# A Prototype Emergency Resource Redistribution System For Disease Control Programmes

Andrew Tuson, Richard Wheeler and Peter Ross
OUTLOOK Group, Department of Artificial Intelligence,
University of Edinburgh 80 South Bridge,
Edinburgh EH1 1HN, U.K.
{andrewt,richardw,peter}@dai.ed.ac.uk

**Abstract**

The problem of logistics and resource management in disease control projects in the developing world can hardly be understated. One problem that arises is the occurance of regional imbalances in supply. Such imbalances can seriously compromise the effectiveness of such programs. A prototype system is described, based on an order-based plan generator and a class of modern combinatorial optimisation techniques, which recommends an *implementable* plan for the redistribution of available resources to minimise such shortages for real-world situations.

## 1 Introduction

Dealing with aid projects, such as treating epidemics and disease control, in the developing world can be made difficult by the fact that though an effective regime or plan exists, the actual implementation of the plan is often more difficult. This problem is compounded further by poor communication and transportation links. For instance, situations may arise where regional imbalances occur. Such imbalances can seriously compromise the effectiveness of a relief program; an example would be a treatment site with plenty of diagnostic kits for a particular disease, but with no drugs to treat the diagnosed cases.

This paper outlines a prototype resource management system to evaluate the feasibility of a system that recommends an *implementable* shipment plan to minimise such shortages in real-world situations.

## 2 The Problem

The problem can be described as follows: we have $N$ sites, each of which need to have minimum amounts of $M$ resources in order to operate effectively. Furthermore, each site has different requirements for each resource, and shortages and surpluses of each resource occur at different sites. A resource management system has to give a list of recommendations of the form 'move X amount of resource Z from site A (which has a surplus) to site B (which has a shortage)'. A simple example of such a shipment plan is given in Figure 1. Also, the redistribution plan needs to be optimised according to some measure of effectiveness.

This problem is not trivial, as such a redistribution plan has to be able to be implemented in practice. This introduces constraints that have to be satisfied. As a result, the proportion of feasible plans in the space of all possible is likely to be low. The hardware limitations are also severe — the computer technology available is hardly state-of-the-art. Therefore the final system aims to run on a 386-class PC compatible, with a 1 MB memory, and a 20 MB hard drive. The goal is to plan regions with at least 250 relief centres within several hours.

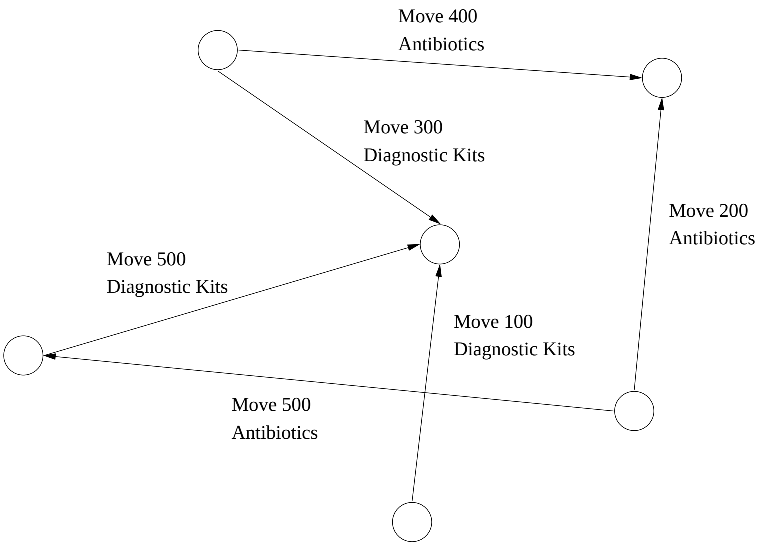


Figure 1: An Example Redistribution Plan For A Simple Domain

As this is a feasibility study, a simplified version of the actual problem was considered. The aim: to maximise the number of resource targets met (referred to as resource-sites), whilst minimising the number of shipments (if more than one resource is moved between two sites then this is counted as one shipment); subject to various constraints on feasibility both hard and soft, which are outlined below.

1. No site can supply more of a resource than it has.
2. If a site has been supplied with a resource, then it cannot supply another site with that resource. This prevents 'daisy-chaining' which can lead to brittle plans, because if one of the shipments in the plan was not to occur, the later shipments then are likely to be disrupted.
3. Do not ship less than a certain minimum amount of a given resource.
4. If possible, a site should get resources from a site that is nearby; as the shipment is more likely to be completed, and with lower cost.
5. If possible, a site should get resources from a site that has a large surplus; as the site will be more likely to send the shipment.

# 3 Implementation

The above problem can be viewed as a constrained optimisation problem. In other words, find a redistribution plan that minimises shortages and the number (and cost) of shipments required, whilst satisfying the constraints outlined earlier. The solution adopted consists of two parts: the representation, which describes the redistribution plan and contains the domain knowledge; and a search algorithm to optimise the plan.

## 3.1 The Representation

The representation of the redistribution plan adopted was indirect [Burke *et al.* 95, Fang 94] — the candidate solution encodes not the redistribution plan; but rather instructions to a plan builder. This approach is compact in its memory usage and always generates feasible plans. This 'anytime' characteristic is of particular use as an answer may be required quickly, and a sub-optimal but feasible answer is preferable to no answer at all. The encoding is the order in which sites are considered for supply (i.e. a permutation) by the plan builder. The pseudo-code for the order-based plan builder is given below. The algorithm is given the order in which the relief sites are to be considered, the current situation as regards resource levels at all the sites, and the desired level of supply, and information on the location of each site.

### 3.1.1 The Top-Level Algorithm

```
FOR each site in turn
BEGIN
  Assume that there is no preferred site for supply
  FOR each resource in turn
  BEGIN
    IF the site under consideration requires more of that resource
    (using REQUIRES-SUPPLY) THEN
    BEGIN
      UNTIL either the site has sufficient resource
        OR no site can supply it
      BEGIN
        Use FIND-SUPPLY to find a site that can supply that resource
        Use SUPPLY-SITE to move the required amount of resource
      END
    END
  END
END
```

### 3.1.2 The Procedure REQUIRES-SUPPLY

This is used to see if a site needs supplying with a given resource.

```
IF the site has less of a given resource than the target set for it
  THEN it is requires more of that resource
  ELSE it has sufficient resource
RETURN the result
```

### 3.1.3 The Procedure CAN-SUPPLY

This finds out whether a site can be used as a supply of a given resource.

```
IF the site has not already been supplied with the given resource
  (this prevents daisy-chaining)
  THEN the amount it can supply is the amount it has above its target
RETURN this amount
```

### 3.1.4 The Procedure FIND-SUPPLY

This is used to find a site that can supply another site with a given resource.

```
IF there is a preferred site for supply and it can supply more than the
  minimum transportable amount of that resource (using CAN-SUPPLY)
  THEN use that site for supply
  ELSE BEGIN
    LET DISTANCE = 1
    UNTIL either a suitable site has been found OR
      all sites have been considered
    BEGIN
      CAN-SUPPLY finds how much each site within DISTANCE can supply
      IF the site which can supply the most of that resource can supply
        more than the minimum transportable amount of that resource
      THEN this a suitable site has been found
      DISTANCE = DISTANCE + 1
    END
  END
RETURN either the suitable site found OR the fact that no such site exists
```

### 3.1.5 The Procedure SUPPLY-SITE

When a site that can supply a given resource is found, this moves the supply and notes the shipment made.

```
Calculate the amount of resource available from the supplying site
Calculate the amount of resource required from the receiving site

IF the amount available is less that the amount required
  THEN the amount moved is the
ELSE IF the amount required is less than the minimum transportable
  amount of that resource
  THEN the amount moved is minimum transportable amount
ELSE the amount moved is the amount required

Add the amount of resource moved to the receiving site
Subtract the amount of resource moved from the suppling site
Make a record of the move made
```

## 3.2 Optimisation

A variety of hillclimbers and 'meta-heuristic' optimisers were implemented for the optimisation of the initial redistribution plan. The neighbourhood operators used by all of the optimisation methods for the encoding described are: permutation-shift and permutation-swap [Reeves 95]. For the Evolutionary Algorithm (see below) the crossover operator used was the 'Modified PMX' operator [Mott 90], which has been found to give good results on some permutation-encoded scheduling problems.

The evaluation function used in the prototype was a simple one — a linear combination of the number of targets met (*targets_met*) and the number of shipments made (*shipments*). This is shown in the following equation.

$$fitness = 10 \times targets\_met - shipments \tag{1}$$

The number of targets met is weighted, so that the system will optimise *targets_met* , before trying to optimise *shipments*. This was deemed a desirable behaviour by the potential end-users. Each of hillclimbers that the optimisation techniques to be used are based upon are described briefly below:

- *Stochastic* hillclimbers (SH) take the current candidate solution and perturb it randomly. This 'move' is accepted (i.e. becomes the new candidate solution) if it is better or the same quality than the preceding solution.
- *Steepest-ascent* (SAHC) hillclimbers, systematically evaluate *all* of the possible moves from the current candidate solution and pick the move with the highest improvement.
- *First-ascent* (FAHC) hillclimbers, operate in a similar fashion, but will take the first improvement that they find.

The remaining methods are know as 'meta-heuristic' optimisers. All of them extend hillclimbing in some fashion to avoid a potential problem with hillclimbers — that they can get stuck in local optima (a good overview of this area is given by [Reeves 93]).

- **Iterated Hillclimbers (IHC)**. As above, but when a certain (user-defined) number of evaluations have been made without an improvement on solution quality, the hillclimber is restarted with a different initial solution.
- **Evolutionary Algorithms (EAs)**. Based upon the theory of evolution by natural selection, these techniques maintain a population of candidate solutions, and allow them to 'evolve' [Holland 75]. Two types of neighbourhood operator are commonly used: mutation,

which is equivalent to the moves made in hillclimbing; and crossover (roughly analogous to sexual reproduction) which selects two candidate solutions and (probabilistically) swaps information between the two. The population-based nature of the search, and the crossover operator often help to avoid the search being trapped in local minima.

- **Simulated Annealing (SA)**. Based upon the physical process of annealing; this is an extension of stochastic hillclimbing [Kirkpatrick *et al.* 83]. Worse quality moves can be accepted with a probability dependent upon a 'temperature' that is reduced as the optimisation progresses.

- **Threshold Accepting (TA)**. An extension of stochastic hillclimbing [Dueck & Scheuer 90]. This variant accepts moves that are better, the same, or worse *within a certain threshold* quality.

- **Tabu Search (TS)** An extension of first-ascent and steepest-ascent hillclimbing which uses memory to avoid local optima [Glover 90]. In a basic implementation which uses 'recency', when a move is accepted, its inverse is added to a 'tabu-list' which stops that move being made for a certain period of time (and therefore stops the search from revisiting points already evaluated). If the search finds a local optima then the least worse move is made — along with the tabu list, this allows the search to escape local optima.

# 4 An Initial Evaluation

The data for the test problems was provided by an expert with experience of the disease control programmes the system is intended to play a part in, and was based upon actual data. A small domain consisting of 21 sites and 12 resources was examined first as to exhaustively investigate a very large dataset over a range of optimisation techniques and their parameters would be computationally prohibitive. This test dataset had the feature that there were roughly sufficient resources available to supply all of the resource targets.

Stochastic hillclimbing was used to give a rough idea of the expected performance of the system. The result obtained was encouraging. The initial (randomly generated) solution met about 85-92% of resource targets when processed by the plan builder, compared to 32% before the system is run. Further optimisation increased this figure to 90-92%, and managed to reduce the number of shipments required by 20% — from about 125 to about 105 shipments — in less than 1000 evaluations. This took about 7 seconds on a SPARC 5 workstation, or 5 minutes on a 386-class PC.

# 5 A Comparison Of The Optimisation Techniques

The performance of each of the optimisers, for a range of technique-specific parameter values was investigated, including the choice of a swap or shift neighbourhood. Fifty runs, each lasting 2000 evaluations were taken in each case (where differences in performance are reported to be significant, this has been ascertained using a t-test). The following experiments were performed.

- **Hillclimbing** (Stochastic, Steepest-ascent and First-ascent), with and without iteration. The iteration interval was varied in the range 200-700 evaluations.

- **Tabu Search** (Steepest-ascent and First-ascent). The tabu tenure was varied in the range 1-10 for a basic (recency-based) TS implementation.

- **Simulated Annealing** (with a logarithmic cooling plan — $T(k) = T(0)/(1 + ln(k))$). The final temperature and interval between temperature adjustments were kept fixed at 0.05 and 100 evaluations respectively. The initial temperature was varied in the range 1-10.

- **Threshold Accepting**. The final threshold value and interval between threshold adjustments were fixed at 0 and 100 evaluations respectively. The initial threshold was varied in the range 1-10.
- **Evolutionary Algorithm**. The population models investigated were: Steady-State with Replace-Worst, Generational, Generational with Elitism, and an N+N Evolution Strategy (N = population size). Crossover probabilities were examined in the range 0.0-1.0. The EA population was fixed at 100, and the selection method used was rank-based with bias 1.5.

Due to lack of space, not all of the results obtained can be given here (the reader is directed to [Tuson 96] for full results). Instead, the best results obtained (in terms of solution quality, and then the number of evaluations to solution) for each technique are summarised in Table 1.

| | Shift Neighbourhood | | Swap Neighbourhood | |
|---|---|---|---|---|
| Algorithm | Quality | Evals. Reqd. | Quality | Evals. Reqd. |
| SHC | 2321.96 (1.15) | 1091.36 (519.64) | 2321.68 (1.19) | 1042.56 (474.51) |
| SA | 2321.64 (1.53) | 1045.92 (527.88) | 2321.26 (1.59) | 1074.68 (534.29) |
| TA | 2321.82 (1.19) | 981.36 (501.17) | 2321.58 (1.30) | 1053.26 (467.23) |
| EA | 2320.42 (1.17) | 1683.02 (247.80) | 2319.98 (1.10) | 1526.56 (296.02) |
| FAHC | 2319.86 (1.76) | 1083.26 (505.30) | 2319.98 (1.41) | 953.40 (587.77) |
| SAHC | 2317.46 (1.88) | 1921.00 (178.89) | 2319.52 (1.99) | 1404.02 (461.24) |
| TS(FA) | 2319.16 (2.35) | 834.26 (427.57) | 2319.24 (1.69) | 800.56 (583.02) |
| TS(SA) | 2318.28 (1.83) | $>$ 2000 (-) | 2319.34 (2.00) | 1346.98 (388.00) |

Table 1: A Summary Of Optimiser Performance

Some points can be made about the optimisers in general; before each technique is discussed individually. The type of hillclimbing that the optimiser was based on was found to be important. Stochastic hillclimbing-based optimisers significantly outperformed first and steepest-ascent based optimisers for both types of neighbourhood; with iterated stochastic hillclimbing found to be the method of choice for this problem.

The choice of neighbourhood made little difference for stochastic and first ascent optimisers. However, for a steepest-ascent optimiser, the swap-neighbourhood gave better quality solutions. A possible explanation may be the different size of the neighbourhoods. The moves $move(i, j)$ and $move(j, i)$ are equivalent for a swap neighbourhood; however for a shift neighbourhood this is not the case. So swap neighbourhoods are roughly half the size of shift neighbourhoods. This makes a difference when the whole neighbourhood has to searched before a move is accepted, as is the case for a steepest-ascent hillclimber.

## 5.1 Iterated Hillclimbing

Iterated hillclimbing gave a significant improvement in solution quality for stochastic (shift-only), first-ascent (both), and steepest-ascent (swap-only) hillclimbers. Also, performance was reasonably robust with respect to the iteration interval.

## 5.2 Simulated Annealing

Compared against stochastic hillclimbing, solution quality was increased, but the improvement was not as pronounced as for iterated hillclimbing. Low initial temperatures were found to be most effective. The improvements attained may be a result allowing some backwards moves initially, which help to find a good region of the search space, followed by rapid exploitation of that region later in the search.

### 5.3 Threshold Accepting

The best results were obtained with a low initial threshold of one, but again the results were not quite as good as those obtained for iterated stochastic hillclimbing. Taking the result with SA into account, it appears that there are gains to be made by allowing a little exploration; but this must be used sparingly.

### 5.4 Evolutionary Algorithms

For all the cases considered here, the EA performed less well than a stochastic hillclimber, in both terms of solution quality and speed to solution. The choice of population model was found to be important. The steady state model gave the best performance, followed by the (N+N) ES model. The generational models performed worst of all. In addition, crossover seemed to have no positive effect, and in some cases a high crossover probability degraded solution quality slightly. However it is unclear whether this lies with the type of recombination operator used, or with population-based search in general; this will require further investigation.

### 5.5 Tabu Search

In most cases, the level of performance is not significantly affected by the length of the tabu list, the quality obtained being comparable to the basic hillclimber. The exception to this was steepest-ascent tabu search with a shift-neighbourhood. As discussed earlier, with SAHC-based optimisers the number of moves that have to be evaluated before a move is accepted is large, so it is possible that this is an effect of the tabu list reducing the size of the neighbourhood that has to be searched.

## 6 Additional Points

So what do these results tell us about the nature of the fitness landscape? We conjecture that the search space contains a number of local minima, all of comparable quality and with a large basin of attraction. Inside these basins, local minima are not common, but there are fitness plateaus on the way down. This is supported by the following evidence. The observation with SA, TA, and EAs that allowing backward steps can degrade solution quality when compared with hillclimbing unless applied sparingly, suggests that local minima do not present a large problem in practice. Also, stochastic hillclimbing-based optimisers appear to perform better, on the whole, than first-ascent and steepest-ascent optimisers. A type of landscape that exhibits this kind of behaviour is a landscape with fitness plateaus; as the additional fitness gradient information obtained by first and steepest ascent hillclimbers is then of no help, but expensive in terms of the number of evaluations required. Additionally, the existence of plateaus seems likely when you examine the plan builder — swapping and shifting sites around often makes no difference to the plan generated.

## 7 Adding A Shipment Distance Constraint

So far, the assumption has been made that the shipment can travel over any distance — this is in fact not true. For the real-world problem this system is aimed at, traversing over the catchment area of more than 3 sites is infeasible. Removing this assumption may reduce the ability of the system to supply all of the sites as some may have to be supplied from quite a distance.

The hypothesis that imposing a constraint on shipment distance would affect the coverage attainable, was tested by running a stochastic hillclimber (iteration gap 250, shift neighbourhood) 50 times, with the constraint that the shipments cannot be made further than a maximum value. The components of the fitness function are given in Table 2.

As would be expected, tightening the maximum distance constraint, degrades the solution quality attainable. Taking the maximum allowed shipment distance to be 3, then the coverage

| Distance | Sites Covered | Shipments | Distance |
|---|---|---|---|
| 1 | 159.90 (0.30) | 43.92 (0.27) | 43.92 (0.27) |
| 2 | 188.56 (0.50) | 65.28 (2.96) | 88.54 (4.30) |
| 3 | 227.00 (0.00) | 91.82 (0.82) | 188.62 (2.16) |
| 4 | 231.90 (0.46) | 97.68 (1.36) | 231.26 (7.26) |
| 5 | 241.90 (0.30) | 105.66 (2.07) | 295.42 (9.27) |
| 6 | 243.00 (0.00) | 107.88 (1.24) | 303.08 (8.76) |
| 7 | 243.00 (0.00) | 108.06 (1.36) | 312.22 (9.80) |
| 8 | 243.00 (0.00) | 108.04 (1.15) | 313.78 (9.56) |

Table 2: Performance Components with Maximum Shipment Distance (Metric 1)

attainable appears to be around 90%, significantly lower that the 96% coverage attained without the constraint; though this could be a result of those 6being too far away to supply.

# 8 Using Shipment Distance As An Optimisation Criterion

It would be desirable for the distance of the shipments to be minimised as well. This is performed to a certain extent by the plan builder which uses the closest site available; but this preference is not expressed in the evaluation function. Therefore, the optimiser may favour plans which use long shipments to reduce the number of shipments, rather than use two short shipments.

The evaluation function was modified to be a linear combination of the number of targets met ($targets\_met$) and the total *distance* covered by the shipments made ($distance$).

$$fitness = 10 \times targets\_met - distance \tag{2}$$

To see if the sensitivity of performance to the maximum shipment distance was reduced as a consequence of the new fitness function, a stochastic hillclimber (iteration gap 250, shift neighbourhood) was run 50 times, with a shift neighbourhood (Table 3).

| Distance | Sites Covered | Shipments | Distance |
|---|---|---|---|
| 1 | 159.90 (0.30) | 43.92 (0.27) | 43.92 (0.27) |
| 2 | 188.50 (0.50) | 65.18 (2.36) | 87.48 (3.58) |
| 3 | 226.90 (0.36) | 92.72 (1.11) | 186.16 (1.75) |
| 4 | 230.88 (0.86) | 97.94 (1.55) | 212.48 (5.02) |
| 5 | 239.98 (0.42) | 105.82 (1.98) | 254.12 (5.66) |
| 6 | 242.12 (0.38) | 107.02 (2.68) | 262.00 (10.54) |
| 7 | 243.00 (0.00) | 111.56 (2.27) | 291.62 (4.27) |
| 8 | 243.00 (0.00) | 111.46 (2.53) | 291.14 (5.06) |

Table 3: Performance Components with Maximum Shipment Distance (Metric 2)

Again, the coverage attained is 90% when the constraint of a maximum shipment distance of 3 is imposed. Also as expected, using distance as a fitness criterion lead to a lower distance solution, though at the expense of using more shipments.

# 9 An Evaluation Using A Large Dataset

The experiments so far have been performed on a relatively small domain. A practical system will have to deal with more than this. So a larger data-set was used, consisting of 283 sites and 12 resources, based on actual data as before.

Stochastic hillclimbing was used in this evaluation, using the first of the evaluation criterion examined. and with the constraint that a shipment cannot go further than 3 site catchment-areas.

Before the system was run, only 19% of the resource targets were met (660 out of 3398 resource-sites). The initial solution met about 54% (1830 resource-sites) of the targets when processed by the plan builder.

Optimisation was then used for 2000 evaluations (which took about 10.5 minutes on a SPARC 5 workstation, or 7-8 hours on a 386-class PC). This increased the number of resource targets met to 57% (1959 resource-sites). The search was then extended to 20,000 evaluations. This gave a best solution of 58% (2002 resource-sites); though it still appeared that the local optimum was not reached at this point.

The lower coverage obtained, compared with the small dataset, was found largely not to be due to any failing of the system, but that at least 30% of sites in the data set were inaccessible to any of the sites with surpluses. This was due to a combination of the sites with surpluses being heavily concentrated (spatially) at one end of the dataset, and the maximum shipment distance constraint.

# 10 Extensions

The problem above has been simplified somewhat, mainly in the constraints and evaluation criteria used. Addressing this simplification will require further discussions with the eventual end-users to ascertain the exact requirements of the final system.

Some form of directed neighbourhood operator (e.g. [Ross *et al.* 94]) could be devised. For example, a site may be experiencing a shortage due to it being too late in the sequence to the plan builder. So preferentially applying operations which push sites with shortages closer to the front may speed the search by making moves that are more likely to improve performance occur sooner.

Finally, the shipment-plan builder could be improved. For instance, if an operator modifies position $N$ and $N+1$, the shipments to sites before position $N$ will remain unchanged; therefore only computing the changes to the shipment plan should result in a significant speedup; this approach has proved successful for timetabling problems [Ross *et al.* 94].

# 11 Conclusion

The system outlined above was constructed as a test of the applicability of some modern optimisation techniques to a real-world problem of resource management. The prototype constructed was able to quickly produce a workable redistribution plan for both small and full-sized versions of the problem which increased dramatically the number of resource targets met. Optimisation can improve upon this still further; but the size of the search space means that this will take some time for large problems. Fortunately the 'anytime' characteristic of the system means that this is not so much of a problem. Also, experiments with the full-sized dataset have indicated that the constraints upon the problem may often prevent all the resource targets being met, even if there are sufficient resources available.

A comparison of the optimisers showed that stochastic hillclimbing was the method of choice. The type of neighbourhood operator used was not found to make any real impact upon performance. Also, the performance of the Evolutionary Algorithm was found to be particularly disappointing.

In summary, these results bode well for attempts to scale the problem up to its full complexity. Although more work needs to be performed before the system could be deployed in a real-world situation.

# 12 Acknowledgements

We would like to express our gratitude to the Engineering and Physical Sciences Research Council (EPSRC) for their support of Andrew Tuson via a research studentship (95306458).

## References


[Burke *et al.* 95] E. Burke, D. Elliman, and R. Weare. The Automated Timetabling of University Exams using a Hybrid Genetic Algorithm. In *The AISB Workshop on Evolutionary Computing*, 1995.

[Dueck & Scheuer 90] G. Dueck and T. Scheuer. Threshold Accepting: A General Purpose Optimisation Algorithm Superior to Simulated Annealing. *Journal of Computation Physics*, 90:161–175, 1990.

[Fang 94] Hsiao-Lan Fang. *Genetic Algorithms in Timetabling and Scheduling.* Unpublished PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1994.

[Glover 90] F. Glover. Tabu Search: A Tutorial. *Interfaces*, 4:445–460, 1990.

[Holland 75] John H. Holland. *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press, 1975.

[Kirkpatrick *et al.* 83] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[Mott 90] G. F. Mott. Optimising Flowshop Scheduling Through Adaptive Genetic Algorithms. Chemistry Part II Thesis, Oxford University, 1990.

[Reeves 93] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Publications, 1993.

[Reeves 95] C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers & Ops. Res.*, 22:5–13, 1995.

[Ross *et al.* 94] Peter Ross, Dave Corne, and Hsiao-Lan Fang. Improving evolutionary timetabling with delta evaluation and directed mutation. In Y. Davidor nd H-P. Schwefel and R. Manner, editors, *Parallel Problem-solving from Nature - PPSN III*, LNCS, pages 566–565. Springer-Verlag, 1994.

[Tuson 96] A. L. Tuson. A Prototype Resource Management System For The Third World. Working Paper, Department of Artificial Intelligence, Edinburgh University, 1996.