2, and fitness weighted x 3. As before, weighting of fitness values results in higher utility. Indeed, the slope and intercept values for Select Cruises best fit lines show broadly similar characteristics to those for Cinema Booking System. Once again, the weighted calculations of utility appear more balanced and useful; their slower decline in utility allows the search to progress. Of the two weighted calculations, 2 x weighting appears the more useful because zero utility is arrived at 120 generations unlike 3 x weighting where zero utility is not arrived at. Thus based on results for the Cinema Booking System and Select Cruises design problems, the 2 x fitness weighted calculation of utility is taken forward to the second experiment which investigates the performance of the Zone Isolator Agent.

In the second experiment assessing the performance of the Zone Isolator Agent, the Cinema Booking System and Select Cruises are used as example design problems, and global search is conducted over 50 runs for each of the three tactics i.e. zero utility, a single fall in utility and two sequential falls in utility. The characteristics of the design population at termination are observed, and are summarised in tables 6.2 and 6.3. Table 6.2 and 6.3 reveal the generation number at which the Zone Isolator Agent halts the search, and then at that generation, average population coupling, sparsity, take-over and lastly, utility. Standard deviation is shown in parentheses. For the Cinema Booking System, the ‘zero utility’ tactic halts search at generation 298 with a coupling fitness of 0.29. While this is a promising coupling value, sparsity and take-over are high indicating a lack of diversity in the population. For the ‘single fall’ tactic, search is

Table 6.2. Zone Isolation Tactics Results for Cinema Booking System

	Zero Utility		*Single Fall*		*Two Falls*	
Generation	291.90	(56.20)	18.55	(7.11)	58.40	(29.54)
Coupling	0.29	(0.19)	0.70	(0.02)	0.62	(0.06)
Sparsity	0.35	(0.06)	0.09	(0.05)	0.13	(0.08)
Take-over	0.50	(0.34)	0.01	(0.02)	0.04	(0.03)
Utility	0.00	(0.00)	0.44	(0.05)	0.45	(0.09)

Table 6.3. Zone Isolation Tactics Results for Select Cruises

	Zero Utility		*Single Fall*		*Two Falls*	
Generation	124.15	(22.81)	4.4	(1.35)	31.20	(5.44)
Coupling	0.66	(0.01)	0.83	(0.01)	0.72	(0.06)
Sparsity	0.38	(0.06)	0.06	(0.04)	0.08	(0.05)
Take-over	0.40	(0.02)	0.07	(0.01)	0.08	(0.09)
Utility	0.00	(0.00)	0.28	(0.02)	0.28	(0.03)

halted at generation 18 with a poor coupling value of 0.70. Although population diversity appears to be promising, the search has not progressed sufficiently to yield a good outcome with respect to coupling. However, for the ‘two falls’ tactic, search is halted at generation 58 giving a better trade-off between coupling and population diversity. Results for the Select Cruises design problem show a broadly similar pattern of results. For example the ‘zero utility’ tactic halts search at generation 124 with a coupling fitness of 0.66. Thus although coupling shows some promise, there are indications of a lack of diversity in the population. For the ‘single fall’ tactic, search is soon halted at generation 4 with a poor coupling value of 0.70. Lastly, for the ‘two falls’ tactic, search is halted at generation 31 giving a better trade-off between coupling and population diversity. Thus overall, it appears that the halting tactic providing the best balance between increasing population fitness versus increasing population diversity is two sequential falls in population utility.

With respect to the utility values at halting for ‘single fall’ and ‘two falls’ tactics shown in tables 6.2 and 6.3, it is interesting to note that utility is lower for Select Cruises compared to the Cinema Booking System. This might be due to the comparatively inferior coupling values inherent in the larger scale of the Select Cruises design problem. It is also noticeable from figures 6.5 and 6.6 that the decrease in average population utility from approximately 100 generations is not monotonic, which may at least in part explain why the ‘two falls’ tactic produces more useful results than the ‘single fall’ tactic. It is logical to speculate therefore that a time-averaged utility calculation might be useful in this situation. Indeed, Cobb and Grefenstette (1993) investigate triggered hyper-mutation in genetic algorithms for tracking changing environments and report that in such situations, time-averaged (or ‘windowed’) best fitness values are effective as a basis for selection. However, it may also be possible that this results in a more complex utility calculation with detrimental impact on agent performance. Furthermore, the results in tables 6.2 and 6.3 do suggest that the ‘two falls’ tactic produces satisfactory outcomes with respect to halting of search based on the trade-off between coupling and population diversity.

6.8 Conclusions

Experimentation conducted in this chapter reveals that multi-objective global search using the object-based representation and a non-dominated sorting algorithm enables multi-objective exploration of the overall software design search space and an effective increase in population fitness. A Global Search Agent has been implemented to conduct global search. In addition, the discrete nature of the object-based representation is exploited to arrive at localised zones of the search space based on the number of classes in the design. This provides an efficient interactive way to narrow the global search and produce useful zones within the search space for subsequent localised search.

Nevertheless, experiments also reveal the limitations inherent in the use of discrete fitness objective values within a non-domination sorting multi-objective evolutionary algorithm. Despite an increase in population fitness, a loss of population diversity is apparent as search progresses. Such a potential loss of population diversity is to be avoided if global search is to be part of an interactive framework of computational support for early lifecycle software design. Therefore, to address this, a Zone Isolator Agent has also been implemented which effectively monitors the fitness / diversity trade-off as search progresses. Should the trade-off become sufficiently unfavourable, the Zone Isolator Agent communicates with the Global Search Agent to halt global search.

Experiments have been conducted using the Cinema Booking System and Select Cruises example design problems. The experimental findings obtained suggest that a sufficiently representative illustration of the inherent characteristics and performance of not only the non-dominated sorting multi-objective search algorithm using a discrete fitness objective, but also halting of global search using the Zone Isolator agent, has been provided.

The value of global multi-objective search lies in enabling the progression from global search to search of localised zones wherein all software designs comprise the same number of classes. Without global search, it seems likely that the software designer would struggle to make any initial sense of the interactive search as a design episode. Indeed, the progression from global search to search of localised zones is judiciously managed by agent collaboration, and thus forms the beginning of an interactive design episode in which the designer and software agents cooperate with each other. The next chapter of this thesis builds on this by introducing additional task-focussed agents to further enable effective human / machine collaboration and evaluates their performance as an interactive framework via an empirical investigation.