4

Genetic Inspired Optimisation

4.1 What are Genetic Algorithms?

Genetic algorithms (GAs) are directed random search techniques used to look for parameters that provide a good solution to a problem. Essentially they are nothing more than educated guessing. The ‘education’ comes from knowing the suitability of previous candidate solutions and the ‘guessing’ comes from combining the fitter attempts in order to evolve an improved solution.

For example, the back propagation algorithm is a gradient based method for finding a weight set for a MLP that best maps the inputs onto the output, a search that can also be performed by GAs [**7**]. The optimisation problem of interest in this work is finding a schedule for electrically charging storage devices (hot water tanks and storage radiators) over a 24 hour period, given that electricity prices vary half-hourly. A solution is sought

that minimises electricity costs whilst satisfying the hot water or thermal comfort requirements.

4.2 How do GAs Work?

The inspiration for GAs came from nature and survival of the fittest. In a population, each individual has a set of characteristics that determine how well suited it is to the environment. Survival of the fittest implies that the ‘fitter’ individuals are more likely to survive and have a greater chance of passing their ‘good’ features to the next generation. In sexual reproduction, if the best features of each parent are inherited by their offspring, a new individual will be created that should have an improved probability of survival. This is the process of evolution.

In nature the ‘blueprint’ of individuals is contained within their DNA. The DNA can be thought of as a string of genes, with each gene or combination of genes representing a particular feature. Reproduction is the ‘crossover’ of two DNA strings to produce a new blueprint that has genes from both parents. Mutation can also occur where a particular gene is not an exact copy of either parent.

In genetic algorithm terms, a candidate solution is often referred to as a chromosome or string, which is a sequence of encoded numbers. This is commonly referred to as a bit string if the numbers are binary encoded.

The process involved in GA optimisation problems is based on that of natural evolution and broadly works as follows,

1. Randomly generate an initial population of potential solutions.

2. Evaluate the suitability or ‘fitness’ of each solution.

3. Select two solutions biased in favour of fitness.

4. Crossover the solutions at a random point on the string to produce two new solutions.

5. Mutate the new solutions based on a mutation probability.

6. Goto 2.

4.3 The GA Operators

Selection, crossover and mutation are the basic operators involved in GAs. How these and other factors can affect the operation of GAs will be demonstrated by means of several examples and experimental observations.

Consider the popular board game ‘Mastermind’ where a player has to determine a hidden sequence of colours starting from an initial random guess. This initial guess is scored with a black marker for each colour in the correct position and a white marker for a correct colour but in the wrong position. Further guesses are made and scored until the correct sequence is determined or a given number of attempts have been made. In this game the correct solution evolves from the more suitable of all previous attempts, with clues from unsuitable candidate solutions also being part of the deduction process. This is a type of ‘blind’ optimisation problem where no information is available on what makes a good solution, only information on how good solutions are.

Given a few initial guesses the player will *select* high scoring attempts and perform *crossover* to see if this results in an improvement. New colours will almost certainly have to be *mutated* into the ‘educated guesses’ in the attempt to find the correct sequence.

Fig 4-1 demonstrates how these three operators work considering a scoring scheme where a point is scored only for a number in the correct position.

The GA search procedure is very easy to understand and implement, with nature providing ready examples of exactly how things could be done.

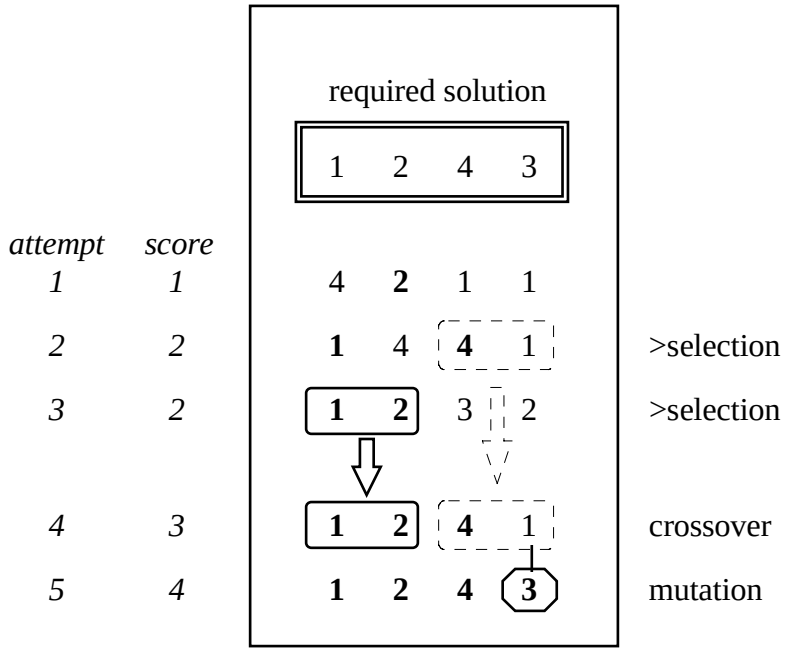


Fig 4-1 *An example of how the required solution evolves using the selection, crossover and mutation operators*

4.4 Implementation

4.4.1 Encoding

In optimisation problems a set of parameters is sought that will give the best solution to a particular problem. In order to implement a GA these parameters must be encoded into a string so that crossover and mutation can be applied. Binary encodings are the most common, due to the fact that Holland used them in his early pioneering work [**66**]. In DNA base 4 encoding is used, as the building blocks of DNA can take on 4 values, translated as A, C, G, or T.

Any base can be used, as it is just a different method of encoding the same information, but the lower the base the longer the string will be. For example, if a number is sought between 0 and 255 then this can be encoded as a binary string of length 8, a base 4 string of length 4, a base 16 string of length 2 or a base 256 string of length 1, as shown in Table 4-1.

Table 4-1 *Representations of the base 10 numbers 0 and 255 in different bases*

base 2 length=8	base 4 length=4	base 10 length=3	base 16 length=2	base 256 length=1
00000000	0000	000	00	0
11111111	3333	255	FF or |15|15|	|255|

It is clear that the importance of the operators will change depending on the base used. In base 2, the two given strings contain all the information required to derive any number between 0-255 by crossover alone. In base 16, two strings can at most lead to only 4 different numbers by crossover alone, mutation being required to introduce new information. In base 256 crossover cannot occur, mutation being the only operator that can introduce new numbers and finding a specific number becomes a pure random search.

In choosing an encoding scheme the nature of the problem will play a major role. If many real valued numbers are required in a solution then binary encoding becomes impractical as the string length increases. In [**67**] a method of selective genome growth is proposed that helps solve the problem of choosing how to represent a genetic algorithm.

4.4.2 Population Size

The population size is the number of candidate solutions in any one generation. In natural evolution the total population size is governed by what is sustainable by the environment and similarly in GAs the larger the population size the more computationally intensive (in terms of memory requirement) is the search.

In nature, the bigger the gene pool the more diverse is the genetic make up of the population with many individuals each with their own set of characteristics that enable them to survive. One advantage of this diversity is that there will be no dominant gene that, for instance, may be susceptible to a particular disease and result in the elimination of the whole species. In the bird family, sub-species have evolved with dominant character-

istics that allow them to survive their local conditions and in effect are sub-optimal solutions in the search for a global ‘super-bird’. With large populations it can be seen how the search for the global optimal solution can be a slow (if not never-ending) process.

If the population size is small (e.g. a pride of lions), then a strong individual quickly becomes dominant and the diversity of the gene pool is restricted. The result is that good individuals (local optima) are quickly created but the dominance of particular genes restricts the search space. The chance of evolving the ultimate ‘super-lion(ess)’ (global optimum) is severely limited and would depend on mutation introducing new genes to diversify the search.

As new solutions are generated it is common to keep the population size constant by eliminating individuals (or letting them die), although this does not have to be the case. Ideas for the selection procedure for elimination are plentiful in nature. For example, each generation could be completely replaced by its offspring, or as a new offspring is created it could be accepted or rejected depending on its fitness. The advantage computers have over nature is that good individuals do not have to die and can be retained for indefinite reproduction. The retention of certain fit individuals is known as ‘elitism’.

4.4.3 Selection

This is the procedure for choosing individuals (parents) on which to perform crossover in order to create new solutions. The idea is that the ‘fitter’ individuals are more prominent in the selection process, with the hope that the offspring they create will be even fitter still.

Two commonly used procedures are ‘roulette wheel’ and ‘tournament’ selection. In roulette wheel, each individual is assigned a slice of a wheel, the size of the slice being proportional to the fitness of the individual. The wheel is then spun and the individual opposite the marker becomes one of the parents. In tournament selection several individuals are chosen at random and the fittest becomes one of the parents.

4.4.4 Crossover

Along with mutation, crossover is the operator that creates new candidate solutions. A position is randomly chosen on the string and the two parents are 'crossed over' at this point to create two new solutions. Multiple point crossover is where this occurs at several points along the string. A crossover probability (P_c) is often given which enables a chance that the parents descend into the next generation unchanged.

4.4.5 Mutation

After crossover, each bit of the string has the potential to mutate, based on a mutation probability (P_m). In binary encoding mutation involves the flipping of a bit from 0 to 1 or vice versa.

4.5 Experiments with GAs

4.5.1 Chinese Hat Optimisation Problem

To empirically evaluate the importance of the various parameters and techniques in GAs, several optimisation tests were performed. The code used is based on that in Appendix D. The experiments used tournament selection and a constant population size with the offspring replacing the parents every generation.

The fitness evaluation function (fitness landscape, scoring template) of candidate solutions for the first optimisation problem examined is shown in Table 4-2. For reference purposes this problem has been named the Chinese Hat because the scoring template diverges linearly outwards from the centre. There are two possible solutions for maximum fitness, one of which is shown by candidate solution 2, the other is the inverse of this where all the bits flip. The total number of candidate solutions is $2^{(\text{string length})}$.

Table 4-2 *An example of how the fitness of the solutions to the Chinese Hat problem are evaluated for a string length of 8. Each bit value in a solution is multiplied by the value in the same position in the scoring template and the total fitness is the square of the sum of all the bit scores. Each bit can have a value of 1 or –1*

Scoring Template	**4**	**3**	**2**	**1**	**-1**	**-2**	**-3**	**-4**
Candidate Solution 1	*1*	*1*	*-1*	*1*	*-1*	*-1*	*1*	*-1*
Bit by bit Score 1	4	3	-2	1	1	2	-3	4
Total Score 1 = 10^2 = 100								
Candidate Solution 2	*-1*	*-1*	*-1*	*-1*	*1*	*1*	*1*	*1*
Bit by bit Score 2	-4	-3	-2	-1	-1	-2	-3	-4
Total Score 2 = $(-20)^2$ = 400								

In the experiments, tests for each particular parameter setting were repeated to convergence 200 times to determine the average number of generations required to find the solution. Each subsequent trial differed by randomly generating a new initial population. After each crossover, mutation was only allowed on one randomly selected bit and whether it occurred depended on P_m.

4.5.2 Results

The results of varying the GA parameters for the Chinese Hat optimisation problem are shown in Fig 4-2 to Fig 4-8. All comments and discussion related to each figure are included below that figure.

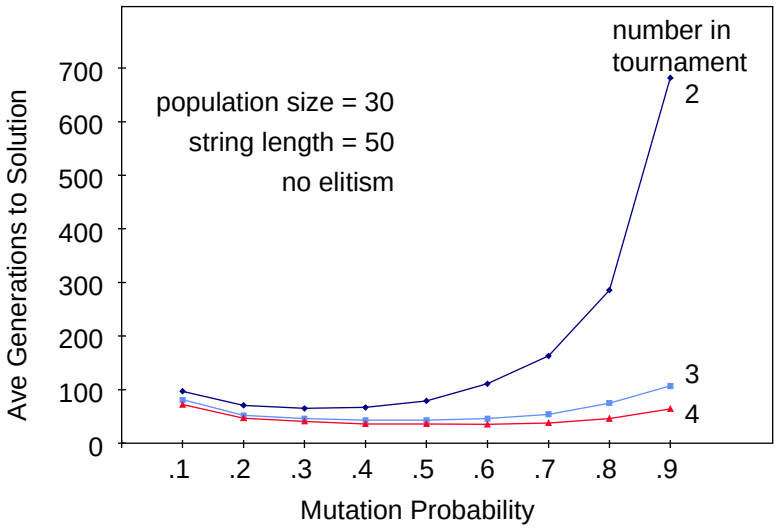


Fig 4-2 *The effects of P_m and the selection procedure*

By increasing the number of candidates (competitors) in the tournament for parenthood the number of generations required to convergence reduces. This would indicate that little diversity in the gene pool is required for this particular problem. There is also an optimum P_m around 0.5. With a higher mutation probability the number of generations starts to increase, although this becomes less significant as the selection procedure is made more competitive. In Fig 4-2, P_c =1.

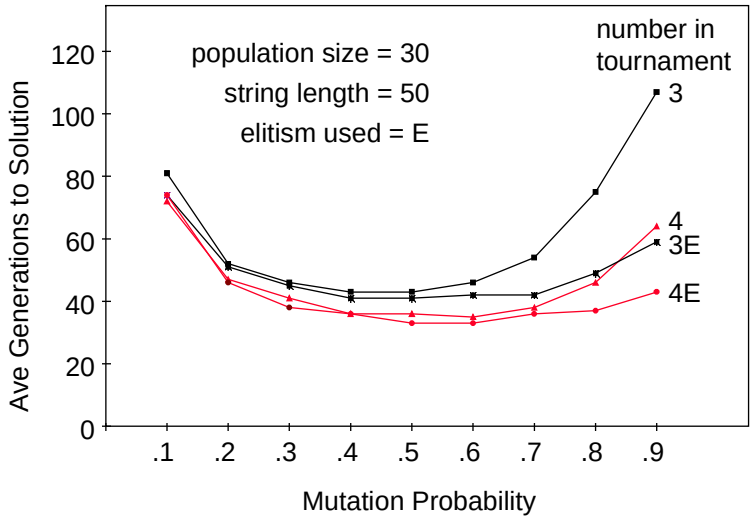


Fig 4-3 *Elitism*

The introduction of an elitist strategy, where the best individual is always retained, shows significant improvements in performance but only for the higher mutation rates, indicating the solution is evolved from mutation of this ‘best’ individual.

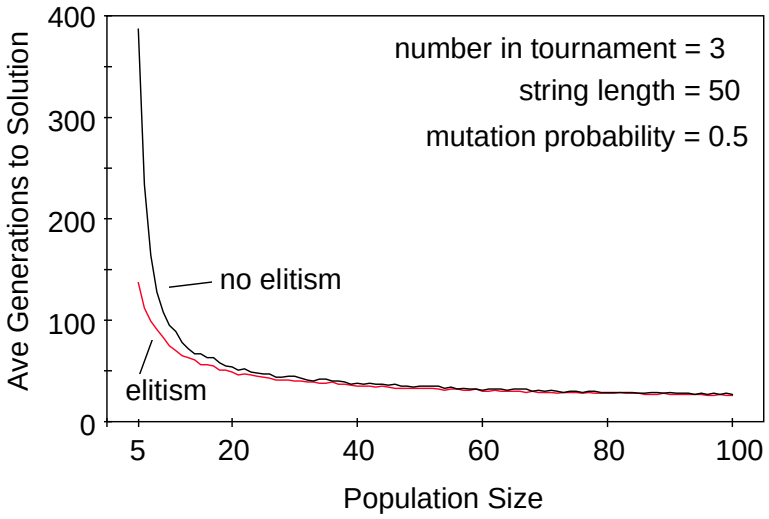


Fig 4-4 *Population size*

As would be expected, the larger the population size the fewer generations are required as the search space is increased. The highest rates of gain are seen by increasing the population size to 20, but even after this consistent reduction still occurs, as shown in Fig 4-5.

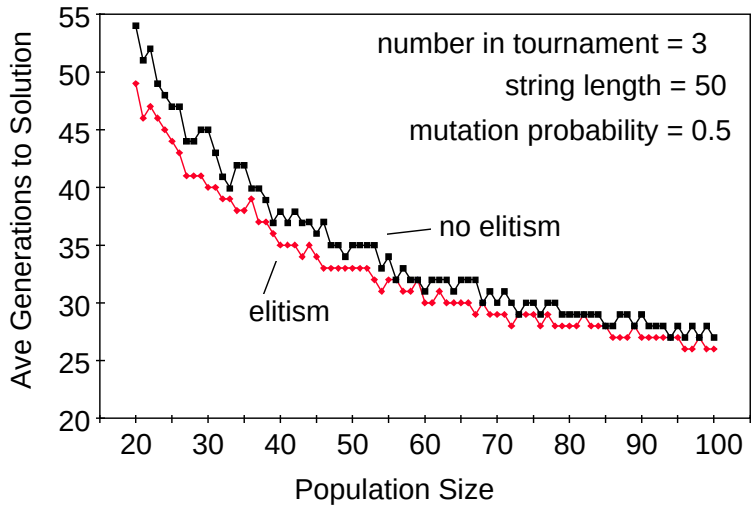


Fig 4-5 *Population size*

A close up of Fig 4-4 shows consistent improvement in performance with increasing population size.

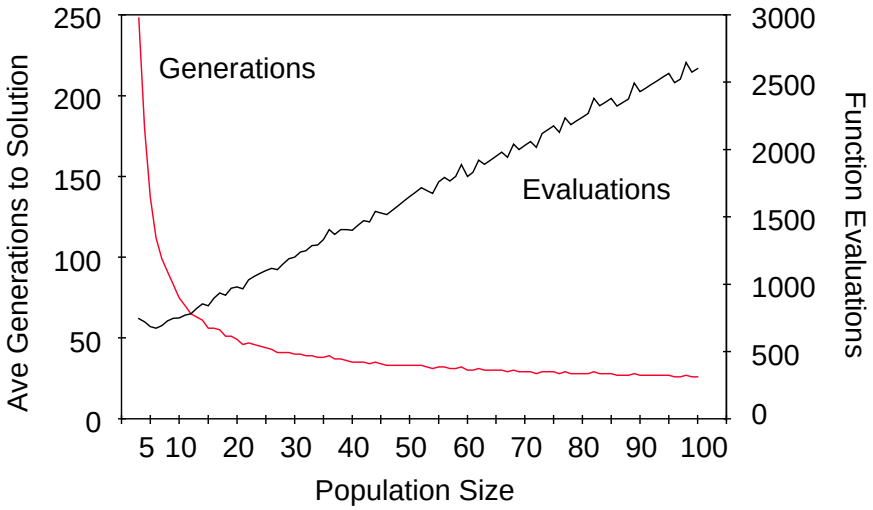


Fig 4-6 *Function evaluations*

In serial computing it is not the number of generations that is important but the number of function evaluations. That is, how many solutions must be evaluated before the optimum is reached, or roughly the number of generations multiplied by the population size. This gives an indication of the computing power (or time) required to solve the problem, assuming that evaluating the cost of each solution is a significant portion of the whole process. Fig 4-6 is the same data as that of the elitist tests in Fig 4-4, but with the number of evaluations also shown. It can be seen that for the given parameters, a population size of 6 is the most economical. After this the number of evaluations increases linearly with population size.

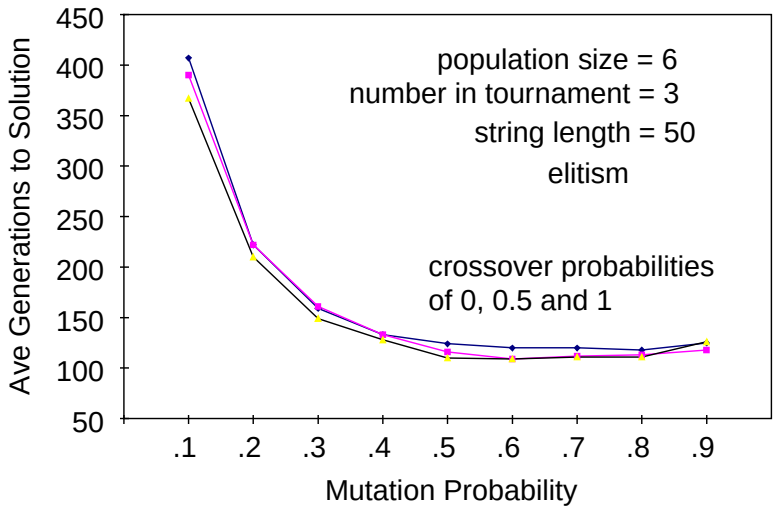


Fig 4-7 *The effect of crossover and mutation probabilities for a population size of 6*

Thus far crossover has occurred in every reproduction. By introducing a crossover probability, the relative importance of crossover and mutation can be examined. This is shown for the most efficient population size of 6 and crossover probabilities of 0, 0.5 and 1. What is seen is that the optimisation procedure for this small population relies solely on mutation with the crossover probability having a negligible effect.

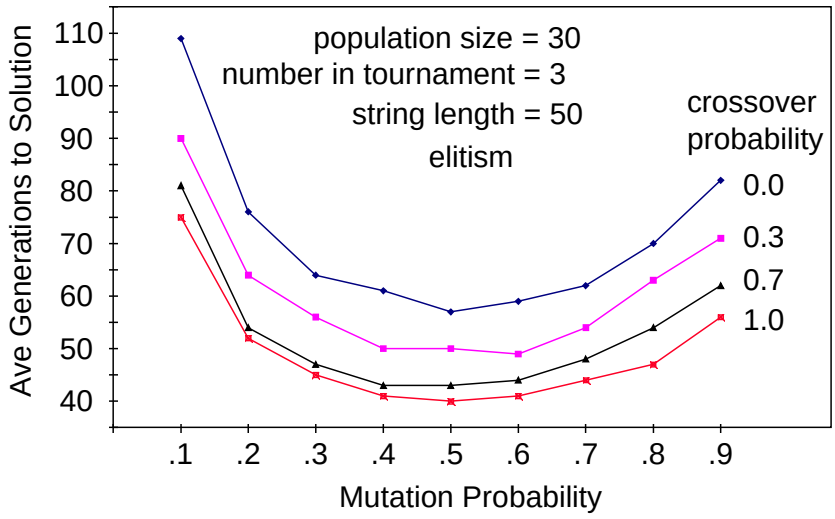


Fig 4-8 *The effect of crossover and mutation probabilities for a population size of 30*

With a larger population size increasing the crossover probability does improve the performance. Fig 4-8 is generated by the same procedure as Fig 4-7 but with a population size of 30. It can be seen with this larger population size there is an optimal P_m but improvements are also made by increasing P_c. What Fig 4-7 and Fig 4-8 show is not unexpected and is reflected in nature in that small populations rely on mutation for diversity whereas in larger populations it is a combination of crossover and mutation.

The experiments on this simple optimisation problem have illustrated that selecting the correct parameters is very important in genetic algorithms. What is also very evident is that there are definite relationships between all the parameters showing that fine-tuning is required to increase the speed to success, or reduce the chance of failure. The technique always managed to solve the problem, but how does it compare with other hill-climbing methods?

4.5.3 Other Iterated Hill-Climbing Methods

Other optimisation methods exist of which three were used for comparison with the GA. The following descriptions of these techniques are reproduced from [**68**].

4.5.3.1 Steepest-Ascent Hill Climbing (SAHC)

1. Choose a string at random. Call this string *current-hilltop.*
2. Going from left to right, systematically flip each bit in the string, recording the fitness of the resulting strings.
3. If any of the resulting strings give a fitness increase, then set *current-hilltop* to the resulting string giving the highest fitness increase (ties are decided at random).
4. If there is no fitness increase, then save *current-hilltop* and goto step 1. Otherwise goto step 2 with the new *current-hilltop.*
5. When a set number of function evaluations have been performed (here, each bit flip in step 2 is followed by a function evaluation), return the highest hilltop that was found.

4.5.3.2 Next-Ascent Hill Climbing (NAHC)

1. Choose a string at random. Call this string *current-hilltop*.

2. For *i* from 1 to *l* (where *l* is the length of the string), flip bit *i*; if this results in a fitness increase, keep the new string, otherwise flip bit *i* back. As soon as a fitness increase is found, set *current-hilltop* to that increased fitness string without evaluating any more bit flips of the original string. Go to step 2 with the new *current-hilltop*, but continue mutating the new string starting immediately after the bit position at which the previous fitness increase was found.

3. If no increase in fitness were found, save the *current-hilltop* and goto step 1.

4. When a set number of function evaluations has been performed, return the highest hilltop that was found.

4.5.3.3 Random-Mutation Hill Climbing (RMHC)

1. Choose a string at random. Call this string *current-hilltop*.

2. Choose a bit at random to flip. If the flip leads to an equal or higher fitness, then set *current-hilltop* to the resulting string.

3. Goto step 2 until an optimum string has been found or until a maximum number of evaluations has been performed.

4. Return the current value of *current-hilltop.*

1000 trials of each of these three algorithms were performed on the Chinese Hat problem for a string length of 50. The average number of evaluations, given in Table 4-3, shows that a GA is not the best method of solving this particular problem. In fact NAHC always reaches a global solution by traversing the string just once because the Chinese Hat is a smooth function when traversed from left to right.

Table 4-3 *The average number of function evaluations over 1000 trials for the Chinese Hat problem with a string length of 50*

SAHC	NAHC	RMHC	best GA	MRMHC
1082	48.6	190	650	430

The best GA based performance had a population size of 6 with a randomly selected gene being mutated with a probability of 0.5. With these parameters it was shown that crossover had a very limited effect (Fig 4-7, on page 84). As mutation appears to be the dominant process a variation of RMHC we called *multiple random mutation hill climbing* (MRMHC) was tried on the Chinese Hat function. This is basically RMHC but with each gene having a mutation probability as opposed to one randomly selected gene being mutated. Another way of describing MRMHC is a GA with a population size of 1 and the mutated offspring only surviving if it is as good as or better than the parent.

Fig 4-9 (on page 88) shows the average performance of MRMHC over 1000 tests with varying mutation probabilities and string lengths. The circled points represent the optimum mutation probabilities for the various string lengths with a pattern emerging that a P_m of (1/string length) appears to be the optimum. On average this is equivalent to 1 bit change per mutation which is an unsurprising result. Any less than this and some evaluations will be wasted as there will be no change, any more and there will be problems in fitting the last bit into position as there is a higher probability that more than one bit will be changed at once. The best performance with MRMHC for a string length of 50 was 430 evaluations, which occurred with P_m at just over 1/50 or 0.02.

The optimum mutation rate for MRMHC is on average 1 bit change per string, which is almost similar to RMHC, in which only one bit change per string is permitted. Similar results would be expected but it is noted that MRMHC requires over twice as many evaluations as RMHC. Why should this be so?

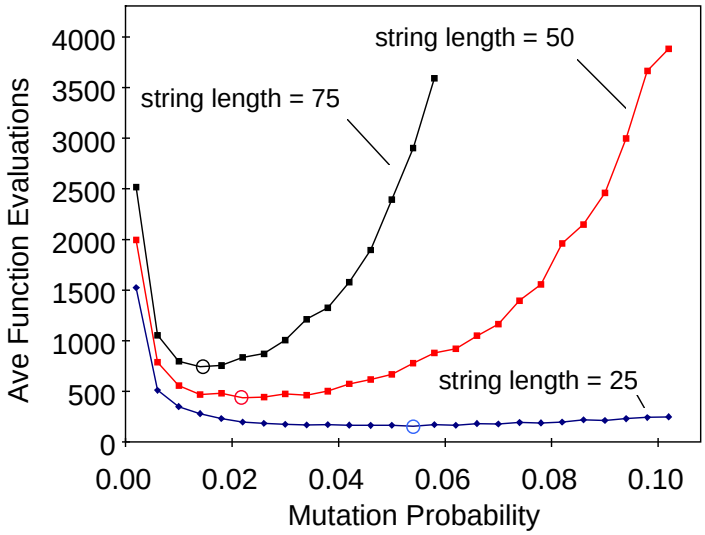


Fig 4-9 *The effect of mutation probability and string length for Multiple Random Mutation Hill Climbing optimisation*

If there is only one bit flip that is required to reach the optimum then RMHC should find this in an average number of evaluations equivalent to the string length. In MRMHC it is possible that in three consecutive evaluations the number of bit flips is 0,1 and 2, giving an average of 1. An evaluation with 0 flips is wasted and because only one bit requires correction, two bit flips will never find the optimum. It can thus be hypothesised why MRMHC gives a worse performance than RMHC for this particular problem.

4.5.4 Royal Road Functions

So what kind of problems will GAs be superior at solving than other search techniques?

The Schema Theorem and Building Block Hypothesis [**66, 69**] play on the idea that solutions are made up of short blocks of fit schema that use crossover to build up these schema into desirable solutions. A set of functions known as the 'Royal Roads' [**68, 70**, **71**, **72**] were developed that provide a fitness landscape designed specifically to be easily solvable by GAs if they did work in this building block manner. As described by the developers (Mitchell et al.), *'given the building block hypothesis, one might expect*

Table 4-4 *The Royal Road* (R_1) *fitness function. A bit string of length 64 contains 8 short schema that are the building blocks of the optimal schema. The wildcard '*' represents a 0 or 1 (or 'do not care'). The fitness of each candidate solution increases with the number of these building blocks present.*

11111111 ******** ******** ******** ******** ******** ******** ********	Schema 1	= 8
******** 11111111 ******** ******** ******** ******** ******** ********	Schema 2	= 8
******** ******** 11111111 ******** ******** ******** ******** ********	Schema 3	= 8
******** ******** ******** 11111111 ******** ******** ******** ********	Schema 4	= 8
******** ******** ******** ******** 11111111 ******** ******** ********	Schema 5	= 8
******** ******** ******** ******** ******** 11111111 ******** ********	Schema 6	= 8
******** ******** ******** ******** ******** ******** 11111111 ********	Schema 7	= 8
******** ******** ******** ******** ******** ******** ******** 11111111	Schema 7	= 8
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111	Schema Opt	= 64
11111111 11110011 01110011 ***11111111 11111111*** 00000001 11110010 ***11111111***	e.g. Score	= 32

that the building block structure of R_1 will lay out a "royal road" for the GA to follow to the optimal string'. Table 4-4 shows one of these Royal Road functions, R_1.

In their analysis, Mitchell et al. used a GA with a population size of 128, single point crossover with P_c fixed at 0.7 and P_m at 0.005, full details are given in [**68**]. Over 200 runs the mean number of GA function evaluations was 61,334, an order of 10 times higher than RMHC (6,179). NAHC and SAHC never reached the optimum solution, which is not unexpected given the nature of the fitness landscape.

In section 4.5.2 the importance of the GA parameters was demonstrated, although only on a simple smooth function that proved easier to solve by other methods. The Royal Road problem was investigated in the same manner to determine if the nature of the problem affected the relationship between the parameters.

Initial tests were performed to see if the results of Mitchell et al. could be replicated and also to examine the effect of varying the GA parameters. Mutation probabilities between 0.002 (0.13 in 64) and 0.05 (3.2 in 64) were tested for crossover probabilities between 0 and 1 inclusive. Each set of parameters was repeated to convergence 20 times and the mean value recorded. Tournament selection was used where each parent was the best of 5 randomly chosen candidates. The results are shown in Fig 4-10.

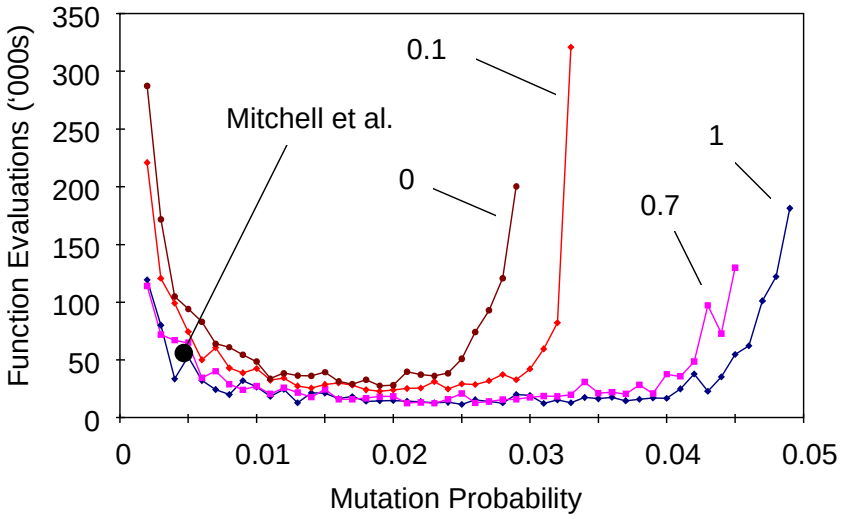


Fig 4-10 *The effect of the mutation probability for four crossover probabilities (0,0.1,0.7,1) on the Royal Roads (R_1) landscape. Each point is the average over 20 tests with a population size of 128. Mitchell et al. used a mutation probability of 0.005 (0.33 in 64) and crossover probability of 0.7 that gave a mean of 61,334 function evaluations to convergence over 200 tests.*

The results of Mitchell et al. were easily replicated even though a different selection procedure was used. By increasing P_m to 0.02 (1.3 in 64) the number of evaluations was reduced to around 14,000, a factor of 4 improvement and only twice as many as RMHC. With this mutation probability the function could be optimised in 28,000 evaluations without using crossover at all, half as many as the evidently poorly tuned GA of Mitchell et al. With no crossover, each offspring is a mutation of a parent chosen due to its fitness.

It has been demonstrated that a GA with no crossover can outperform a poorly tuned GA on a fitness landscape purposely designed to suit the crossover operator. If it can be discovered what determines a good mutation probability with no crossover then this should be generally applicable when crossover is applied.

With no crossover, the relationship between the mutation probability and selection procedure was examined. In previous tests on the R_1 landscape tournament selection was used where each parent was the fittest of 5 randomly selected candidates. The number of candidates was varied along with P_m, as shown in Fig 4-11 (on page 91). What is clear is that the less stringent the selection, the tighter the band is for an acceptable value of P_m.

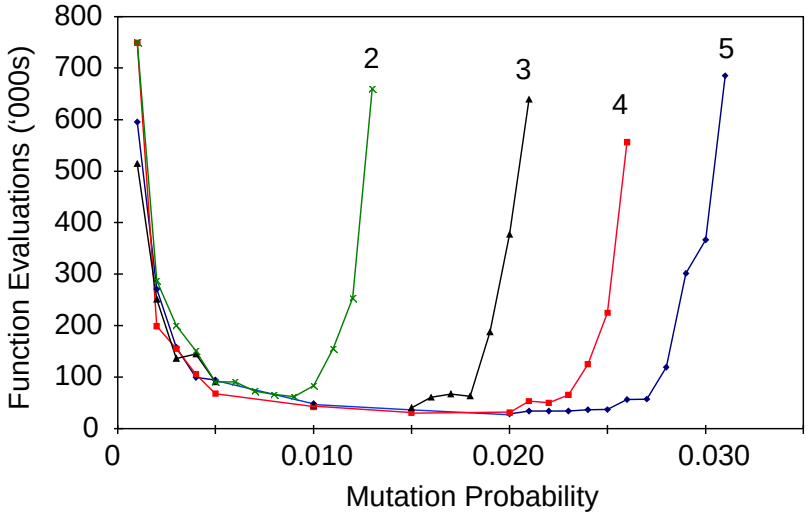


Fig 4-11 *The relationship between the mutation probability and the number of contestants in tournament selection (2,3,4, and 5). The crossover probability is 0 and the population size is 128. The mean of 20 trials was recorded.*

For each selection policy there is also an upper mutation probability past which the required evaluations increase exponentially; the more stringent the selection then the higher is this upper limit.

The objective of this exercise was to discover what determines a good mutation probability, which has been shown for R_1 to also depend on the selection procedure used, becoming more important the weaker the selection procedure. There is a definite lower limit around 0.005 or 0.33 in 64.

In order to determine a desirable mutation rate the effect of the population size must also be investigated. Fig 4-12 (on page 92) shows that given a near optimal P_m (0.01) there is little sensitivity to population size, but as P_m increases so does the sensitivity to population size.

The conclusion reached thus far is that the mutation probability is the most important GA parameter in solving the R_1 landscape. There is also much evidence (and common sense) to suggest that the optimum mutation probability is related in some way to the string length. In order to test this theory MRMHC (a GA with no crossover and population size 1 with the new solution being retained if it is better than or equal to the parent)

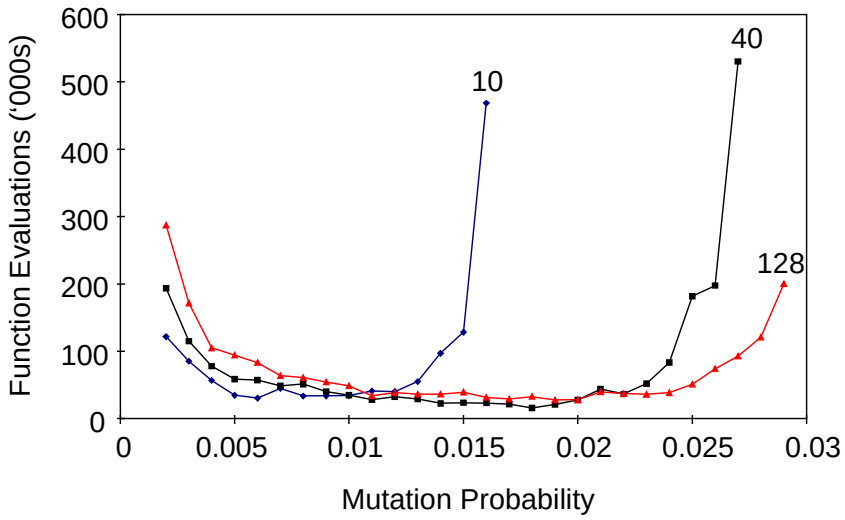


Fig 4-12 *The effect of the mutation probability and the population size (10,40,128). In these cases the P_c = 0 and the number of candidate parents is 5.*

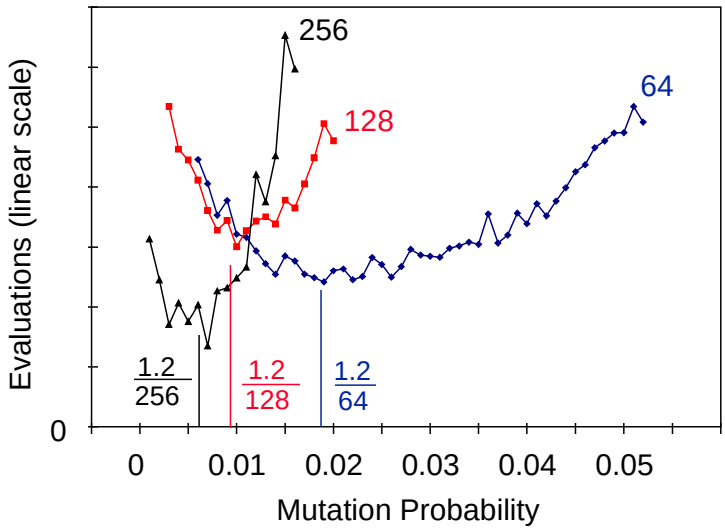


Fig 4-13 *The effect of the mutation probability on MRMHC, averaged over 200 tests. The optimum is (1.2/64).*

was used to solve three versions of R_1, with string lengths of 64, 128 and 256. The results in Fig 4-13 show that the longer the string the more sensitive is the search to P_m (the y-axis scale in Fig 4-13 is different for each population size). The optimum value of P_m was observed to be about (1.2/string length). The question to be investigated now is what is special about this mutation rate?

In their work, Mitchell et al. analysed the RMHC algorithm with a simple derivation based on probability that gave the expected number of function evaluations to solve R_1.

Consider R_1 as in Table 4-4. In each schema of length 8 the number of possible combinations is 2^8. If one and only one bit is changed in each evaluation then the chance of this bit being in a specific schema is 1/8, since there are 8 schemas in total. Thus the chance of randomly creating a particular schema is once every $2^8 \times 8$ evaluations. Initially there are eight schemas to choose from so the chance of creating any schema is once in every $2^8 \times 8/8$ evaluations. Once one schema is found the chance of finding a further schema decreases to 7/8 of that of finding the first since 1 in 8 bit changes are likely to be wasted changing the already discovered schema. The number of evaluations required to find this second schema thus increases to 8/7 that required to find the first. The expected number of evaluations to find a single schema is in fact slightly more than 2^8 and as determined by a Markov-chain analysis it is 301.2 [**68**]. The expected number of evaluations to solve the problem is thus,

$$301.2 \times 8 \times \left(\underset{\text{1st}}{\uparrow}\frac{1}{8} + \frac{1}{7} + \frac{1}{6} + \frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + \underset{\text{8th}}{\uparrow}\frac{1}{1} \right)$$

$$\text{1st} \quad \text{discovered schema} \quad \text{8th}$$

Tests were performed for RMHC that tracked the creation of the schema in the solution in order to confirm the theoretical performance. Table 4-5 shows the results averaged over 1000 trials which almost mirror the theoretical expectations.

Table 4-5 *The theoretical and experimental (averaged over 1000 trials) number of evaluations to discover each subsequent schema for R_1 using RMHC. The total theoretical evaluations = 6,549, experimental = 6,542 and Mitchell et al. = 6,179.*

schema	1	2	3	4	5	6	7	8
theoretical evaluations	301.2	344.6	401.6	481.9	602.4	803.2	1204.8	2409.6
experimental evaluations	284	355	384	508	622	797	1182	2410

RMHC has been shown to behave as the probability theory predicted. In GAs the theory of how they behave remains a theory, with little experimental evidence to try to observe their actual behaviour.

Tests were performed using GAs on the R_1 landscape that tracked the formation of the schema. The trials were performed 500 times with the maximum number of generations set at 800. The maximum, minimum and average fitness of the population were recorded at each generation and averaged for the trials that had not converged. Initially the crossover rate was set at 0.7, population size 128 and the number of competitors in the tournament was 5. Three mutation rates were used, 0.33/64, 1.3/64 and 2.7/64. The results are shown in Fig 4-14 to Fig 4-17.

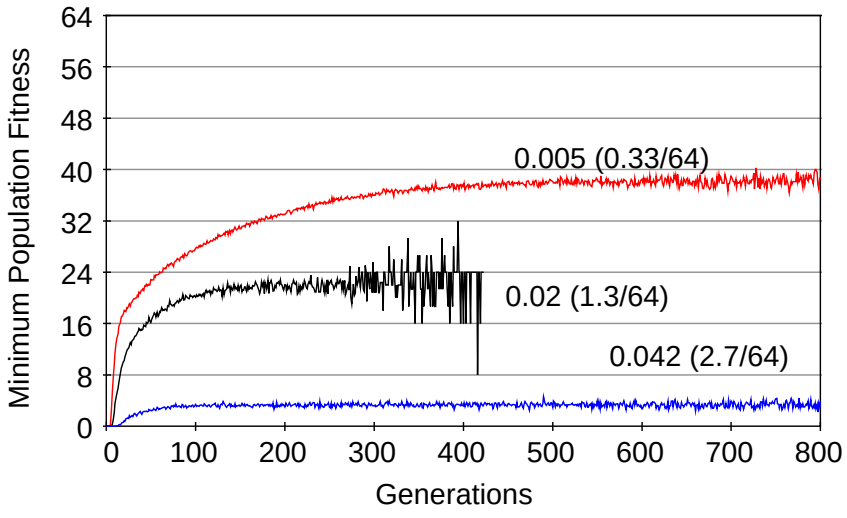


Fig 4-14 *The effect of the mutation rate on the minimum population fitness*

The lower the mutation rate the fitter is the worst individual in the population. Note that in all cases the minimum fitness reaches a plateau and only for the middle mutation rate of 1.3/64 do all the trials converge.

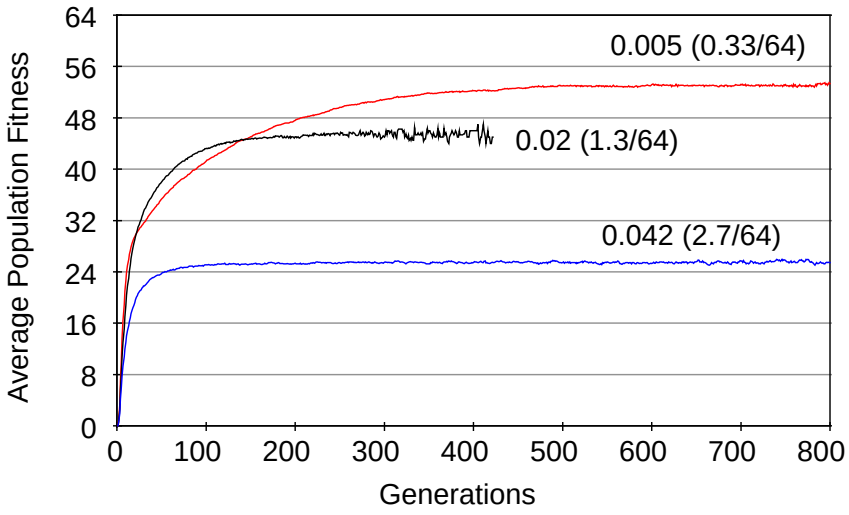


Fig 4-15 *The effect of the mutation rate on the average population fitness*

It can be seen how the rise in average population fitness is initially high for all cases. The best average population is with the lowest mutation rate, but this does not find the global solution in all cases.

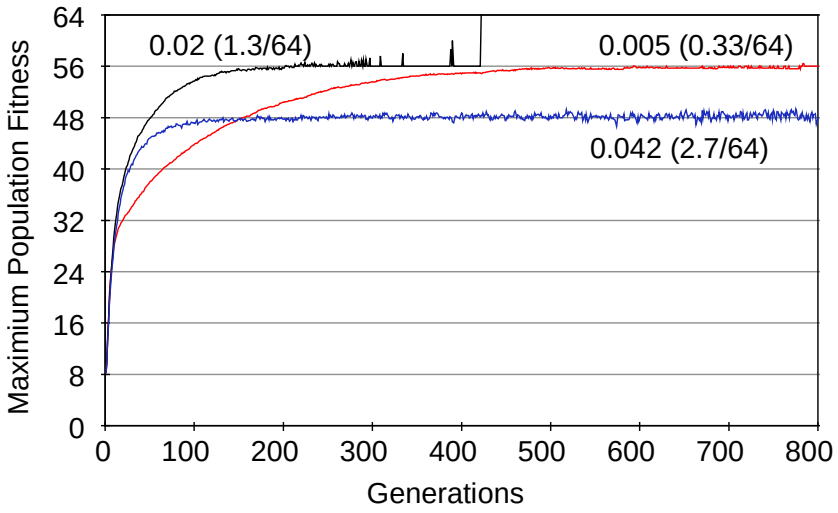


Fig 4-16 *The effect of the mutation rate on the maximum population fitness*

Note that the high mutation rate generally limits the maximum population fitness to 6 schema (a fitness of 48). This is because schema are destroyed as new ones are created.

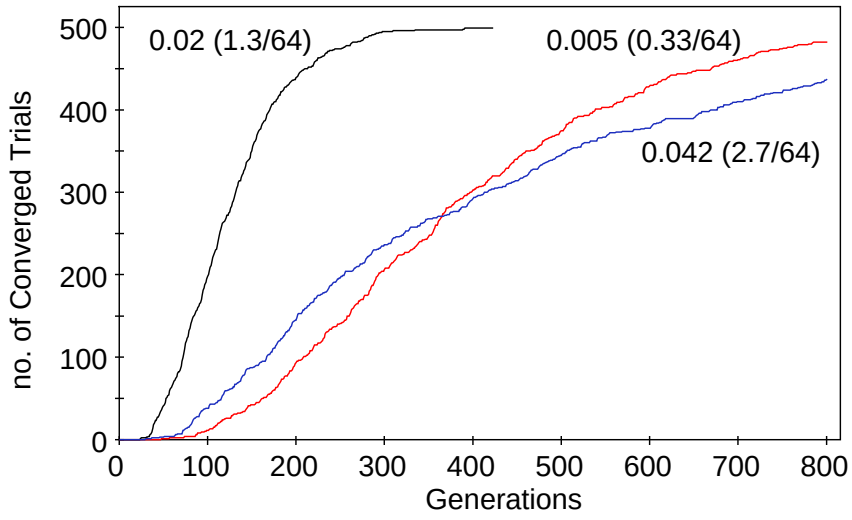


Fig 4-17 *The number of converged trials at each generation for the three mutation rates*

Quite clearly from a pure optimisation perspective, where the goal is to find the global solution, the mutation rate of 1.3/64 is superior in all respects, as shown by Fig 4-17.

Fig 4-18 to Fig 4-21 show the effect of the crossover probability for the near optimum mutation rate of 1/64. It can be seen that increasing P_c only improves the speed to convergence, with no other effect on the behaviour of the GA, as identified by all the lines converging to the same fitness value. In all cases every trial converged, even with P_c=0. The conclusion drawn is that mutation is the most important operator for this particular problem.

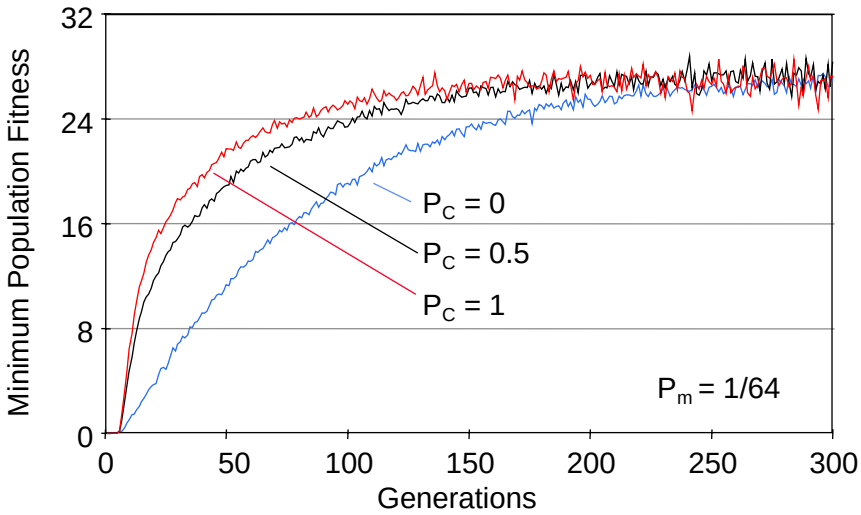


Fig 4-18 *The effect of P_c on the minimum fitness*

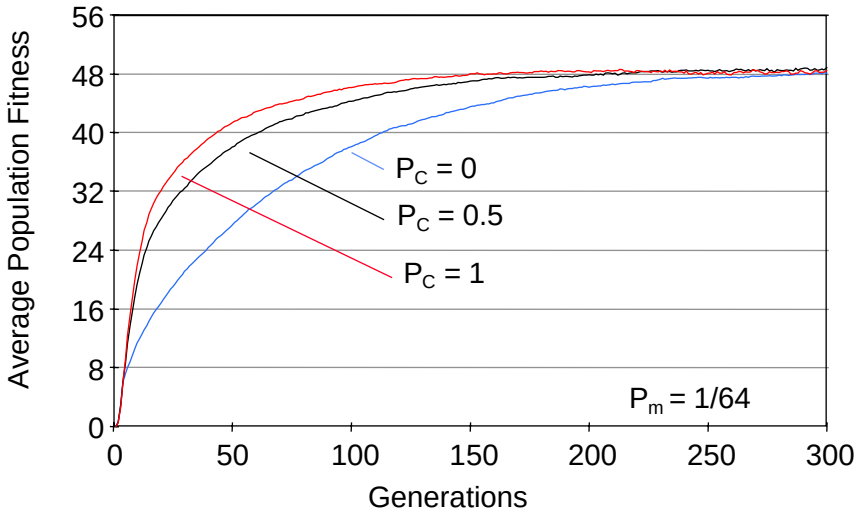


Fig 4-19 *The effect of P_c on the average fitness*

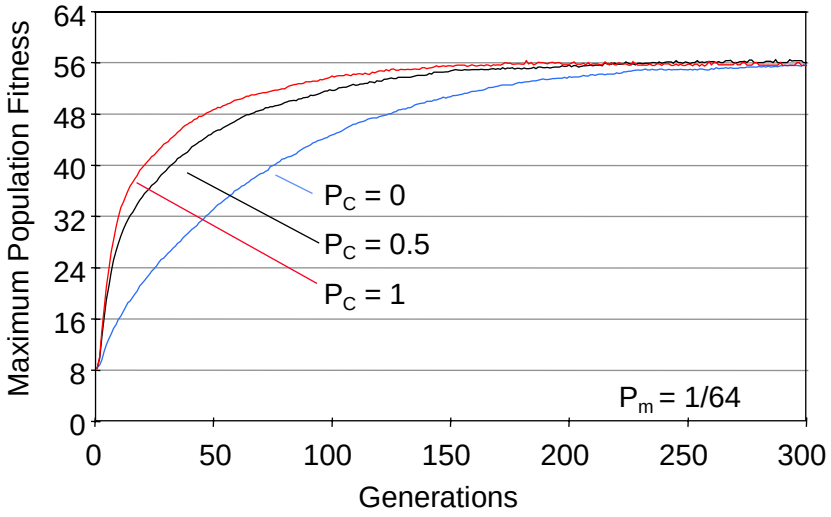


Fig 4-20 *The effect of P_c on the maximum fitness*

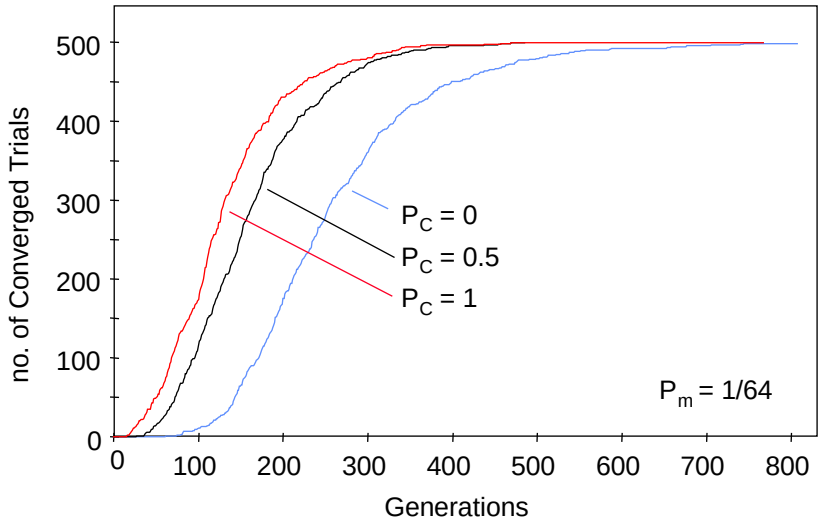


Fig 4-21 *The number of converged trials at each generation for the three crossover probabilities*

4.6 Chapter Summary

The work in this chapter has been an empirical investigation of parameters that affect GA performance. As commented in [**73**], '*there is a growing realisation that results obtained empirically are no less valuable than theoretical results*'.

What has been concluded is summed up in [**74**], '*From a function optimization point of view, GAs frequently don't exhibit a killer "instinct" in the sense that, although they rapidly locate the region in which a global optimum exists, they don't locate the optimum with similar speed*'.

This 'killer instinct' has been shown to be dependent on the mutation rate, which is critical for efficient GA performance in global optimisation. In humans, characteristics of individuals that enable them to stand out from the norm are often a result of mutation. This is exemplified by Veikko Hakulinen, a Finnish cross-country skier who won medals in the 50k, 30k, 15k and 4x10k relay at the 1956 winter Olympics. On medical examination it was found that he had an excessive red blood cell count that enabled him to take in more oxygen and not become out of breath. This was caused by a genetic defect with a probability of occurring equal to that of picking a specific light bulb with all the light bulbs on earth to choose from.

There has been much theoretical academic work in trying to improve the efficiency of GAs by optimising parameter settings. In other work, previously claimed 'good' settings are taken and used on totally unrelated problems. If GAs are to be used for function optimisation then a thorough investigation of the parameters is required.

It must be remembered that '*Genetic algorithms are NOT function optimizers*' [**74**] and that other techniques do exist, that, although they do not sound as interesting, may be more appropriate for solving a particular class of problem. Optimising a system where there is no information on the dynamics ('black box optimisation') is essentially a directed random search, with the direction being guided by the strategy used. The purpose of these strategies is to guide the search to increase the probability that in time, a solution will be found. As was demonstrated (see Table 4-5), on average over many trials, random mutation hill climbing behaves exactly as a Markov chain analysis predicts. Nix and Vose [**75**] performed a similar Markov chain analysis for a simple genetic algorithm and claim that '*if the finite population is sufficiently large, we can accurately predict the convergence behaviour of a real GA*'.

Along with GA's, simulated annealing [**76**] is another popular strategy for 'black box' optimisation that is inspired by nature. This is based around the fact that close temperature control must be maintained when cooling liquids into solids in order to attain a specific lattice structure. The most energy efficient lattice structure is obtained by very slow cooling and sometimes slight heating. This is reflected in the optimisation by only applying slight random perturbations and limiting the 'temperature gradient' (the amount of improvement allowed in new solutions). Successive solutions are also allowed to be 'hotter' (or worse) than previous attempts.

Many other optimisation strategies exist [**77**], including and tabu search [**78**] and branch and bound [**79**] (branch and bound methods are not strictly black box since they rely explicitly on the cost structure of partial solutions [**80**]).

In conclusion '*for any algorithm, any elevated performance over one class of problems is offset by performance over another class*' [**80**].

In chapter 5 a variation of RMHC is used for the optimisation of a domestic hot water tank based on real-time pricing of electricity.

5

Domestic Hot Water Optimisation

5.1 Introduction

The objective of this thesis is to develop control strategies for electric thermal storage (ETS) systems under real-time pricing tariffs. The ETS devices under consideration are domestic hot water tanks and storage radiators. In the previous chapters the tools that are to be employed were investigated and chapters 5,6 and 7 evaluate the effectiveness of these tools in both simulation and actuality.

In this chapter the charging schedule for a hot water tank is optimised. Computer simulations using actual consumption data compare the real costs of an optimised schedule and existing charging schedules. Eleven houses are simulated for one month.

In chapter 6 the controller for a storage radiator is simulated. This uses a similar optimisation method to that used for the hot water tank, but introduces neural networks as a means of creating a thermal model from which to evaluate the candidate charging schedules. A ten week simulation compares the performance of the learning-optimised strategies to that of existing control options.

Finally in chapter 7, a storage radiator in a real room is controlled using a neural model predictive controller. Data was recorded for five months and an empirical neural thermal model of the room created. This model was then used to determine control set points five hours in advance to track a given room temperature profile, but with no optimisation. The controller was in continuous operation for 2 weeks.

Optimising ETS devices has been widely studied from various perspectives. In [**81**] the approach taken is to centrally control the water heating of blocks of houses, the main objective being to reduce peak load, a utility benefit. In [**82**] storage radiators are optimised for cost and comfort but using time-of-use (ToU) tariffs, genetic algorithms and a resistance-capacitance (RC) building thermal model.

Neural networks have also been used to model building energy consumption [**83**,**84**]. In [**85**] a recurrent neural network was used to model a crèche with a heated floor. The objective here was to optimise the start-up time so as to minimise energy consumption. A particularly ambitious project for using neural networks for domestic control is outlined in [**86**], where a house has been ‘computerised’. Optimising the heating control is being attempted in simulation [**87**] but the initial work only used neural networks to predict occupancy with a RC model used to predict the building response. The planning horizon is 120 minutes.

Model predictive control using neural network empirical models rather than first principle models has been attempted in simulation mainly for the chemical process industries [**88**,**89**].

Any controller that is developed will ultimately rely on communication so that it can receive price and weather information. There will also be the need for half-hourly meter-

ing if real-time tariffs are to be introduced. Such technology is already available and under trial in domestic houses [**90**]. Actual experiments in the logistics and hardware requirements of real-time control for thermal storage have been performed as far back as 1989 [**91**,**92**].

This thesis is concerned with the development of control technology that is required to make real-time pricing feasible. An analysis of such tariffs is not given but sources for reference are [**1,2,3,4**,**93**,**94**,**95**,**96**,**97**]. What has to be considered is that using a prediction of the next day's demand sets the daily pool price. If this demand has the potential to adapt to the set price then the initial forecast is wrong. Will this have the desired effect of flattening the demand profile?

5.2 Model to be Optimised

Fig 5-1 shows the water heating system to be optimised. For each half-hour period there is a demand (litres) and price (pence/kWh), profiles of which are given at midnight for the following 24 hours. The criterion to be satisfied is that the demanded water (in the 24 hours following midnight) must be supplied at a set temperature in the cheapest manner. Two heating elements exist, one in the storage tank (element 1) and one at the outlet (element 2). The latter is to ensure adequate supply temperature ($T_{required}$) and can be supplied with warm water from the storage tank via tap 1 or ambient water from the mains via tap 2. For a 24-hour period of known demand and price, the challenge is to determine for each half-hour the water source (tap1/tap2) and the state of heating element 1 (on/off) that will give the cheapest cost. Heating element 2 is not controlled but delivers the required amount of energy to maintain the delivered water at $T_{required}$.

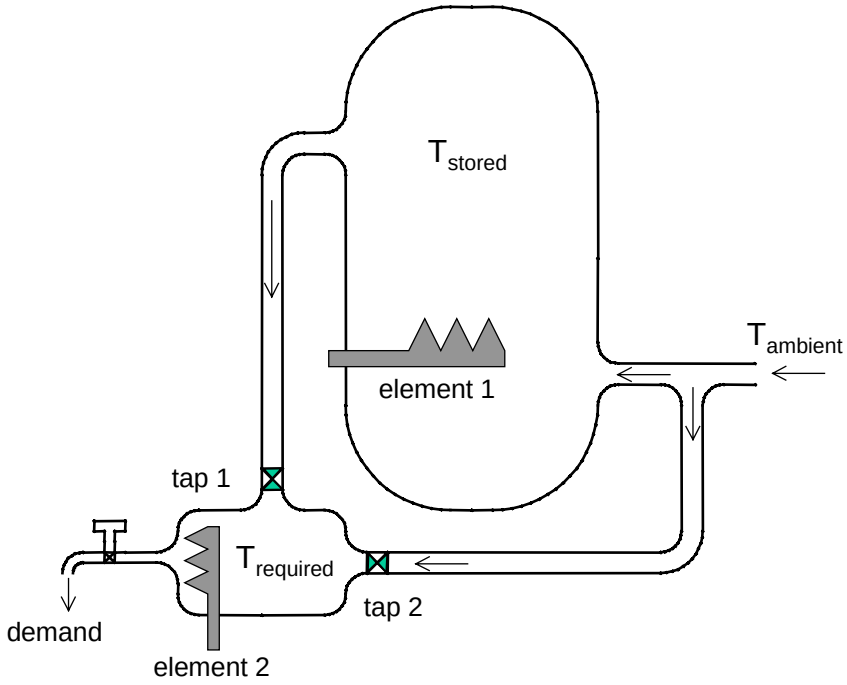


Fig 5-1 *Schematic of the hot water system*

For each half-hour two decisions have to be made,

1. If there is demand then shall the source be tap 1 or tap 2?

2. Shall the tank be charged by activating element 1?

There are thus two options for each decision. If water is consumed in all of the 48 periods during a day then there are $2^{(48\times2)}$ potential solutions, of which the cheapest is sought, as depicted by Table 5-1.

Table 5-1 *The control sequence to be optimised for the water-heating model*

time slot 1		time slot 2		::	time slot 48	
decision 1 Source?	decision 2 Charge?	decision 1 Source?	decision 2 Charge?	::	decision 1 Source?	decision 2 Charge?
tap 1 ? *or* tap 2 ?	yes ? *or* no ?	tap 1 ? *or* tap 2 ?	yes? *or* no ?	::	tap 1 ? *or* tap 2 ?	yes ? *or* no ?

5.3 Simulated Water Heating Model

A simplified computer model of Fig 5-1 was created to determine the fitness of each candidate solution, which is the total cost over the 24 hour period (see appendix E for an example of the code used). The process was continuous in that the final tank temperature (time slot 48) was used as the starting temperature for the following day. The absolute accuracy of the model compared with a real hot water tank is not vital since the comparative costs of the existing schedules are being simulated from the same model. Several assumptions and rules were made to simplify the model,

1) No heat loss from the tank.

2) Complete mixing of water in the tank so it is always at a uniform temperature.

3) All demand is given instantaneously at the start of each half-hour, charging commencing on the recalculated tank temperature.

4) Charging stops when the tank water reaches the set point (demanded) temperature (70° C).

5) All water being delivered is topped up to the set point temperature by the direct acting electrical element (element 2) costing whatever the price is in that specific half-hour.

6) In the simulations for the existing charging profiles (E7 and E10) all the water was delivered from the tank via tap 1 and extra heat was added from heating element 2 if it was below the required temperature.

The E7 charging profile used is 00:00-07:00. The E10 profile is 02:30-07:00, 12:30-15:00 and 19:00-21:30. Element 1 was set at 2kW. Although heating element 2 is generally not present in domestic tanks it is required so that a fair comparison can be made between the charging schedules, as it ensures all schedules deliver water at the required temperature so that comfort is guaranteed.

5.4 Data Used

The data (purchased commercially) used in these simulations originated from 100 houses monitored over the course of a year. Each water outlet was logged every half-an-hour and from this all hot water outlets were grouped to find the total hot water demand in each half-hour period. There was no indication in the data of how the water was actually heated. Eleven houses were randomly chosen for simulations, which were performed for November 1994, with the corresponding actual pool selling price (PSP) used to calculate the cost.

Fig 5-2 shows actual hot water consumption and PSP over four days for a particular house. It can be seen that there are small price peaks just after midnight caused by the surges due to the existing E7 and E10 tariffs. This is even more pronounced on Saturday when the early morning price is almost as high as the maximum price for that day. The difference between weekdays and weekends can also be seen, with the weekend price generally lower because of reduced overall demand. The high peak on Thursday occurs at evening meal time and is a result of increased domestic heating, lighting and cooking electricity consumption. The water consumption tends to be concentrated between 8am and 10am that can be a period of high price. The consumption pattern on Sunday is spread throughout the day, highlighting how usage is related to lifestyle.

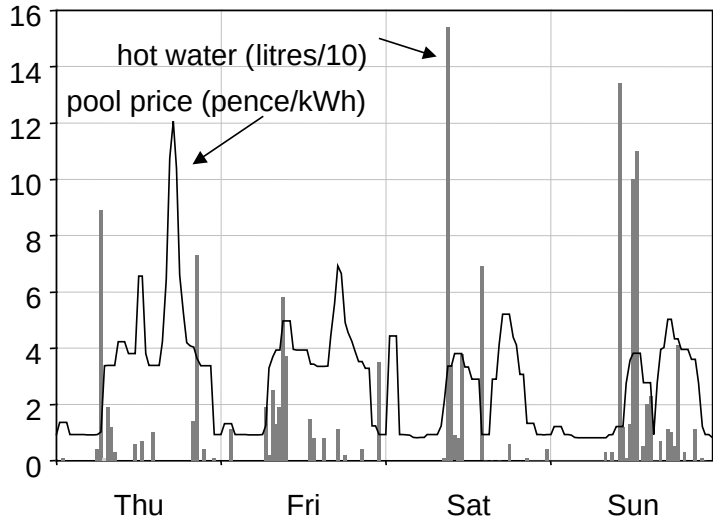


Fig 5-2 *Actual pool price and hot water consumption from a random house for four days in November 1994*

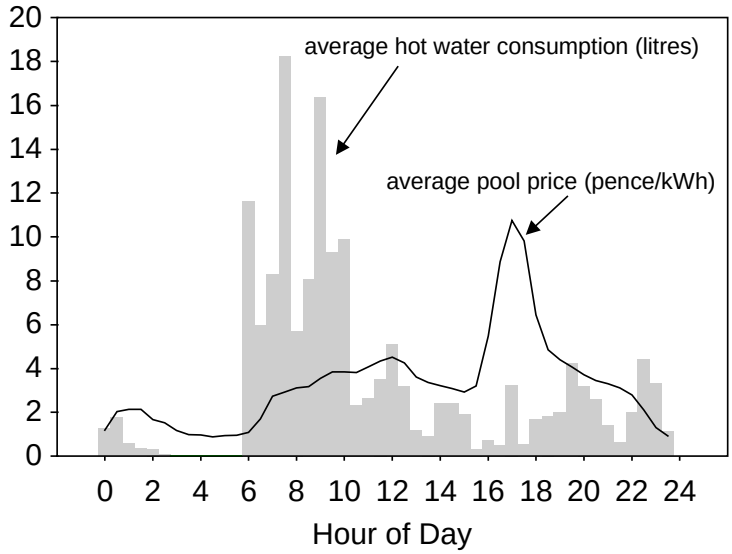


Fig 5-3 *The November daily averaged price and consumption for the same house*

Fig 5-3 shows the average daily consumption profile for the same house throughout November and the corresponding average price. The morning water consumption is emphasised (hours 6-10), as are the early evening and midnight peaks in pool price.

These two figures show that although on the average things look predictable, on a day-to-day level there is much variation and potential for customised control strategies.

5.5 Optimisation Procedure

The optimisation technique used was based on random mutation hill climbing, as described in section 4.5.3.3 on page 86. As well as proving superior in performance to GAs it is also more desirable from a controller memory standpoint as only two solutions have to be stored, as opposed to many if GAs were used.

In the original version of RMHC only one bit change is allowed between successive potential solutions, which means that it is unlikely to escape from any local minima. Three bit changes were introduced to overcome this potential problem. Introducing more than one bit change also has the effect of speeding up the process. This is because if there is no demand in any particular half-hour then the choice of tap 1 or tap 2 is irrele-

vant and a bit flip will make no change to the solution. Pre-processing the string to eliminate redundant bits would reduce the search space but require more processing power.

In any stochastic (i.e. having an element of chance) search procedure, there is no guarantee that the global optimum solution will be found. Once a solution had been given adequate time to reach a steady value, it was found more beneficial to restart the search as opposed to continue searching from the current position. In the optimisation the search was repeated three times with the best overall solution used. Each search consisted of 2,000 evaluations, with the whole process taking about 7 seconds on a P133 to optimise all 30 days.

The hot water tank is an example of a system where one change can have a profound effect on the outcome. If the tank is at its maximum temperature then the thermostat in the model will ensure that no more heating is allowed, regardless of the control signal to the element. For instance, if there is no demand all control signals for heating the tank would be ignored once it was at its maximum temperature. A change early in the day could result in a previously ignored signal becoming active. This makes the search more random rather than gradient based. To overcome this, all signals indicating that the tank should be charged were reversed if the tank was already at its maximum temperature.

The initial starting point can affect the search procedure, especially if there is low demand throughout the day. To capture this possibility the initial guess is ‘do not charge the tank at all’, for which the associated cost will be that of using direct acting heating to satisfy the requirement. This is often the cheapest solution if there is low demand as it saves heating the tank and having excess hot water at the end of the day.

5.6 Profiling Usage Patterns

In the optimisation process the actual half-hourly consumption data was used. In reality, the controller will have to use estimated values on which to base the optimisation. This

could be done via a keypad, with the occupants entering times at which they are likely to take showers, baths or use washing machines. An alternative method is to use past consumption patterns to make educated guesses as to a likely profile for the following day.

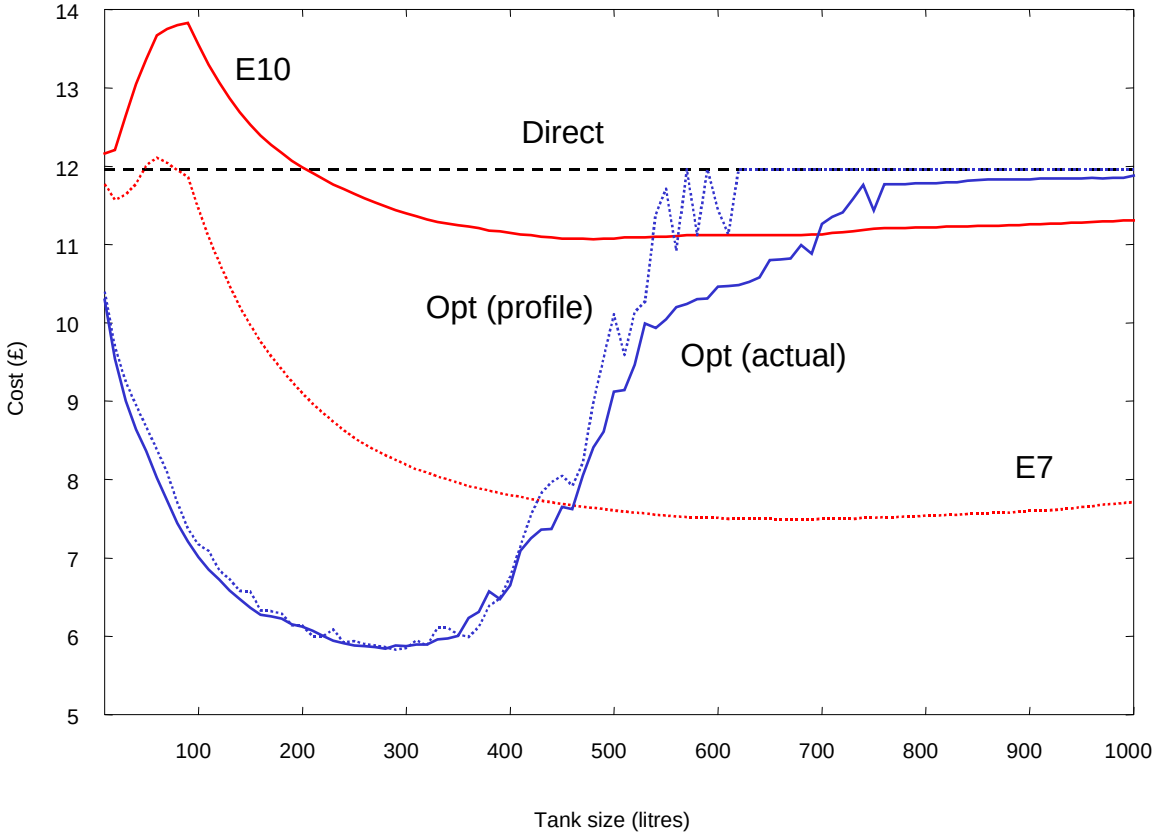
If consumption is to be predicted based on previous occurrences, it has to be assumed that there is some cyclical pattern involved. This is likely to be a predominant daily cycle with an underlying weekly cycle, which life generally revolves around.

A simple method of predicting consumption is to take an average value of volumes that occurred in the same half-hour of previous weeks. The method actually employed was to use a neural network to create a curve fit with daily and weekly components. This was achieved by having inputs representing hour-of-the-day and day-of-the-week, appropriately coded as sines and cosines in order to achieve the cyclic pattern. Each unique combination of inputs thus had four output values for the four weeks of data available, an ill-posed problem. This has the effect of basically averaging the consumption but fitting a generally smooth curve through the data, achieved by limiting the number of hidden neurons. 15 hidden neurons were used in this case.

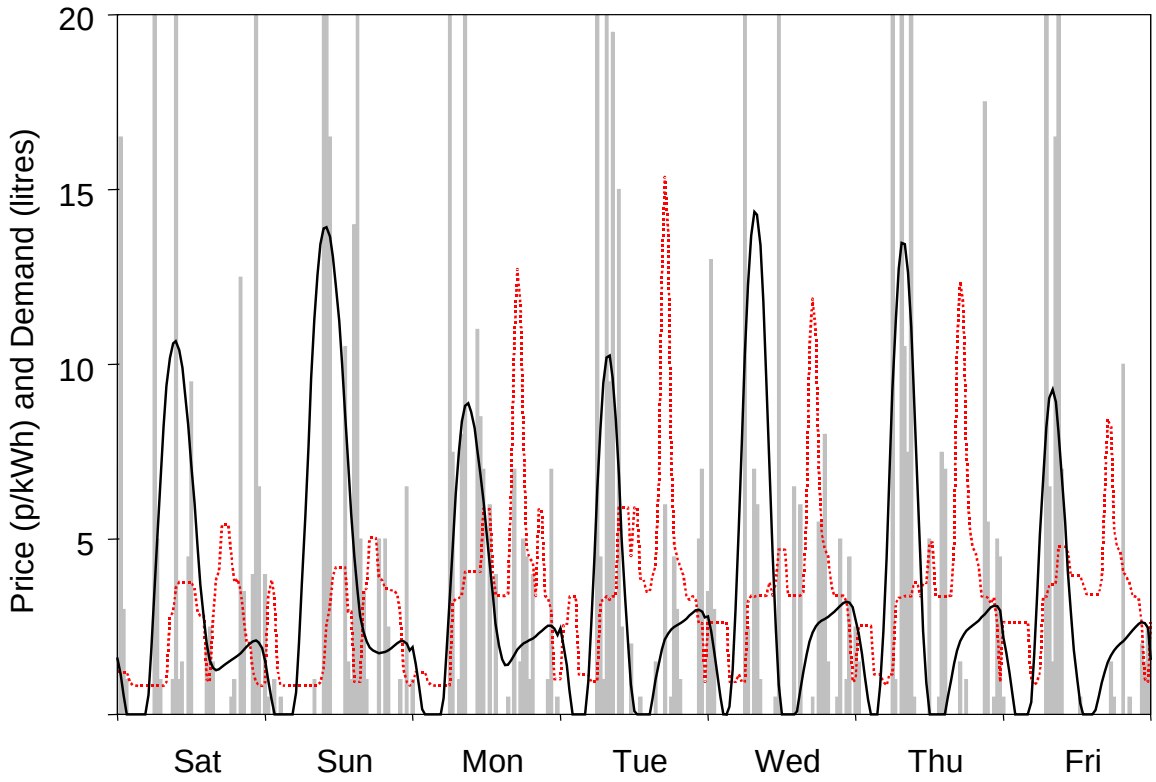
The resultant profiles were used to optimise the heating system and the costs calculated by then using the actual consumption patterns. Because only four weeks of data were used the actual consumption figures for any half-hour contribute to the predicted profile. A more realistic test would be to use a running profile and use it for the week ahead, with the prediction day's data not being involved in creating the profile.

5.7 Results

Consumption data from 11 houses for November 1994 was simulated for boiler sizes of 10 to 1000 litres. Costs for E7, E10 and direct acting only heating strategies were also recorded. The results are shown Fig 5-4 to Fig 5-14. The actual demands in b) are truncated at 20 litres and the x-axis in a) starts at 10 litres.

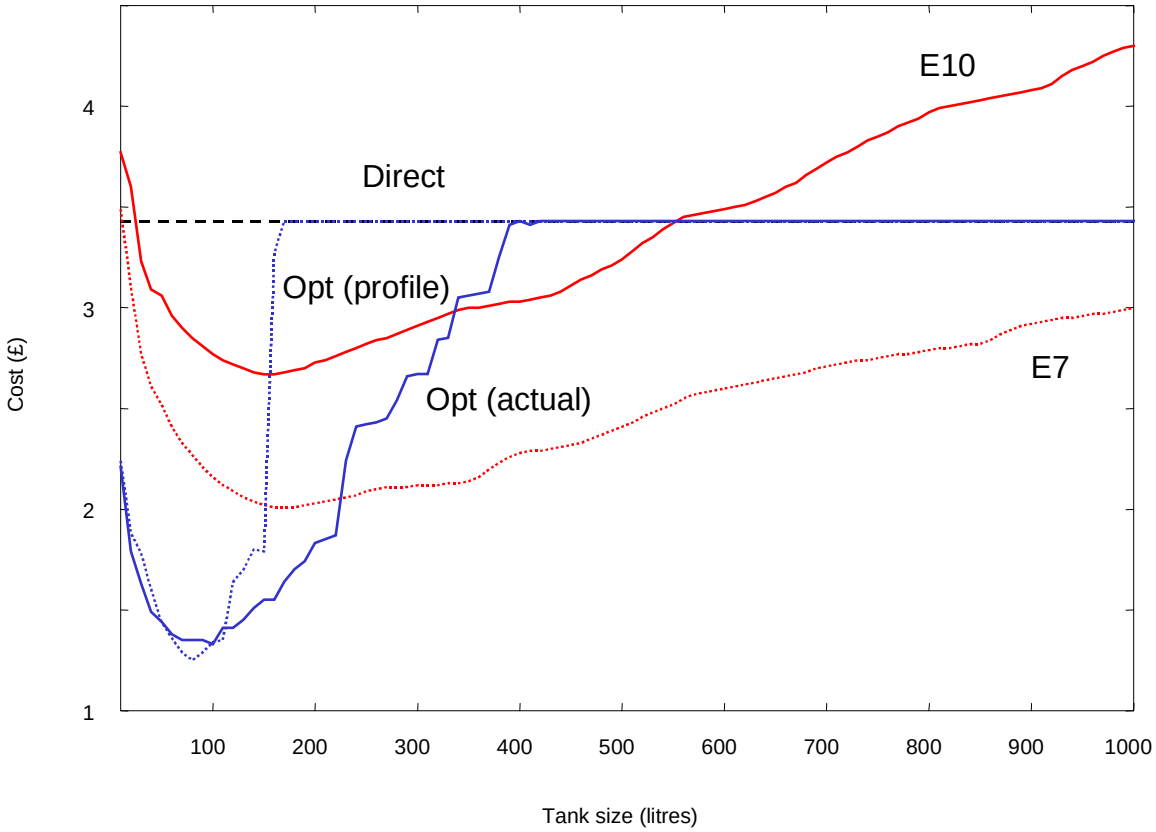


a) *Costs as a function of tank size*

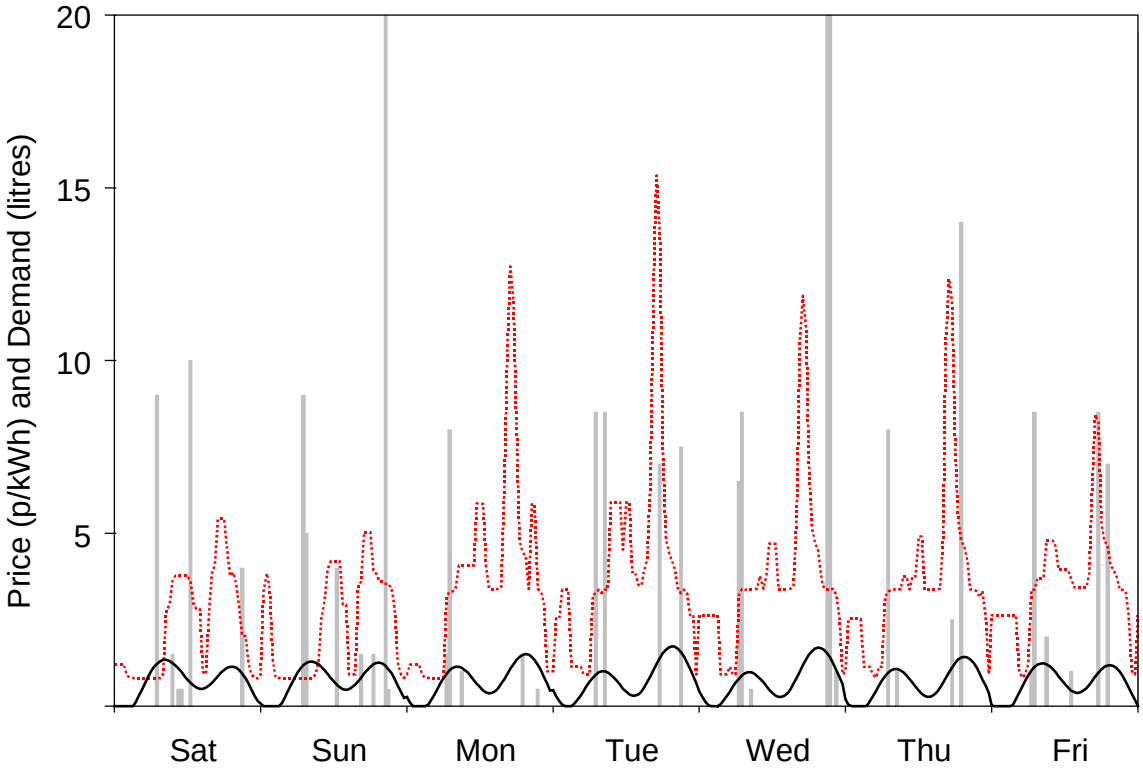


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-4 HOUSE 1 mean daily demand = 157 litres

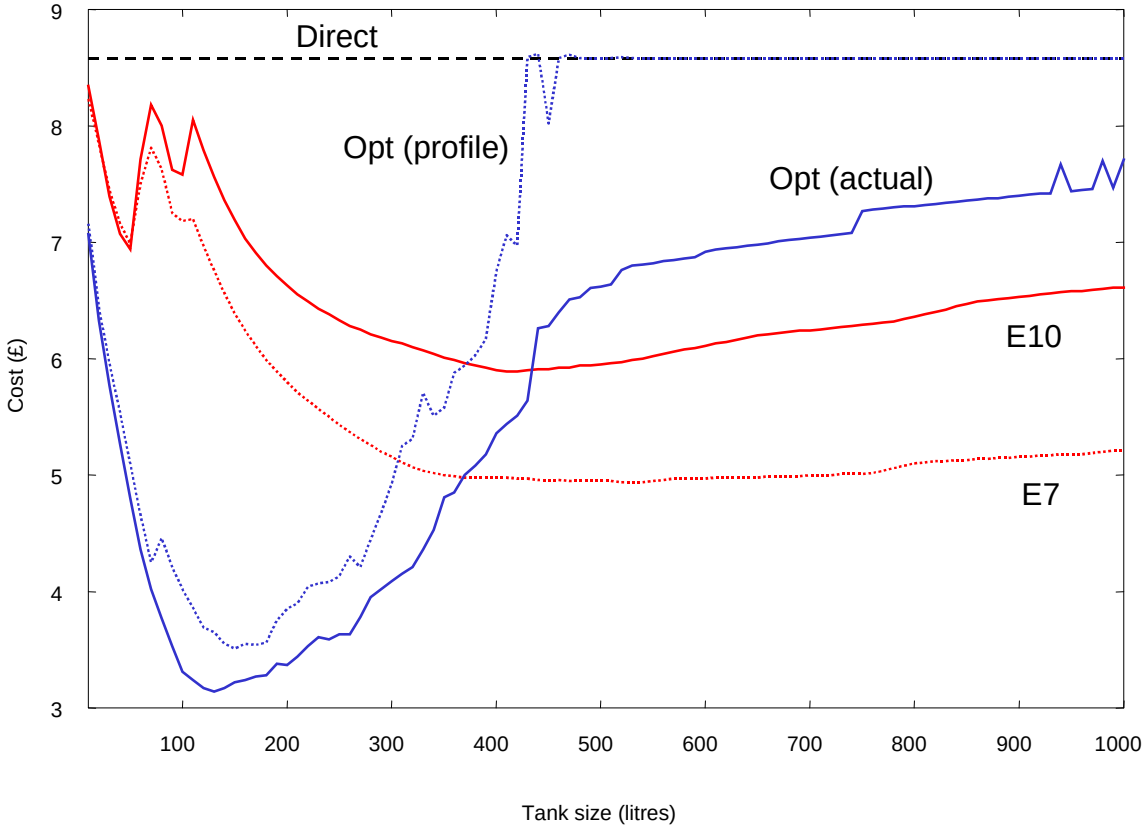


a) *Costs as a function of tank size*

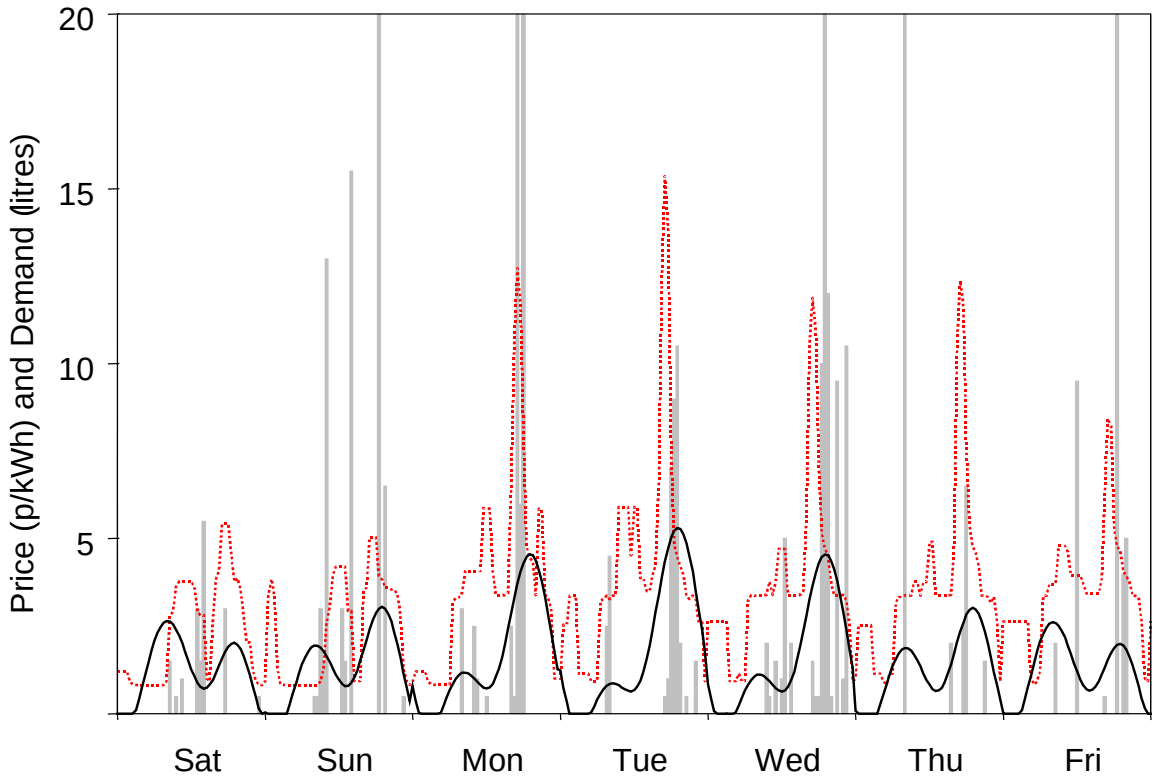


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-5 HOUSE 2 mean daily demand = 36 litres

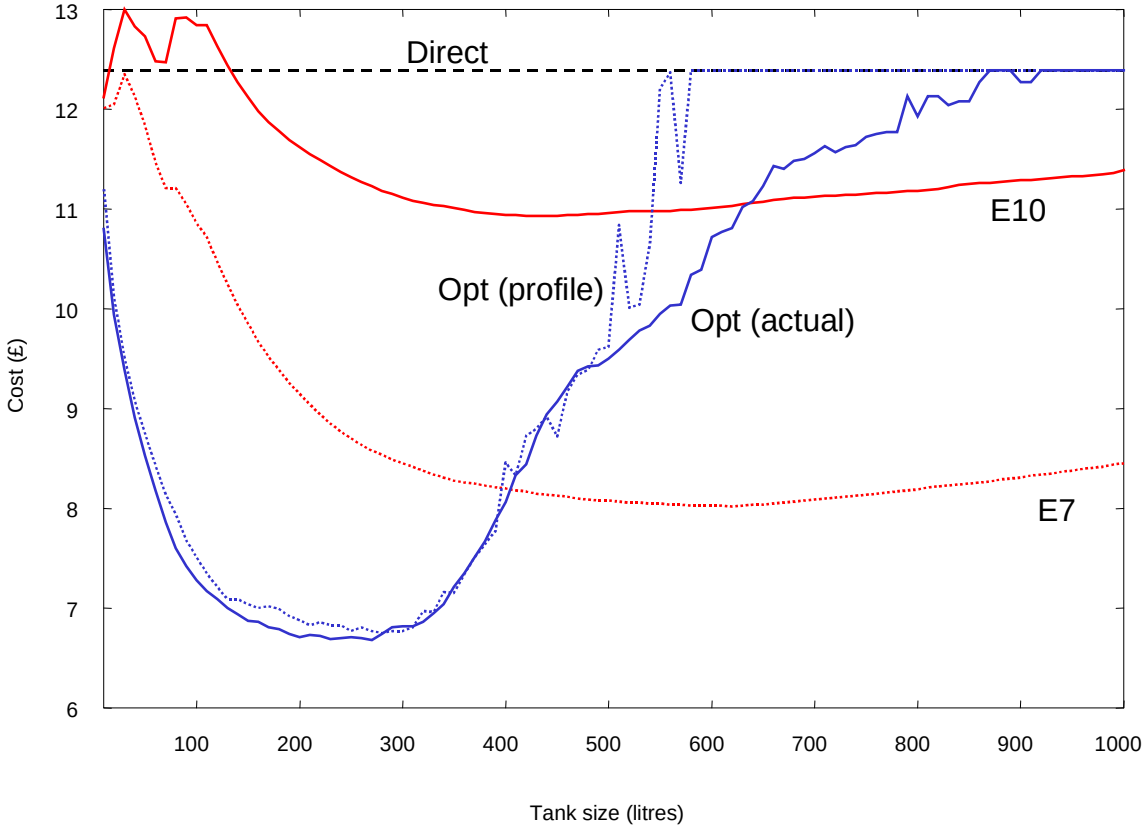


a) *Costs as a function of tank size*

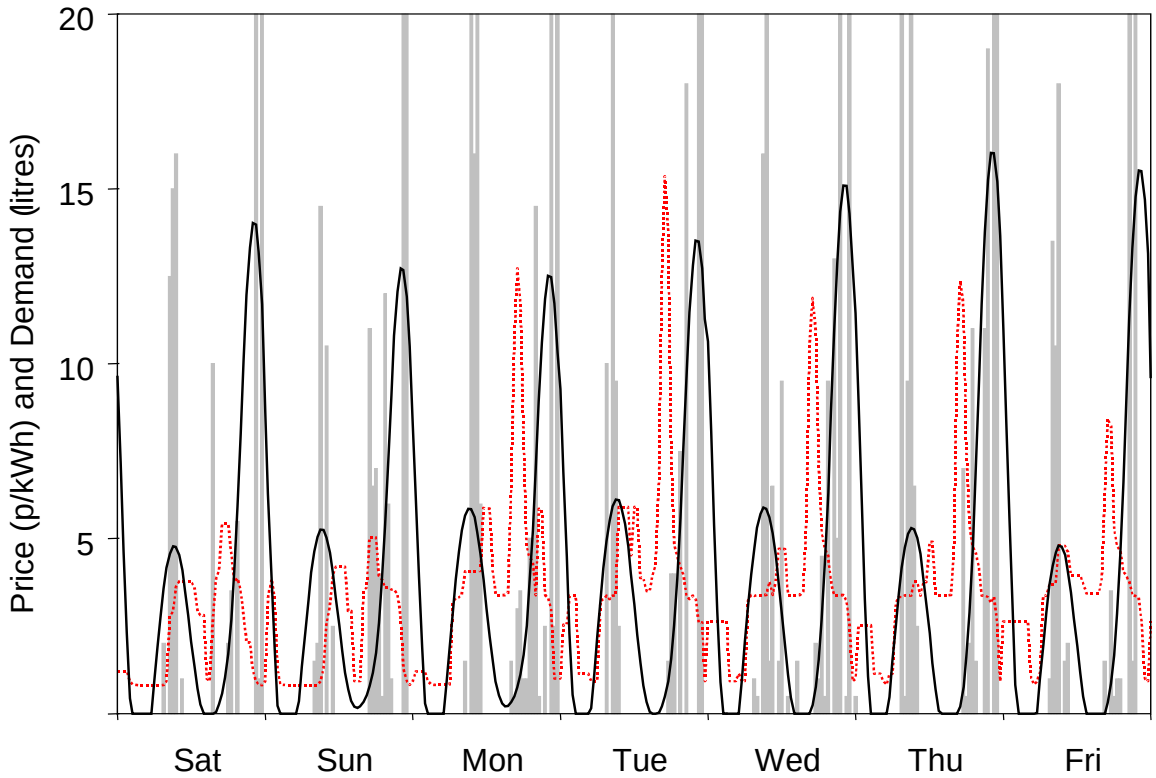


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-6 HOUSE 3 mean daily demand = 74 litres

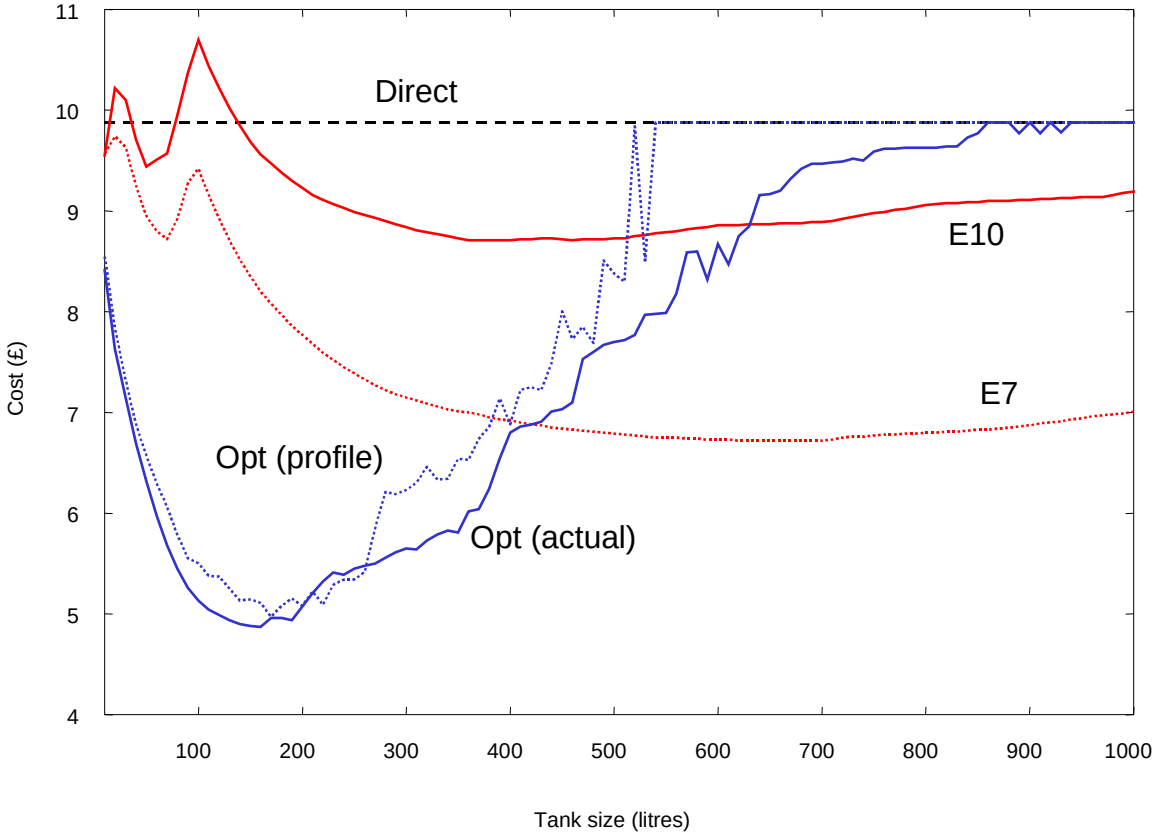


a) *Costs as a function of tank size*

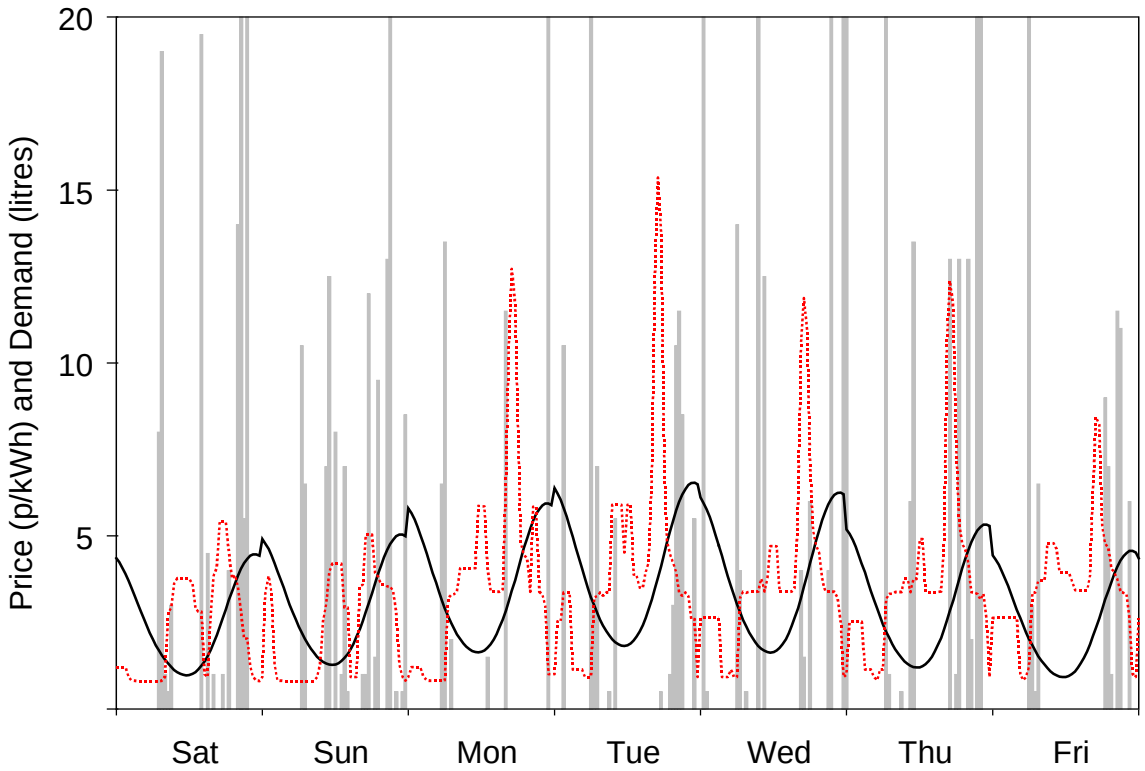


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-7 HOUSE 4 mean daily demand = 186 litres

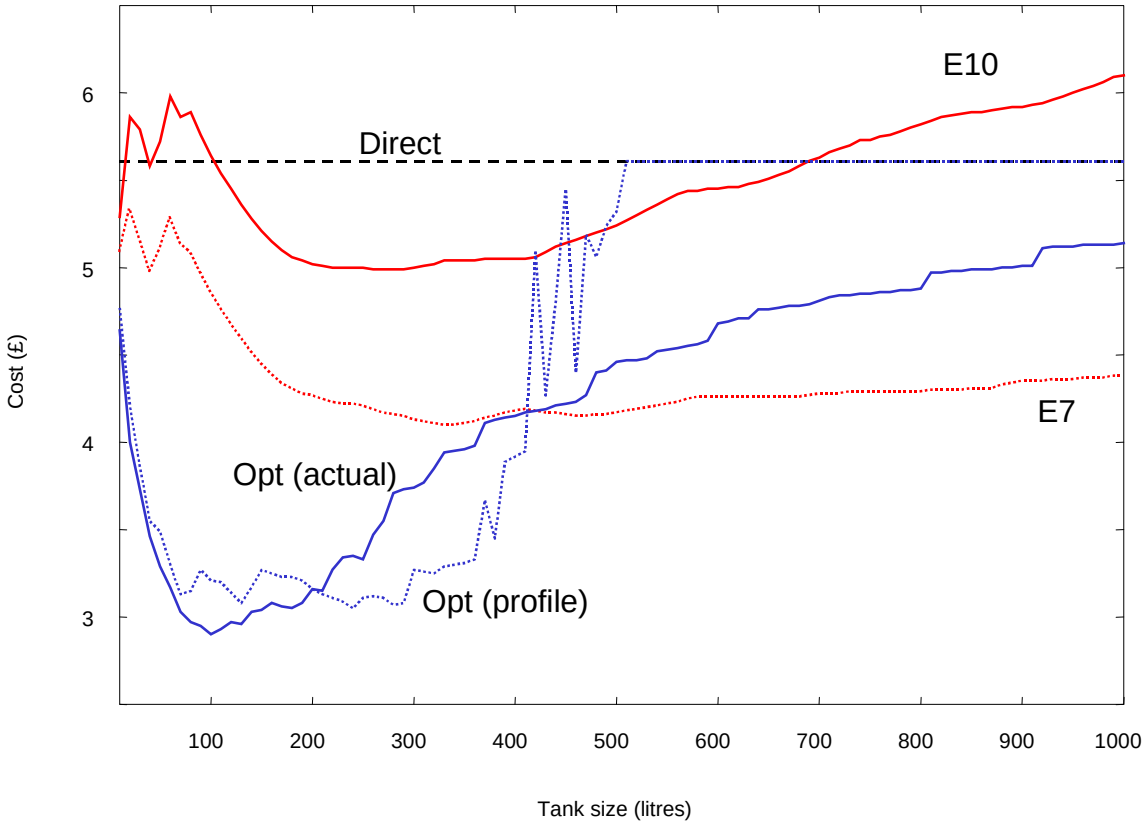


a) *Costs as a function of tank size*

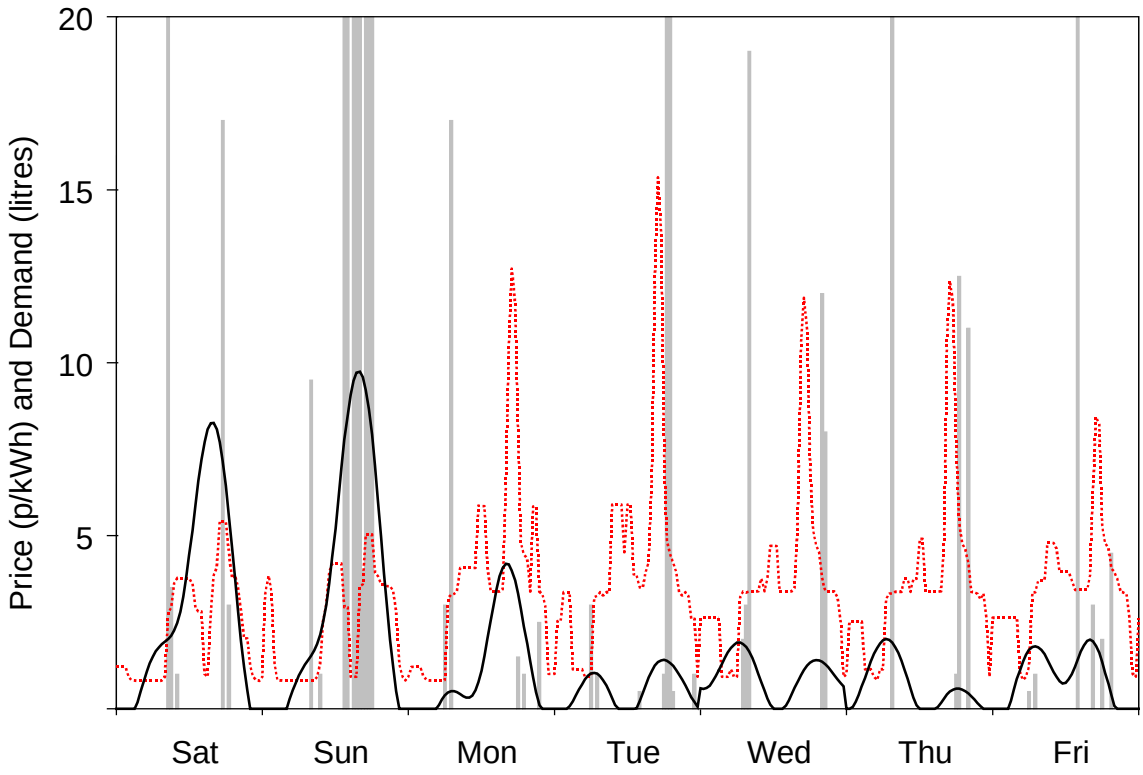


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-8 HOUSE 5 mean daily demand = 150 litres

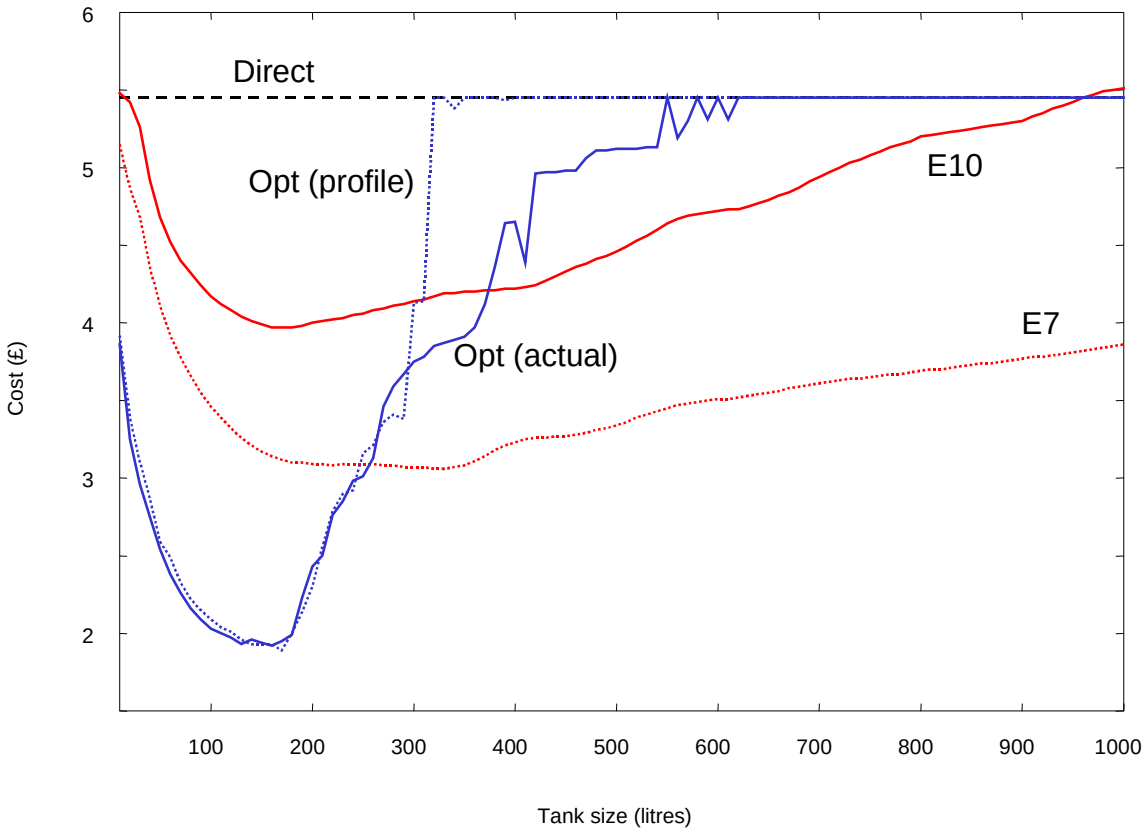


a) *Costs as a function of tank size*

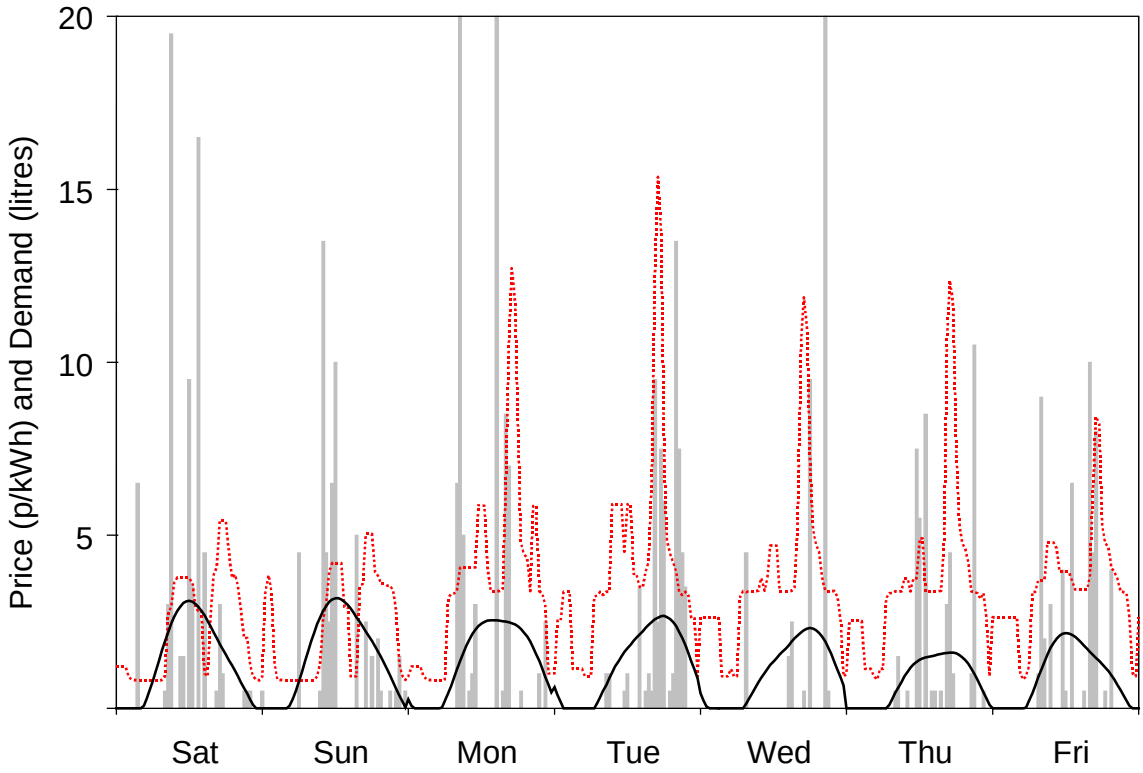


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-9 HOUSE 6 mean daily demand = 71 litres

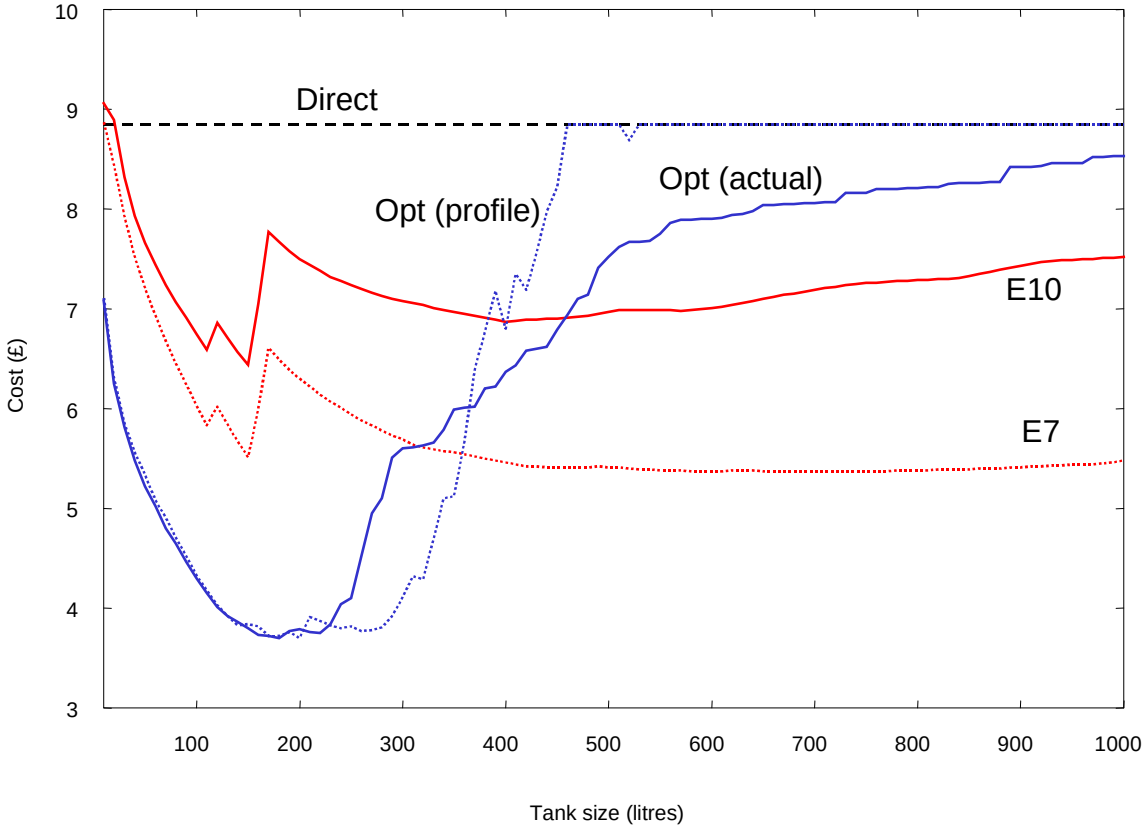


a) *Costs as a function of tank size*

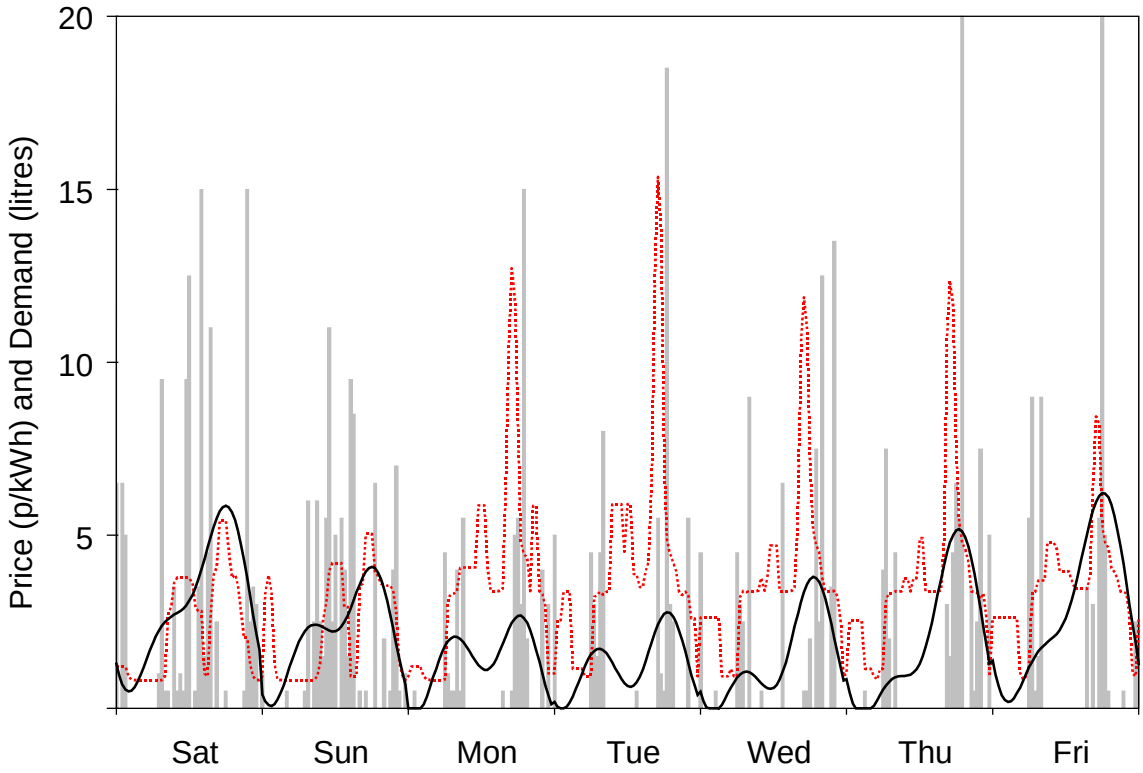


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-10 HOUSE 7 mean daily demand = 55 litres

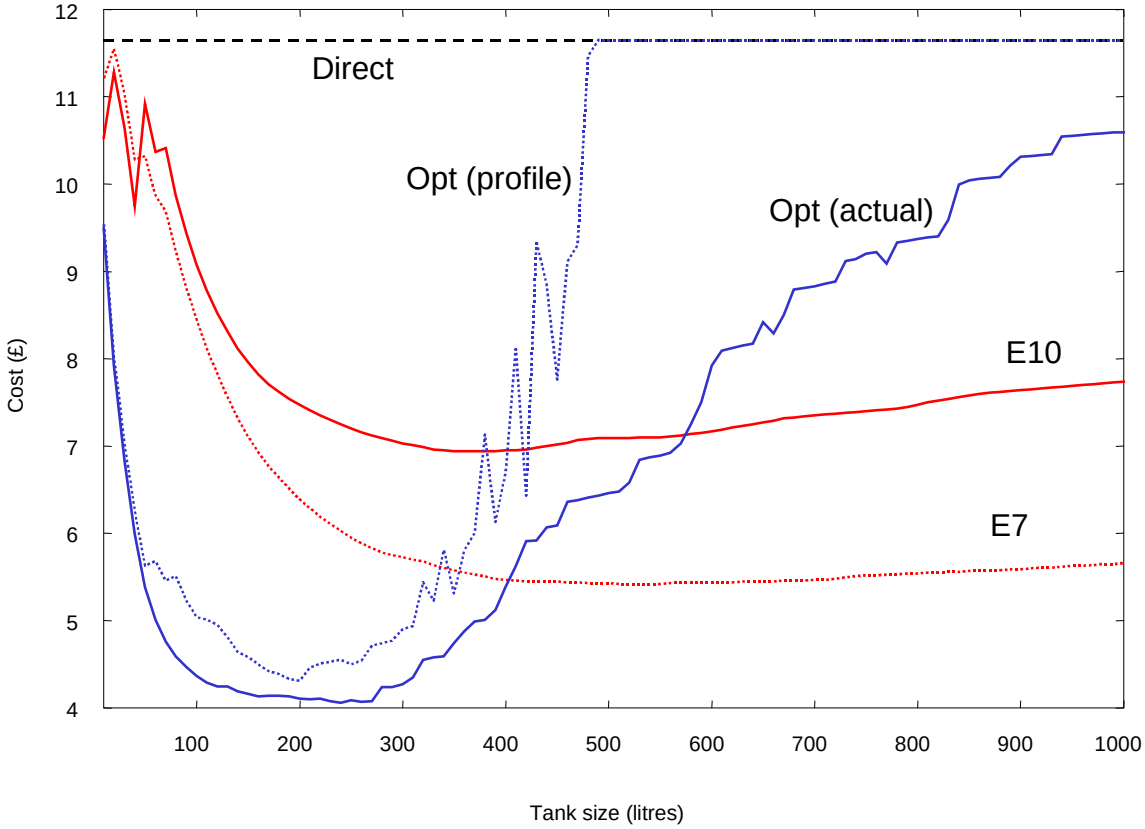


a) *Costs as a function of tank size*

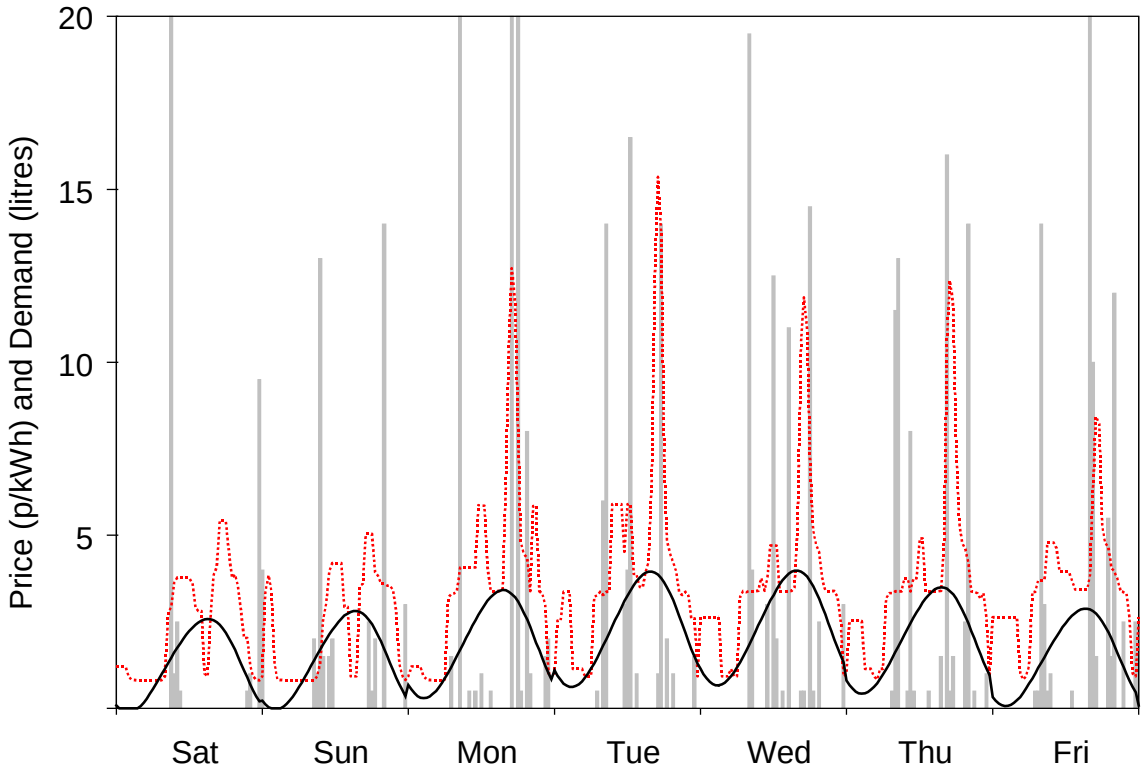


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-11 HOUSE 8 mean daily demand = 96 litres

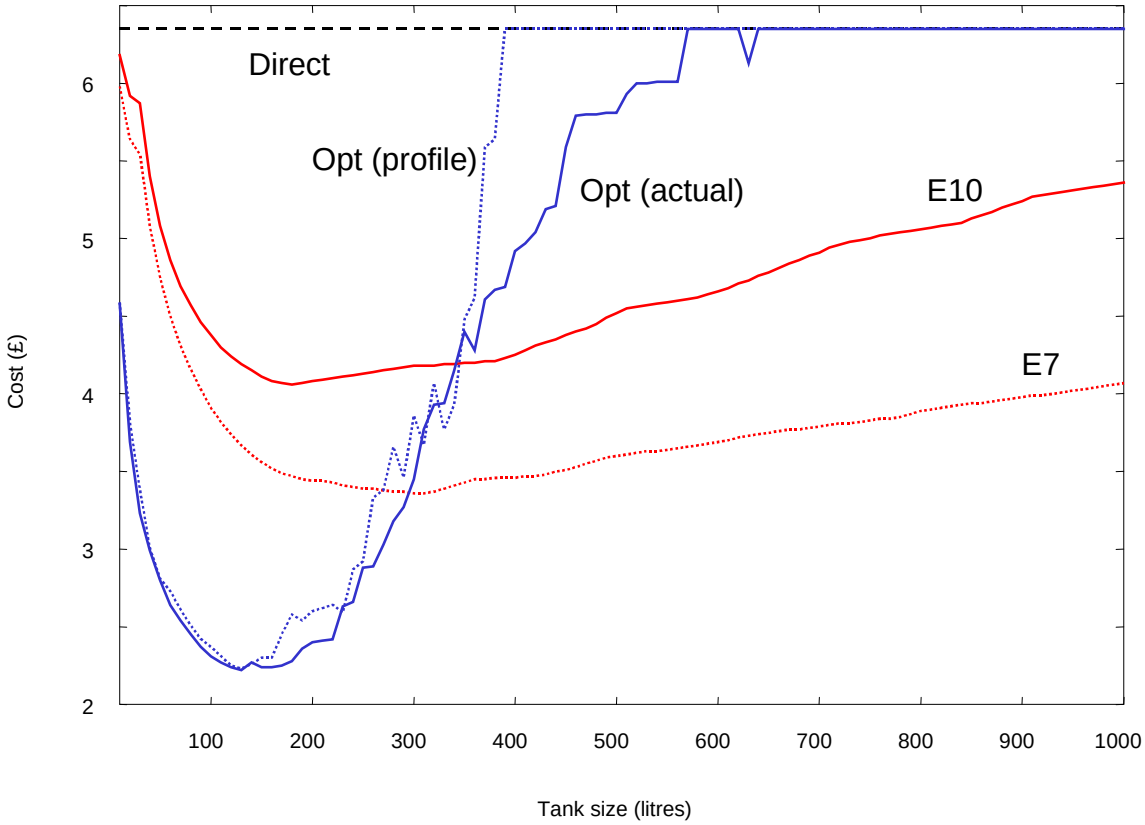


a) *Costs as a function of tank size*

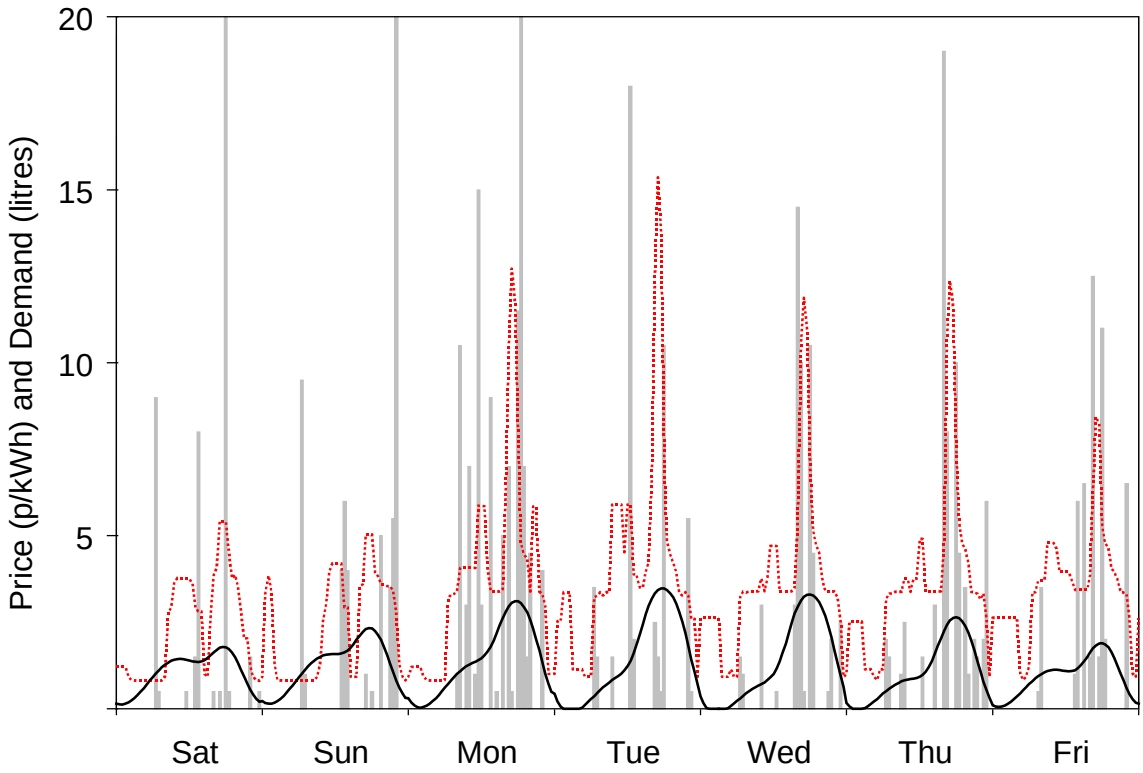


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-12 HOUSE 9 mean daily demand = 86 litres

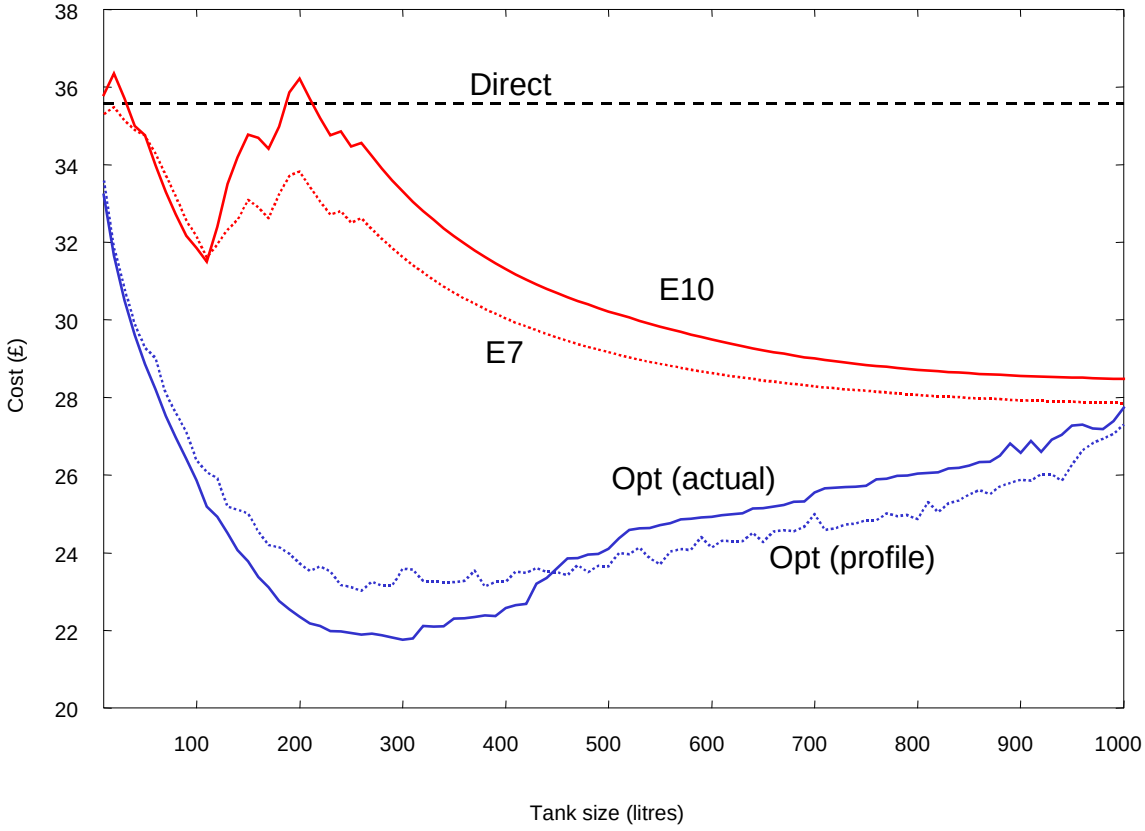


a) *Costs as a function of tank size*

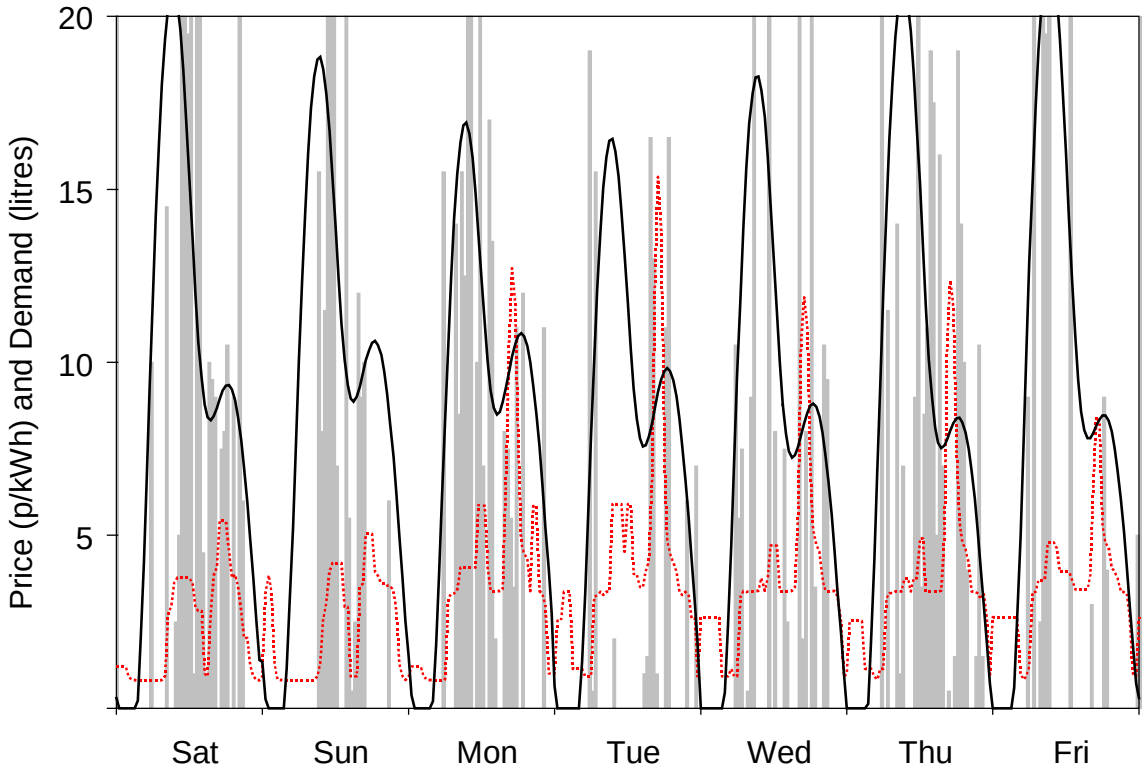


b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-13 HOUSE 10 mean daily demand = 57 litres



a) *Costs as a function of tank size*



b) *The first weeks pool price (dotted) and actual consumption (bars) and the weekly profile (solid)*

Fig 5-14 HOUSE 11 mean daily demand = 395 litres

5.8 Discussion of Results

5.8.1 Does Water Storage Save Money ?

If storage tanks did not exist then the water must be heated on demand at the cost of the current pool price. The cost of this option is shown in Fig 5-4 a) to Fig 5-14 a) by the straight line labelled ‘direct’.

In all cases the E7 and the optimised charging schedules are cheaper or as cheap as direct acting heating. E10 is generally cheaper but depends on the tank size and demand levels. House 2 shows that for a very low demand excessive charging with large tanks is wasteful.

5.8.2 How did the Profiling Perform ?

Two methods of optimisation were performed. The first was to use a predicted ‘*Opt(profile)*’ daily demand and the second was to use the actual ‘*Opt(actual)*’ demand. After the optimised schedules were derived the costs were then calculated based on the actual demand.

Fig 5-4 a) to Fig 5-14 a) show that using the profiled demand compares very favourably to using the actual demand. Generally as the tanks get larger the actual demand is required to give a cheaper solution. This is because the more continuous nature of the profiled patterns will result in the tanks being over charged at times of low demand. This cannot be avoided as there is more storage capacity and hence increased energy consumption.

In the case of house 6 the profiled optimisation costs were cheaper than the actual optimisation costs for tank sizes of 200 to 400 litres. This means that the optimisation procedure did not perform satisfactorily when using the real data. By looking at the demand profile for week 1 (Fig 5-9 b) it can be seen that water is only consumed in about 5 half hours of the day, immediately making 44% of the search space redundant. By using the profiled consumption there is predicted demand in most half-hours, which

assists the search procedure. For instances similar to this the search space could be severely reduced by eliminating the redundant bits.

The individual profiles in Fig 5-4 b) to Fig 5-14 b) show a wide variation from house to house and there is no ‘typical’ profile. The profiles are similar to a smoothed time-averaged demand, and in all cases the total profiled demand was within 5% of the actual total demand.

It might be suggested that ‘group’ profiles could be created for specific users, which would alleviate the need to measure actual consumption. The groups might be related to the number of residents, but as Fig 5-15 shows there is no obvious relationship between the number of residents and average daily consumption.

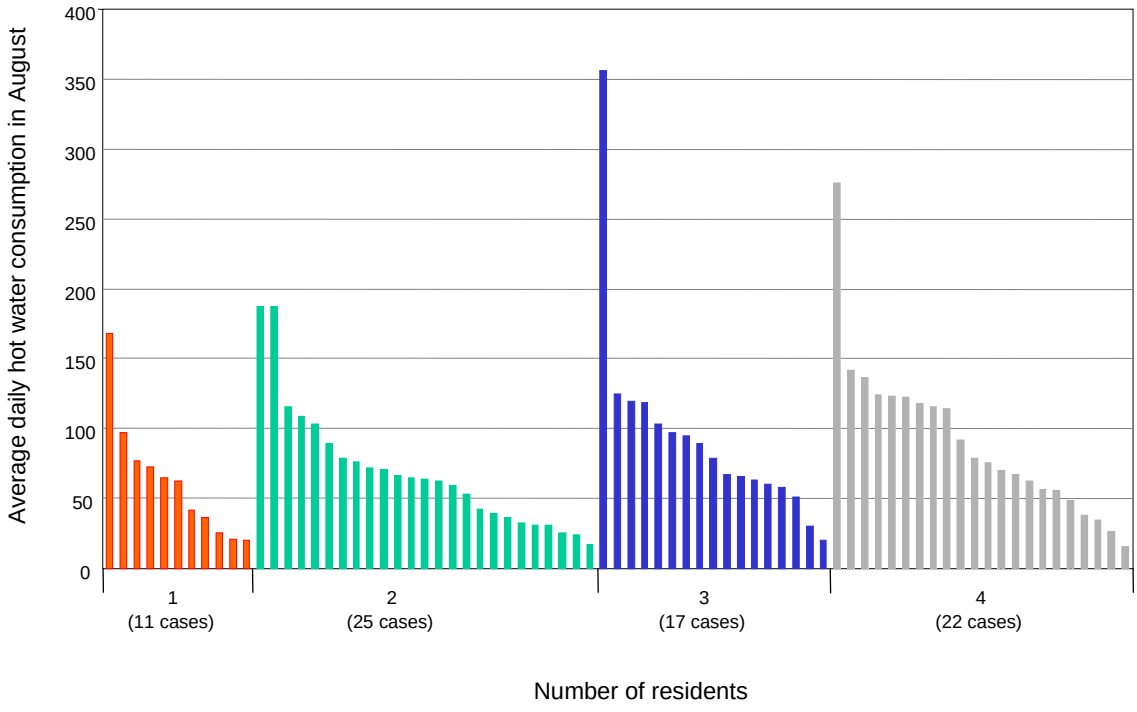


Fig 5-15 *Mean daily hot water consumption related to the number of residents*

5.8.3 How Much Money could be Saved?

Fig 5-16 shows the percentage cost savings over E7 for the 11 houses. The data is from the optimisation results using the profiled consumption patterns, which is close to what

could be achieved in reality. For most houses savings of between 20-40% are possible for tank sizes between 50 and 250 litres. If it is assumed hot water accounts for 40% [**81,90**] of the domestic electricity bill then this relates to savings in the range 8-16%. Existing domestic tank capacities are within the range 100-250 litres.

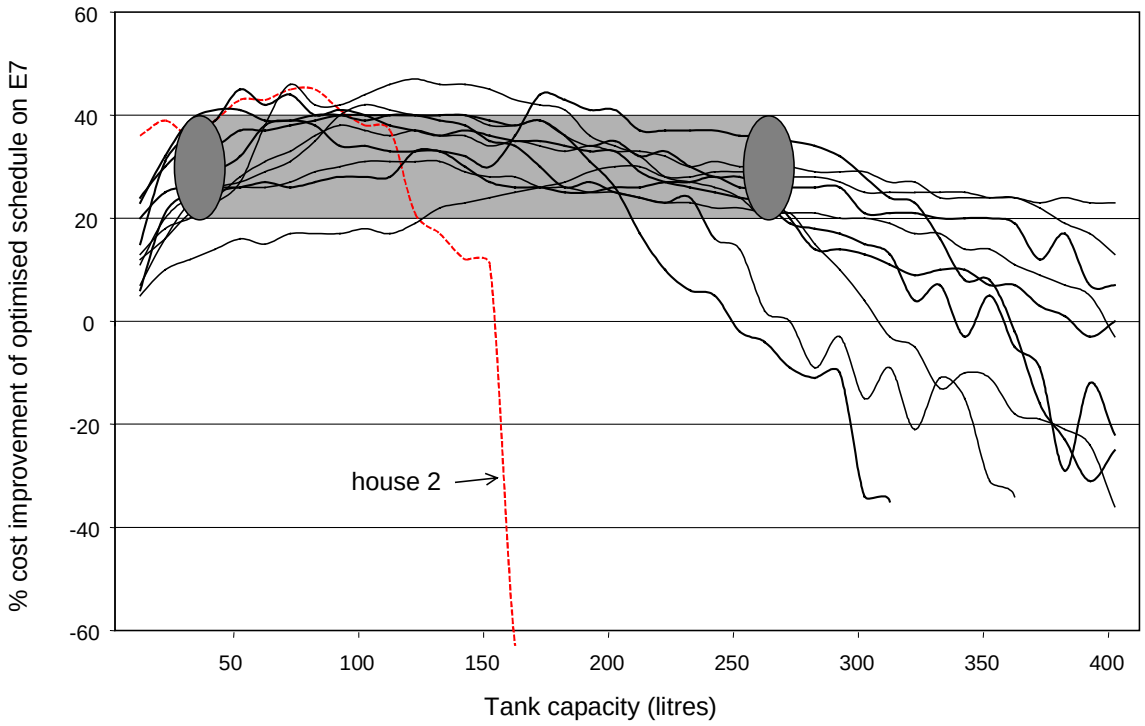


Fig 5-16 *For tank capacities between 50-250 litres the optimised schedules show consistent savings between 20-40% compared with E7.*

5.8.4 Why is the Optimised Schedule Sometimes Worse?

For tank capacities over 300 litres the relative performance of the optimised schedules degrade compared with E7. For house 2, which has very low consumption, this decline starts at 150 litres (Fig 5-16). E7 is outperforming the optimised schedule, so why did the optimiser not arrive at a schedule similar to E7?

The reason for this is the balance between tank size, consumption and the optimisation process. In the system used the optimisation window was 24 hours and the schedule was calculated once per day. With larger tanks better performance will be achieved by increasing the optimisation window. A tank of 600 litres can typically hold enough hot water for three days consumption. An optimal 24 hour schedule will probably not involve charging the tank as it can be wasteful because of excess heating that is not required within that 24 hour period. Similarly for low demand a shorter window or continuous optimisation would result in improved performance.

Fig 5-17 shows the relationship between the mean daily water consumption and the tank size resulting in the cheapest cost for the E7 and optimised schedules. The optimised schedules are a significant improvement and suggest a range within currently available tank sizes.

It is interesting to note that the optimum E7 tank size has 5 times the capacity of the average daily requirement. By keeping a large volume of hot water the daily temperature reduction is small, so less input will be required by element 2. Introducing time dependent heat losses into the model would give a more realistic situation. In reality the system behaviour is not like that of the model, as water is not always delivered at the required

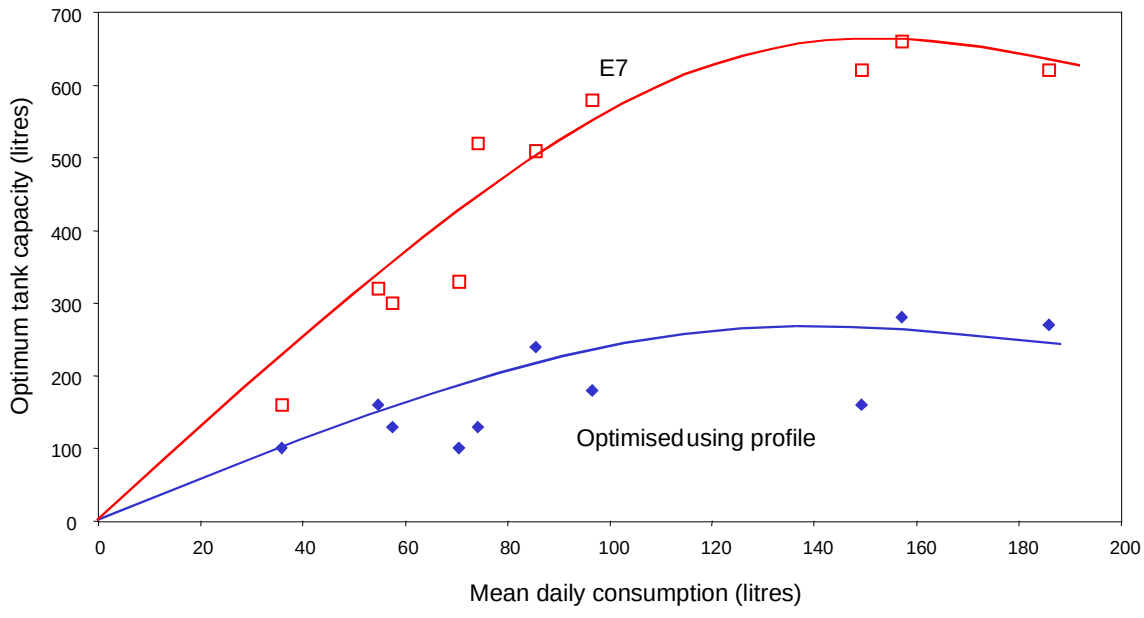


Fig 5-17 *Optimum tank sizes*

temperature.

5.8.5 How is the Optimisation Working

Fig 5-18 compares an optimised solution with the E7 situation, showing how the tank temperatures and cumulative daily costs vary for a case with a 200-litre tank.

It can be seen that for the E7 schedule the tank is charged to full capacity starting at midnight. The tank does not require a full 7 hours charge, typically only 2 or 3 hours are required before it reaches the set point temperature. Examination reveals that the difference in cost occurs in the way the tank is charged over this night time period. The optimised solution delays charging in order to miss the peak prices that occur after midnight. Ironically these prices are a result of the night time tariffs being introduced, as there is a surge in demand at midnight when appliances are switched on. This is seen consistently for all three days. Days 1 and 2 show that the relatively low demand enables a reduced temperature in the water tank. Rather than heat a full tank of water it is more economical to part heat it and then just top up the demand to the required temperature with direct heating. On day 3 the optimised schedule is roughly half the cost of the E7 schedule but has warmer water in the tank at the end of the day even though it started

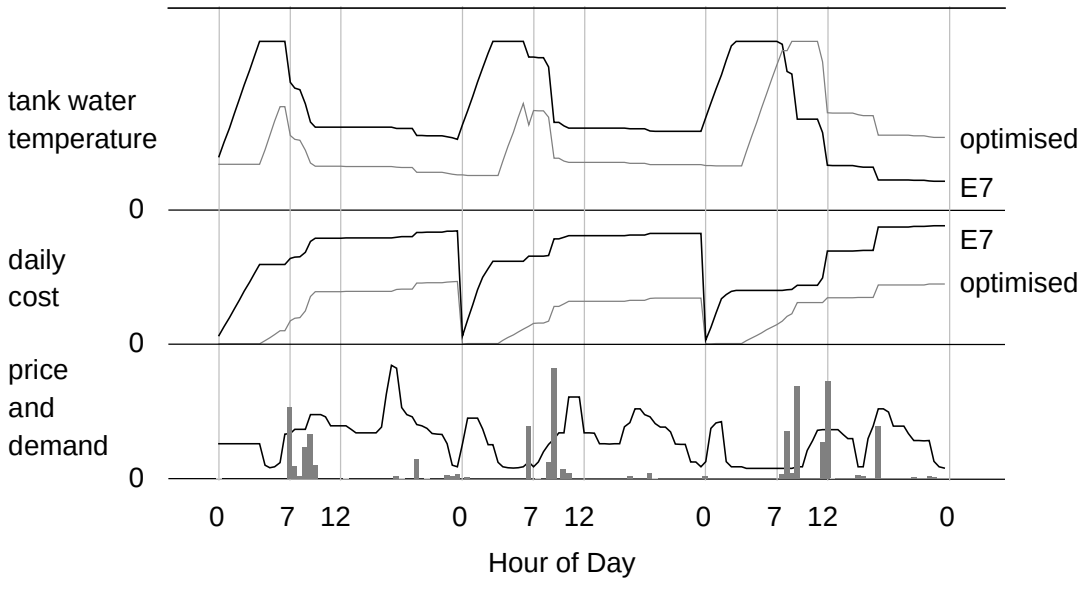


Fig 5-18 *Comparison of an E7 and optimised solution over 3 days*

colder.

5.8.6 Local Minima

An example of how a search can become trapped in a sub-optimal solution is demonstrated by Fig 5-19. Two charging schedules are shown (schedule 1 and 2) with their resultant cost and tank temperature profiles. The only difference in the two solutions is the hours at which the tank is charged, schedule 1 being charged in time slots 8,9,10 and 11 while schedule 2 is charged in slots 1,7,8 and 9. The tap source was identical for both solutions. When this particular day was optimised in isolation these two solutions were constantly found, but schedule 2 would never be reduced to schedule 1. In order to achieve schedule 1 the search would have to be restarted.

What can be seen is that for any improvement more than one bit flip in the charging schedule is required. Four charging periods appear to be required but one bit flip will result in 3 or 5 charging periods. In order to keep four periods but redistribute the times, two bit flips are required, one to destroy and one to create. To jump directly from schedule 2 to 1 requires 4 bit flips to simultaneously flip periods 1,7,10 and 11, which is why it never occurred as only 3 flips were allowed. With four bit flips allowed the chance of jumping from schedule 2 to 1 is less than 1 in over 80 million, assuming the string is not reduced in length from 96 bits.

No intermediate solution is possible because of the demand in time slot 4. This illustrates the benefit in frequently restarting the stochastic search as opposed to continuing the search from a local minimum and also having the possibility of more than one bit flip between solutions.

What is interesting in these optimised schedules is that although both have a full tank of hot water available for the demand in time slot 15, cold water from tap 2 is selected and the hot water saved for the higher level of demand in slot 16. Once the tank is almost emptied (due to the demand at time slot 16) it is not recharged because there are no further cheap periods of which to take advantage.

5.9 Chapter Summary

The simulations have demonstrated the potential for large improvement in water heating

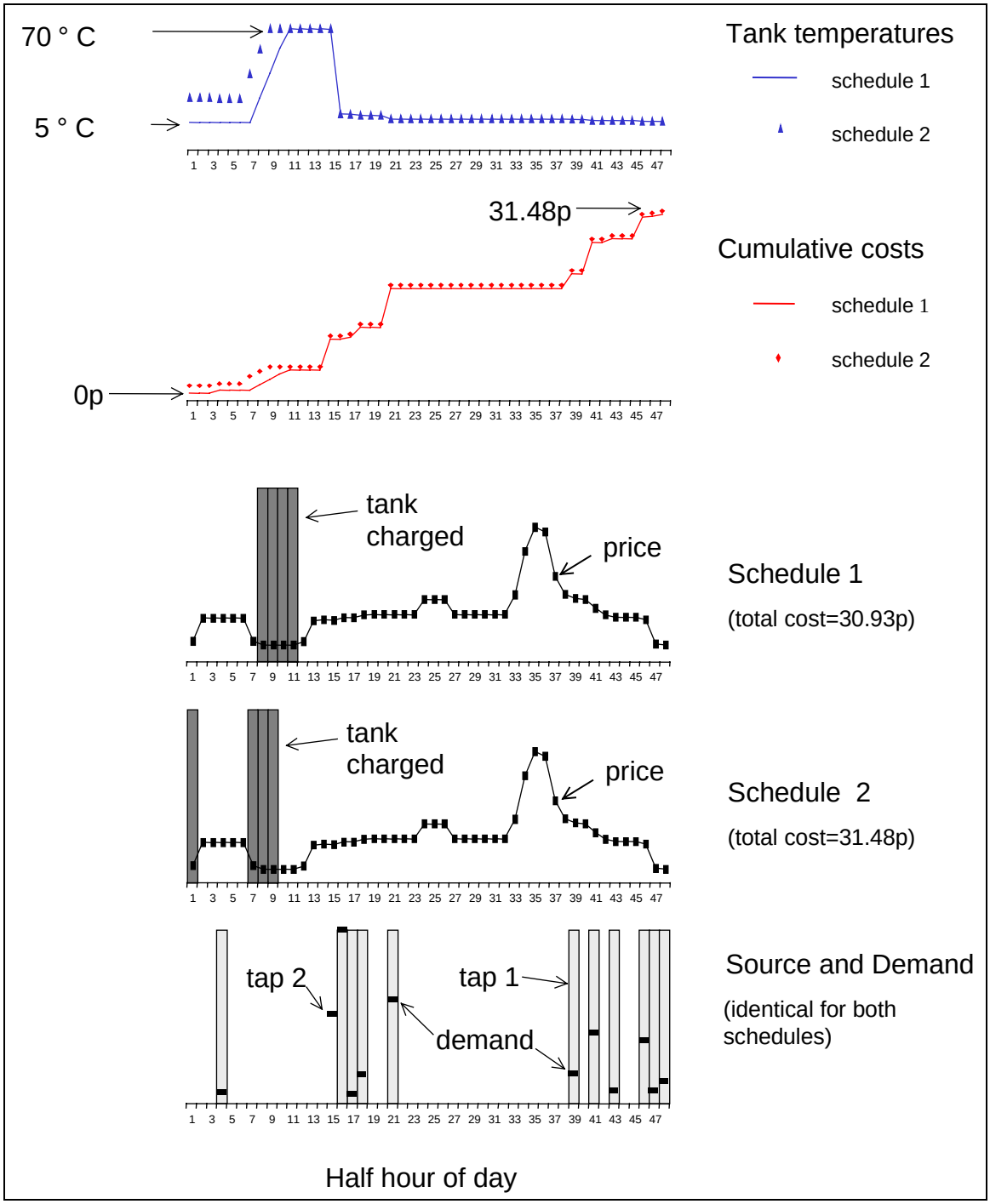


Fig 5-19 *Global and local minimum solutions. The x-axes are relative linear values for visual comparison only.*

strategies based on real consumption data and prices. Saving of the order of 40% were not untypical, which would reduce the cost of electricity supply to domestic customers by 16%, based on water heating being 40% of the total consumption [**81,90**]. There are no technological barriers to implementing such a control scheme.

6

Storage Radiator Controller Simulation

6.1 What are Storage Radiators ?

A storage radiator (also commonly known as a storage heater) is essentially a brick that is electrically heated during the night and dissipates the stored heat gradually throughout the day. The idea is that the thermal storage capacity of the bricks is utilised to shift electrical heating load in order to help increase the load factor.

Fig 6-1 shows a cross section through a typical radiator. An electrical heating element is encased within the bricks and is used to heat up this ‘core’. Surrounding the core is thermal insulation that helps retain the heat. The core contains channels through which air circulates by natural convection, heating up the room.

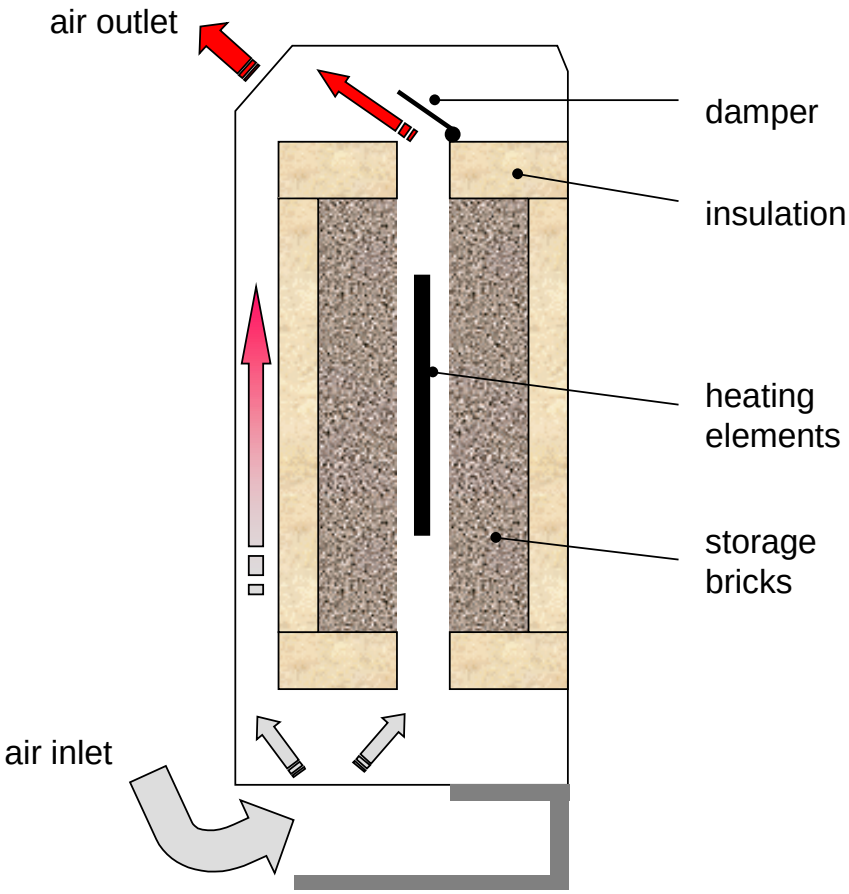


Fig 6-1 *A cross section through a basic storage radiator*

There are two manually operated controllers on the basic radiator. The first controls the energy input and adjusts the thermostat regulating the core maximum temperature. The second controls the heat output and adjusts the damper position, regulating the amount of air that is allowed to circulate through the core. The radiators (and water heaters) are hard-wired to a separate electrical circuit, which is activated from the electricity meter by a time clock or radio tele-switch.

In order to improve their controllability, fan storage radiators were developed. These have a high level of insulation to minimise heat loss, and an electric fan that can be activated to force air through the core when heat is required. The air gap in the core is designed so that this forced convection is required to extract the heat. This is done by having an inverted ‘u’ shaped air passage.

6.2 Room Thermal Model

In order to simulate the heating controller, a thermal model of a room is required which the neural network has to attempt to emulate. The neural model is then used to evaluate heating strategies, of which an optimum is sought. The thermal model is derived from first principles and attempts to emulate the response of a real system.

An explicit finite difference method was used to create the thermal model. This is a nodal approach that calculates temperatures at nodes within building elements (walls, core etc.) at discrete time intervals, which was every 5 seconds in this particular simulation. Heat transfer within the elements is by conduction, with convection taking place at the element surfaces. Each wall can be given different thermal properties and exterior air temperatures. Wind speed and outside air temperature were the only weather variables required, as solar gains were ignored. Every 5 seconds the net energy input into the room is calculated and the room air temperature updated. Appendix F gives more details of how the storage radiator was simulated and code for a room with a storage fan heater.

The simulated room had a 2kW storage radiator and a 1kW direct acting heater. The direct acting heater was only allowed to operate at times when there was a required internal temperature. The storage heater had a thermostat that stopped charging if the core temperature was above 700 °C. The room dimensions were $4 \times 5 \times 2.5$ metres high, with 50% of the wall area exterior, 10% of this glazed. There were 0.1 air changes per hour with the outside air. Data on the room conditions was recorded at the end of every half-hour.

The exact details of the room configuration are not important and do not have to reflect any ‘typical’ type of room that the heater will be placed in. What is being investigated is if the neural network can learn the behaviour of the given room , whatever its properties.

6.3 Neural Network Emulator

The purpose of the neural network is to generate an empirical model that will emulate the behaviour of the theoretical model. The objective is to be able to predict what the room temperature will be in half-an-hour, given the current conditions and the heat inputs in that half-hour. The heat inputs can then be optimised so that the required temperature is satisfied in the cheapest manner.

The neural model was created by reducing the heat transfer process into three distinct parts. Fig 6-2 is a schematic of the process and Fig 6-3 the neural emulator created.

Network 1 predicts the next inner core temperature (TCin+1) given the current core condition (TCin, TCout) and the storage charge occurring in that time slot (SHcharge+1).

Network 2 predicts the next outer core temperature given current core conditions, the room temperature and the previously predicted next inner core temperature.

Network 3 predicts the next room temperature given the current and predicted core states, the direct acting input (DAcharge+1) and historical weather temperatures for the previous 12 hours.

Networks 1 and 2 had two hidden neurons and network 3 had three. Hyperbolic tangent (tanh) and linear neurons were used in the hidden and output layers respectively, and all data scaled to lie in the range [–1,1]. Any network output outside this range was rounded to –1 or 1, effectively using an activation function known as softmax.

The time step of the neural model is half-an-hour. Starting at midnight, two charge values for the storage and the direct acting heaters are fed into the emulator and the thermal changes predicted. The time step is advanced and the predicted state is fed-back to become the current state. This process is repeated for all 48 time steps so the emulator can predict the room temperature response to the given 24 hour charging schedule, but based on a single model predicting half-an-hour ahead.

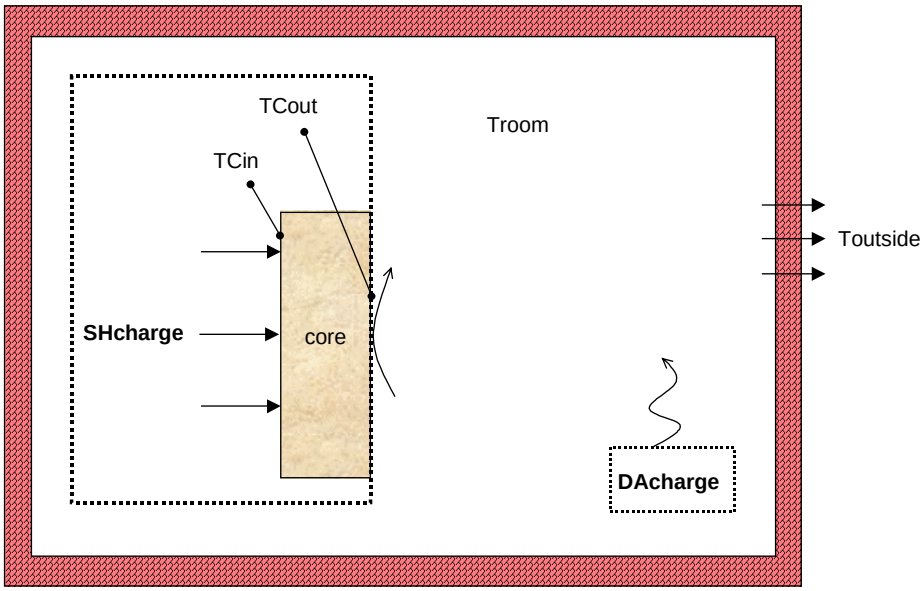


Fig 6-2 *Schematic of the heat transfer boundaries in the model*

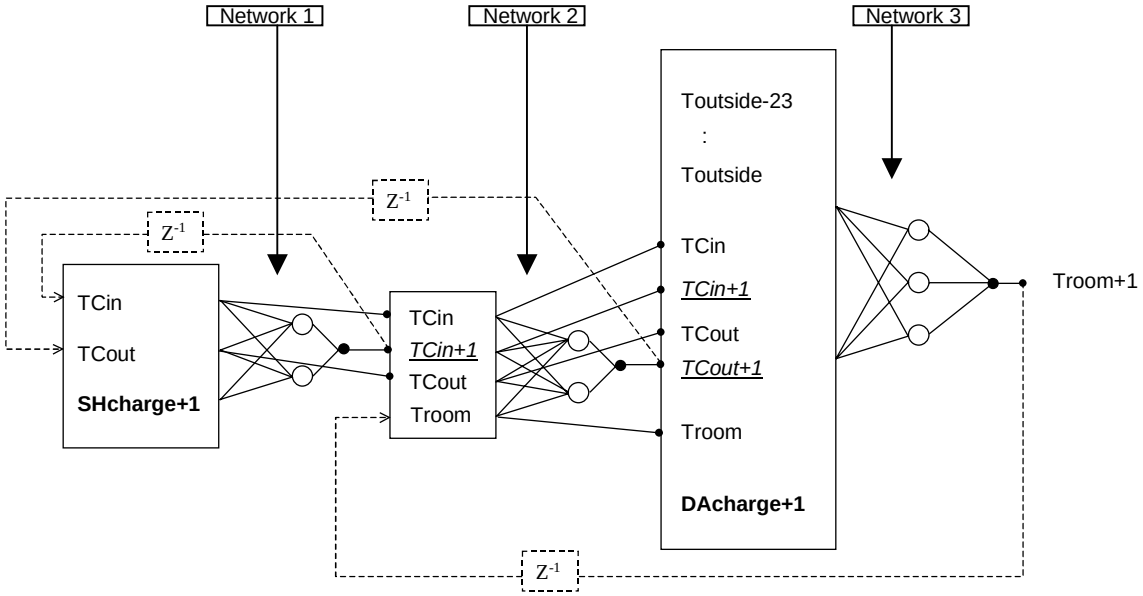


Fig 6-3 *The neural emulation of the theoretical model for predicting the room temperature in half-an-hour, given the storage and direct acting energy inputs*

TCout = core outer surface temperature
TCin = core inner surface temperature
Troom = room temperature
Toutside = outside temperature
SHcharge = energy to storage heater
DAcharge =energy to room by direct acting heater

In the neural emulator, outside weather temperatures are required for the 12 hours previous to the current time slot under consideration. In the simulations retrospective actual temperatures are used, although in reality these would have to also be predicted.

6.4 Optimisation Procedure

The neural emulator was used to evaluate proposed daily schedules of half-hourly charges for the storage and direct acting heaters. A schedule consists of a string of 48 or 96 numbers, depending on the optimisation used. The evaluation was equated in terms of cost, calculated by using the pool price.

Occupancy profiles were created giving the hours at which set point temperatures are required. Both the occupancy times and set point temperature values were varied so as to create a diverse range of conditions for the neural network to learn. If no set point was given (room unoccupied) then the temperature could behave in any manner, but a condition was given that the direct acting heater could not be on in these periods.

To compare the neural controller performance with existing heating strategies on a like-for-like basis, it was a requirement that the schedules must attempt to satisfy the room temperature set points given by the daily profile. This was the reason for including the direct acting heater, which operates only when there is occupancy in order to make up any shortfall in temperature.

In each half-hour there is a storage heater electrical charge and a direct acting electrical charge. In the theoretical model these are either 2 kW (storage) or 1 kW (direct). Because the thermostats can operate every 5 seconds, the recorded value at the end of each half-hour was equivalent to a continuous charging at a fixed percentage of full power. The load inputs to the neural emulator are thus real numbers, equivalent to the percentage of full power that should be utilised.

In order to implement this in the optimisation procedure, the string representation has to be changed. The control actions, previously represented by a bit at a particular location

on a string, are no longer binary choices. Each action can now be any real number between 0 and 100, so the string could be real valued with a mutation being the random generation of a new real number as opposed to a bit flip. From the experience gained in chapter 5, a random number of mutations (between 3 and 10) were allowed between each evaluation.

The primary optimisation task is to track the given temperature profile. The ability to give the heaters variable charge levels (although constant in each half-hour period) means that there is an infinite number of solutions to this problem. Because heat can be stored and the price varies every half-hour, each solution will have a different associated cost, the one giving the lowest being sought. This is a difficult optimisation problem and attempting to find the global solution would be a futile task. What is required is a method that can give a reasonable solution.

To reduce the search space the storage heater loads were limited to 5 discrete values equivalent to 0, 0.5, 1, 1.5 and 2 kW. This is more realistic to how a real controller would operate by activating a set of four 0.5kW elements. It also has the advantage over using continuous real numbers in that a random real mutation is unlikely to set the charge to zero, or ‘off’, whereas there is a 1 in 5 chance with the discrete coding.

As there is no control of the storage radiator output, the optimisation is simpler than that of the hot water system in chapter 5 as no decision has to be made as to the source of the heat. If a storage fan heater were simulated then the decision to switch the fan on would be equivalent to taking hot water from the tank.

Because the set point temperature has to be satisfied then only the storage heater charge needs to be optimised. If the storage schedule does not meet the required set point in any particular time slot, the direct acting heater is activated by incrementing the emulator load from 0 to 1kW until it is predicted that the set point will be achieved. A storage schedule will thus automatically have an associated direct acting cost, giving the total cost for that solution.

By optimising for minimum cost, a likely solution is never to charge the core. To prevent this scenario, a high penalty cost was added to the total cost if there was under heating at times when set points were required. Because of this it was necessary for the initial starting schedule to include a high degree of charging. If the initial solution was not to charge then there would be a high penalty cost due to the under heating. If random mutations did not instantly eliminate under heating in at least one time slot, then the same penalty cost would still be incurred as well as an added cost due to the charging, thus increasing the overall cost. The search procedure would thus never advance.

6.5 Simulation Procedure

Two simulations were performed in the optimisation. One was for a situation where the thermostat switches on the radiator core and direct acting heater were operational when the optimised storage schedule was passed through the building model (NNopt). The model then automatically activates the direct acting heater when required, giving a similar control scheme to existing strategies.

The second simulation (NN) was performed where both direct acting and storage charges were simultaneously optimised (using a real valued string of length 96) to maximise the comfort satisfaction. The cost function being minimised was thus the total absolute error, where the error is the difference between the required set point and the model prediction. The optimised solution was fine tuned by adjusting the direct acting charge so that the set points were satisfied, and the schedule then passed through the building model with no thermostat switches. This was performed to provide a yardstick from which the effectiveness of the cost optimisation could be assessed and to test the accuracy of the emulator predictions.

E7, E10 and direct acting only performances were also evaluated. In these cases the control rule was given that the direct acting heater would switch on if at any time during the occupancy periods the room temperature was below the set point, and switch off again once the set point was exceeded. During non-occupancy periods the direct acting

heater was switched off. For the simulation with only a direct acting heater, the element size was increased to 2 kW to prevent under heating.

The controller was simulated for 72 days starting from 1st January using weather data from Kew. The neural networks need some initial data on which to train, so 3 days on an E7 charging profile were simulated and used to create this initial database.

Fig 6-4 is a representation of how the simulation proceeded. From the database of actual past behaviour, training patterns were created and the three networks trained. The networks were then were used as an emulator to evaluate candidate charging schedules, which had been suggested by the optimisation process. Once a solution had been obtained, the charge levels were then applied to the theoretical model and this 'actual' behaviour recorded and added to the database. The process was then repeated for the next day.

In re-training the networks, the previous weights were used and given 25 epochs training on the patterns in the updated database. The optimisation was allowed to proceed for 800 function evaluations, which took considerably longer than the network training,

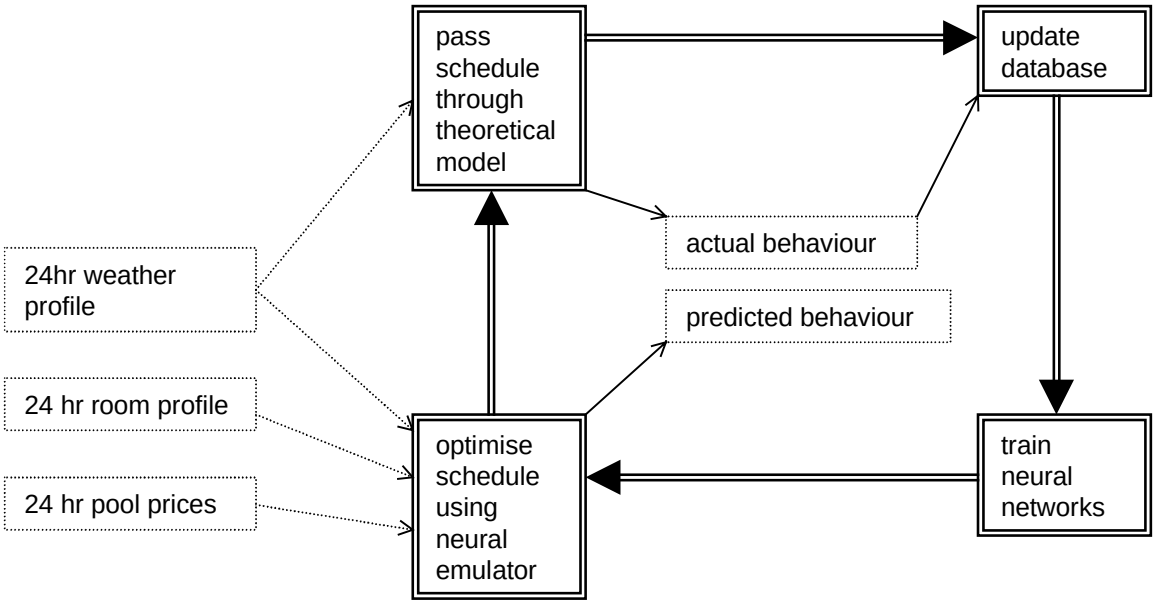


Fig 6-4 A *schematic of how the simulations were performed*

although still only in the order of 10's of seconds.

6.6 Results

6.6.1 Did the Controller Work ?

Without analysing the results too deeply, did the controller generally achieve a satisfactory performance in terms of temperature control – or basically, 'did it work? The results of simulation NN, where there is no thermostatic control, is shown in Fig 6-5. The differences between the predicted and achieved temperatures are shown chronologically over the 72 days.

It can be seen that eventually, after some initially large errors, the achieved temperatures consistently fall well within 1 °C of the predictions. For building thermal control this is within acceptable limits and it can thus be stated that in simulation the model predictive controller does work.

6.6.2 Why do the Large Initial Errors Occur ?

From Fig 6-5 it is observed that early in the simulation there are days when the errors are relatively large, indicated by A, B and C. A close examination of the data, shown in

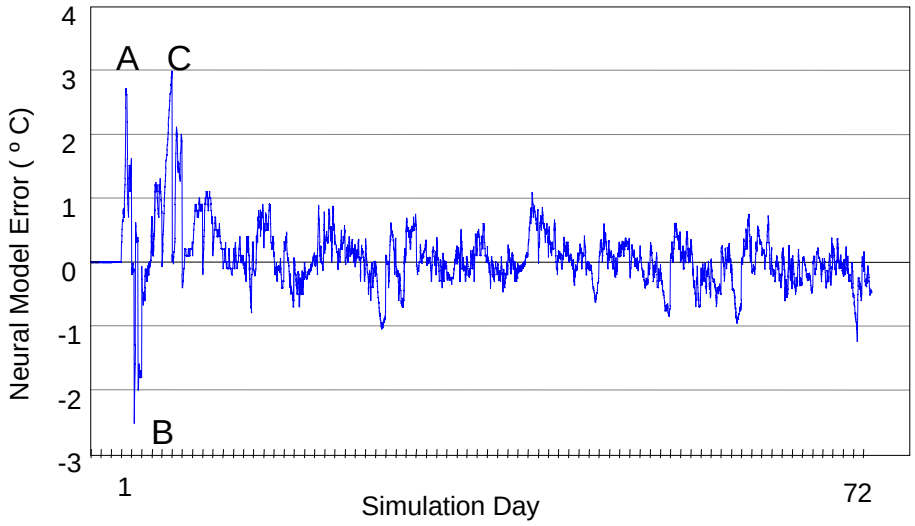


Fig 6-5 *Half-hourly errors over the 72 day simulation. The error is the difference between the emulated and achieved temperatures.*

Fig 6-6, reveals why these errors occur.

Large errors occur on day A, the first day that the controller is used. The three previous day's data were used for training, which originated from an E7 charging schedule. During these three days, the core temperature never falls below a certain value and the data would be scaled between these current limits of experience. The emulator core temperature will never fall below its previous minimum whatever the charge, due to the fact that all predictions fed back have to lie within the scaled range [-1,1]. There can thus be a solution with no storage charge but a heat input equivalent to the core being at the previous minimum. This is what has happened on day A, with the actual response of the core temperature to no charging falling below what the network predicted. As a result the room is under heated.

The same effect is repeated for day C. There is no occupancy during this day so there will be no storage charge. The core is already at the minimum temperature previously experienced at the start of this day, which the emulator guarantees will not to be surpassed. Also at the start of this day the room temperature is close to its lowest ever value and the emulator again has restricted the predicted temperatures to lie within current experience. This is why the predicted temperature for day C is almost constant at its previous minimum. Once the new data has been added to the database the limits change, as can be seen for the following day.

For day B the direct acting charge is at a higher level than had previously occurred. Any charge level above that previously experienced will not have the corresponding increase in effect. This is due to the saturation of the tanh activation function that limits extrapolation. When the optimised charge is then applied, overheating occurs. This shows an inefficiency of the optimisation process, as the direct acting input could have been reduced giving the same comfort for less cost. The fine tuning did not correct this as it only adjusted the direct acting heat input upwards.

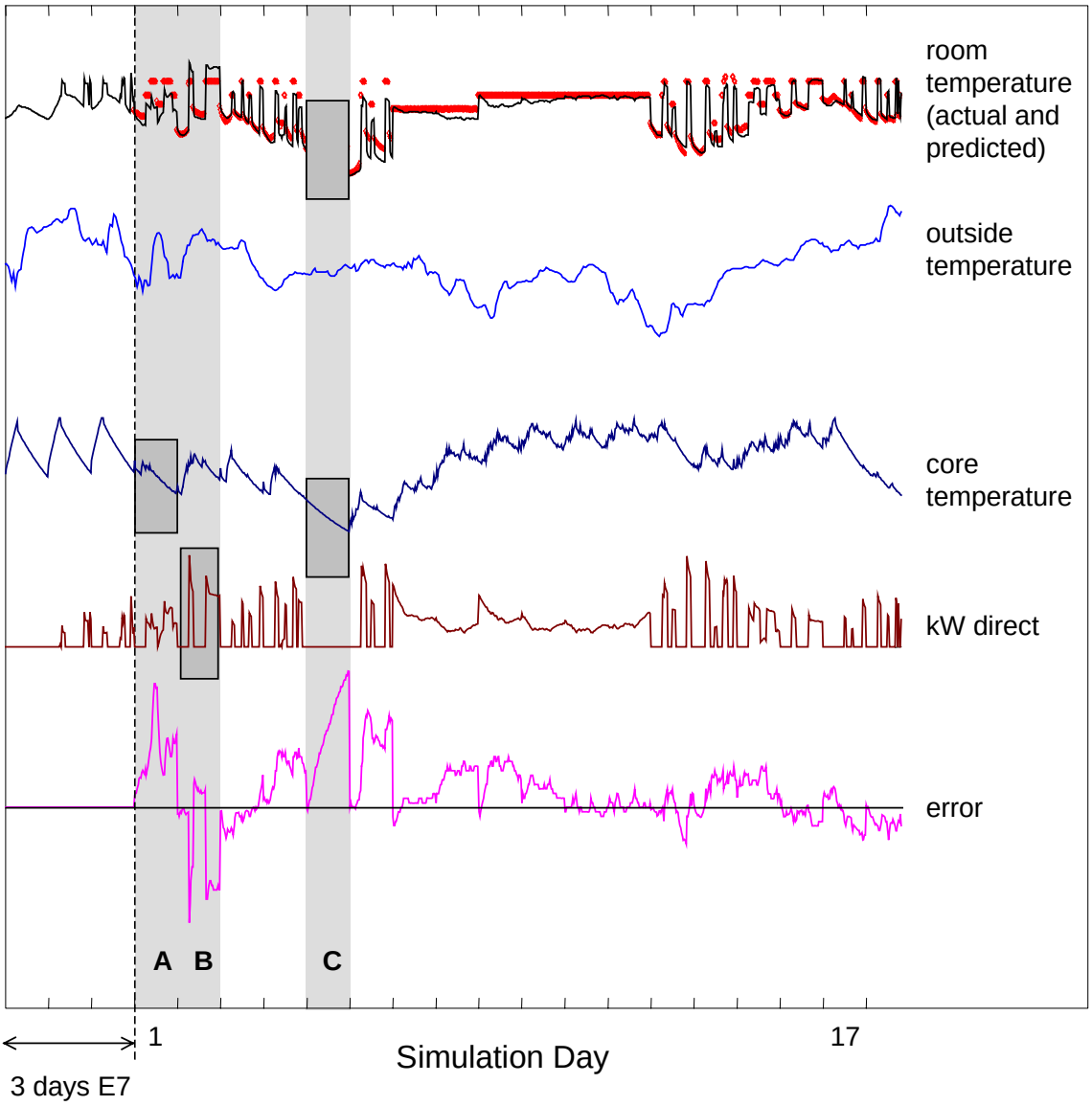


Fig 6-6 *Investigating why the early errors occur*

The initially large errors occur due to the constraints placed on the emulator to prevent it from extrapolating (by using the softmax output function). If these constraints had not been set (by having a linear output function) then similar results would have been expected if the model had used the full range of the tanh activation function. This would ensure that any fed-back input outside current experience would be in the saturated regions and thus automatically set at either –1 or 1.

The neural network learns the room characteristics within its current limits of experience almost immediately. Errors occur when these limits are exceeded but once the full range of data has been experienced the performance is satisfactory.

6.6.3 Performance of the 1-step-ahead Predictor as a Recursive 48-step-ahead Predictor

The neural emulator was created in such a way so that there were predictions used within the current time step to estimate the room temperature – or a prediction based on predictions. These predictions were then fed-back to be used as inputs for a further 47 time steps through the day. This deliberately created 'worst case scenario' has the potential for multiple error accumulation and therefore should rigorously test the accuracy of the emulator models.

Fig 6-7 shows how the prediction errors accumulate throughout the day. The initial one step ahead error for hour 0.5, where the exact previous conditions are known, is around 0.07 °C. This immediately doubles for the next time step but gradually levels off at 0.4 °C around time step 20 (hour 10). The sharp jump at hour 7 is due to the occupancy

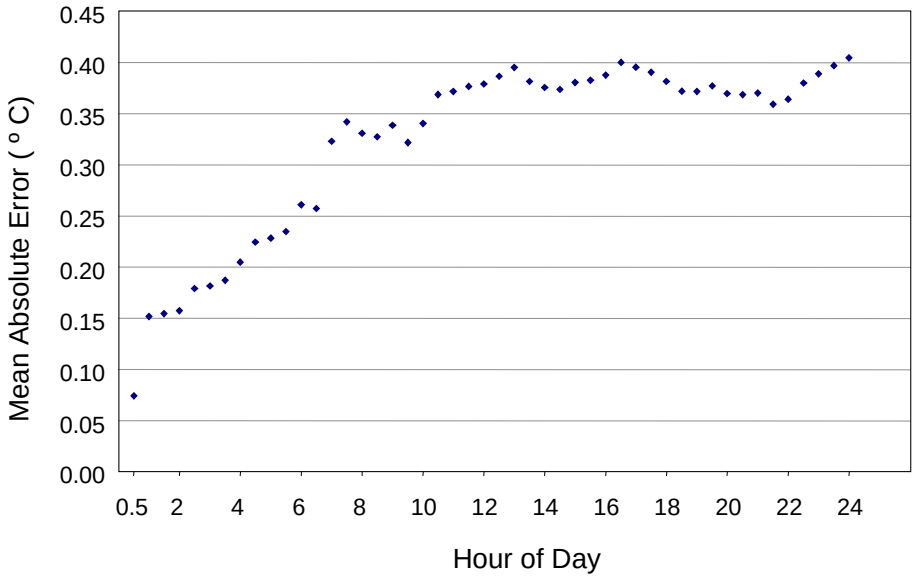


Fig 6-7 *The absolute error of the emulator for each time step averaged over the 72 days*

patterns consistently requiring a set point temperature for the start of the day. This will result in direct acting heaters being switched on, giving a large change in room temperature and the potential for larger errors.

Averaged over time it can be seen that there is a gradual deterioration in performance up to hour 10 and then a relatively constant error for the remainder of the day. It could easily be assumed from these results that the errors of each individual day will follow a similar pattern, with errors gradually accumulating as the fed back predictions become gradually worse.

A closer inspection of the actual daily errors shown in Fig 6-6 and in more detail for a different period in Fig 6-8 reveals this is not the case. For day 50 (Fig 6-8) when there is no occupancy and thus no heating, there is a general drift in the error up to a point, but even here it starts to improve in the later stages of the day.

6.6.4 Comparison with other Heating Strategies

To assess the relative performance of the model predictive controller the simulations were repeated for E7, E10 and a direct acting (DA) only charging schedule. The results are shown in Table 6-1.

The comfort optimised (NN) neuro-controller is almost twice as expensive as all the other heating strategies, although its actual energy consumption is comparable to E7 and E10. This is unsurprising, as its only objective is to satisfy the demand without any cost considerations. This demonstrates how electric heating can be very expensive if not efficiently regulated.

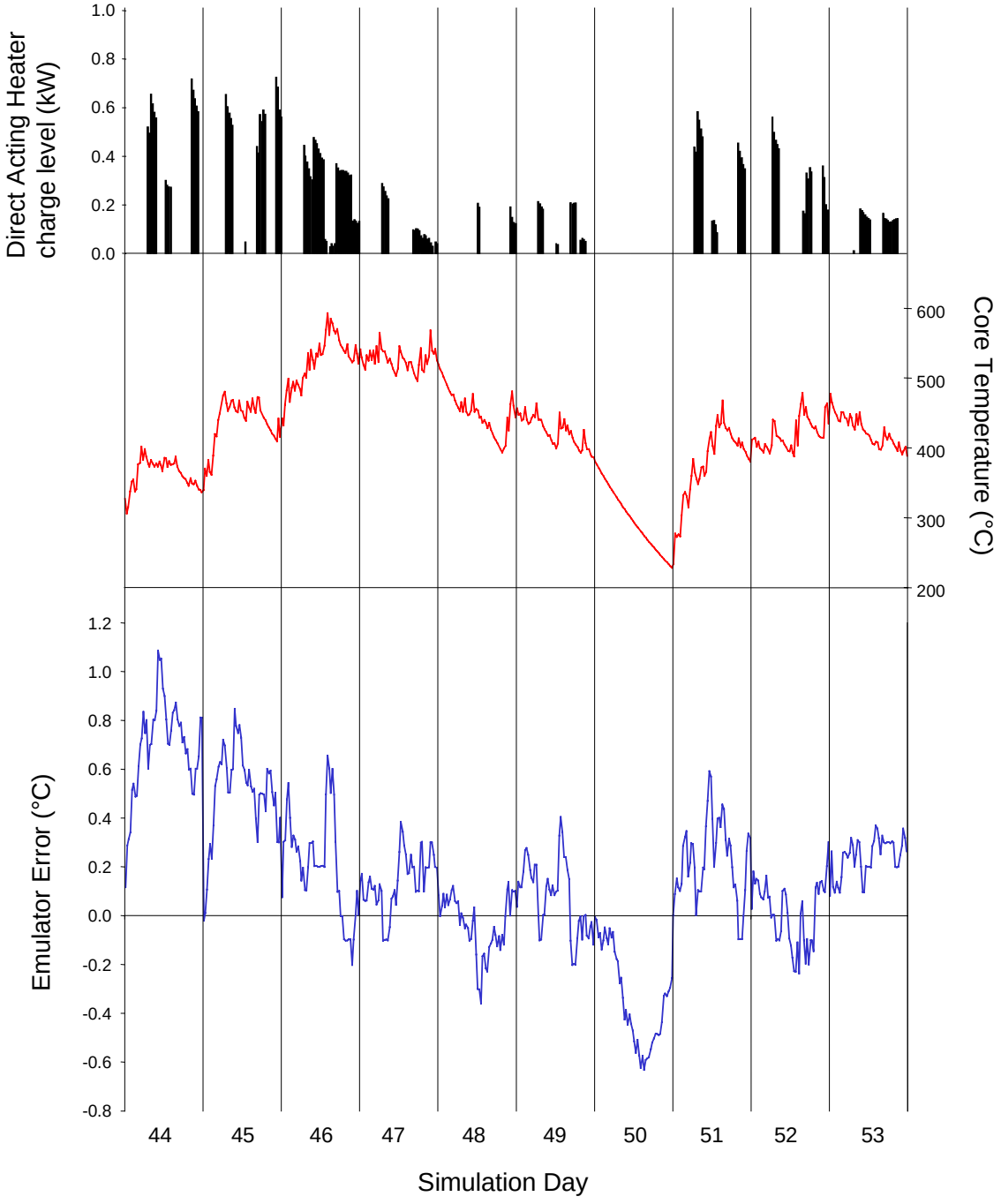


Fig 6-8 *The actual daily errors do not follow the time averaged pattern of Fig 6-7*

The cost optimised solution (NNopt) is 5.5% cheaper than E7 and uses 27% less electricity. This imbalance in the cost savings is in part due to the fact that there were some high pool prices. The peak pool price was 70p/kWh whereas the mean price of the cheapest 90% of half-hours was only 1.5p/kWh. For NNopt, 34% of the direct acting cost was accumulated in only 4% of the time that the direct acting heater was operational.

The improvement brought by the controller is in its ability to accurately regulate the temperature. In E7 there is excessive overheating of effectively 2.29 °C for a continuous period of 10 days, with E10 being worse. The neuro-controller has weather information for the day ahead so it can set the loads at a level so that overheating will not occur. Overheating does occur for 21 slots out of a possible 1,383, due to set points of 16 °C being given, the cooling rate being too low for the temperature to fall enough in the time specified.

Table 6-1 *The relative performances of 5 heating strategies*

	NN	NNopt	E7	E10	DA only
COST (£)					
Storage	30.99	6.70	12.17	17.16	----
Direct	4.55	9.37	4.83	3.01	21.86
TOTAL	35.54	16.07	17.00	20.17	21.86
CONSUMPTION (kWh)					
Storage	986	429	959	1097	---
Direct	104	341	104	62	622
TOTAL	1090	770	1063	1159	622
COMFORT					
Overheat (>0.5 ℃)					
Occurrences	404	21	476	696	---
Average (°C)	1.44	2.44	2.29	2.62	---
Underheat (>0.5 ℃)					
Occurrences	47	---	---	---	---
Average (°C)	-0.73	---	---	---	---

The most energy efficient solution is direct acting only, as energy is not wasted heating periods that do not require a set point. The neuro-optimised solution (NNopt) is 23% less efficient than direct acting but 26% cheaper.

6.7 Emulator Improvements

The neural emulator created was a 1-step-ahead predictor that used its own predictions to extrapolate to 48 time steps ahead. At each time step the only information it receives that is not predicted directly from the starting conditions is the outside temperature, for which actual values are used. By feeding back the predictions to advance a time step there is thus no new information being introduced apart from potential errors.

A better approach would be to have 48 networks, each trained to predict the temperature for a given time step ahead. The inputs would be the initial starting conditions and the weather and loads that had occurred up to that time step. Intermediate temperatures are thus not introduced as they do not need to be known for the current prediction.

By taking the average temperature from a population of models for each time step (see section 3.10 on page 61), the accuracy would be further improved.

The potential effect on the error of ‘wrong’ temperature predictions needs to be quantified. It is hypothesised that this will only be important when there are sudden unpredicted cold fronts and the model underestimates the heating requirement.

6.8 Chapter Summary

The simulations performed in this chapter have demonstrated that theoretically neural networks could work as model predictive controllers of domestic storage heating. The main benefit over existing systems is anticipating when overheating will occur and reducing the charge appropriately.

[66**]** J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[67**]** L. Altenberg, Evolving better representations through selective genome growth,' *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, vol.1, 1994, pp. 182-187

[68**]** M. Mitchell, *An introduction to Genetic Algorithms*, The MIT press, 1996

[69**]** D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[70**]** M. Mitchell, S. Forrest, and J.H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," *Proceedings of the First European Conference on Artificial Life*, 1992, pp. 245-254.

[71**]** S. Forrest and M. Mitchell, "Relative building-block fitness and the building-block hypothesis," In *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, 1993, pp.109-126.

[72**]** M. Mitchell, J.H. Holland, and S. Forrest, "When will a genetic algorithm outperform hill climbing?," In *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, 1994.

[73**]** T. Walsh, "Empirical methods in AI," *AI Magazine*, summer 1998, pp.121-124.

[74**]** K.A. De Jong, "Genetic algorithms are NOT function optimizers," In *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, 1993, pp.5-17.

[75**]** A.E. Nix and M.D. Vose, "Modeling genetic algorithms with Markov chains," *Annals of Mathematics and Artificial Intelligence*, vol. 5, 1992, pp.79-88.

[76**]** S. Kirkpatrick, D.C. Gellat and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, 1983, pp. 671-680.

[**77**] A.H.G. Rinnooy Kan and G.T. Timmer, "Global optimisation: a survey," International Series of Numerical Mathematics, vol. 87, 1989, pp. 133-155. Bikhauser Verlag Basel.

[78**]** F. Glover, "Tabu search I," *ORSA J. Comp*., vol. 1, 1989, pp. 190-296.

[**79**] E.L. Lawler and D.E. Wood, "Branch and bound methods : a survey," *Journal of Oper. Res*., vol. 14, 1966, pp. 699-719.

[**80**] D.H. Wolpert and W.G. Macready, “No free lunch theorems for optimization,” *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, April 1997, pp. 67-82.

[81**]** B. Rautenbach and I.E. Lane, “The multi-objective controller: a novel approach to domestic hot water load control,” *IEEE Trans. Power Systems*, vol. 11, no. 4, November 1996, pp. 1832-1837.

[82**]** S.J. Dickinson, “A building heating system simulation and optimisation tool incorporating bond graphs and genetic algorithms,” PhD Thesis, Lancaster University, 1996.

[83**]** Symposium papers on “Predicting hourly building energy usage: the great energy predictor shootout – the aftermath,” *ASHRAE Transactions*, 1994, vol. 100, pt. 2, pp. 1053-1118. (also see refs 37 and 43)

[84**]** A. Dhar, “Development of Fourier series and artificial neural network approaches to model hourly energy use in commercial buildings,” PhD Thesis, Texas A&M University, May 1995.

[85**]** P.Y. Glorennec, “Modélisation d’un bâtiment par un réseau neuronal récurrent,” Présenté aux Septièmes Journées Internationales Les Réseaux Neuromimétiques et leurs Applications, Paris, October 1994. (in French)

[86**]** M.C. Mozer, R.H. Dodier, M. Anderson, L. Vidmar, R.F. Cruickshank III, and D. Miller, “The Neural Network House: An overview,” In L. Niklasson & M. Boden (Eds.) *Current trends in connectionism* , 1995, pp. 371-380. Also online http://www.cs.colorado.edu/~mozer/nnh/index.html

[87**]** M.C. Mozer, L. Vidmar, and R.H. Dodier, “The neurothermostat: predictive optimal control of residential heating systems,” In M.C. Mozer, M.I Jordan & T. Petsche (Eds.), *Advances in neural information processing systems 9*. 1997.

[88**]** K.O. Temeng, P.D. Schnelle, and T.J. McAvoy, “Model predictive control of an industrial packed bed reactor using neural networks,” *J. Proc. Cont.*, vol. 5, no. 1,1995, pp.19-27.

[89**]** G. Gvazdaitis, S. Beil, U. Kreibaum, R. Simutis, I, Havlik, M. Dors, F. Schneider, and A. Lubbert, “Temperature control in fermenters: application of neural nets and feedback control in breweries,” *J. Inst. Brew*., Vol. 100, 1994, pp. 99-104.

[90**]** L. Clark, “Smart meter gives you the power,” Sunday Star-Times, November 30 1997, pp. D5. (New Zealand newspaper) relating to Exicom Technologies (1996).

[91] B. Daryanian, R.E. Bohn, and R.D. Tabors, “An experiment in real time pricing for control of electric thermal storage systems,” *IEEE Trans. Power Systems*, vol. 6, no. 4, November 1991, pp.1356-1365.

[92] B. Daryanian, R.E. Bohn, “Sizing of electric thermal storage under real time pricing,” *IEEE Trans. Power Systems*, vol. 8, no. 1, February 1993, pp. 35-43.

[93] K.L. Lo, J.R. McDonald, and T.Q. Le, “Time-of-day electricity pricing incorporating elasticity for load management purposes,” *Electrical Power and Energy Systems*, vol. 13, no. 1, 1991, pp.230-239.

[94] M. Räsänen, J.Ruusunen, and R.P. Hämäläinen, “Customer level analysis of dynamic pricing experiments using consumption-pattern models,” *Energy*, vol. 20, no. 9, 1995, pp.897-906.

[95] M. Räsänen, J.Ruusunen, and R.P. Hämäläinen, “Optimal tariff design under consumer self-selection,” *Energy Economics*, vol. 19, 1997, pp. 151-167.

[96] J. Zarnikau, “Customer responsiveness to real-time pricing of electricity,” *Energy*, vol. 11, 1990, pp. 99-116.

[97] A.K. David and Y.Z. Li, “Effect of inter-temporal factors on the real time pricing of electricity,” *IEEE Trans. Power Systems*, vol. 8, no. 1, February 1993, pp. 44-52.