Emergency Resource Redistribution in the Developing World: Towards a Practical Evolutionary/Meta-Heuristic Scheduling System

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 in aid 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 a relief program. A system is required that can suggest a workable redistribution plan. Initial results for a prototype system based on a heuristic plan generator and evolutionary/meta-heuristic optimisers, on real data, were encouraging; however more work still needs to be undertaken before the system can be deployed in the real-world. This paper describes some ongoing work to address some of these issues.

1 Introduction

Dealing with aid projects in the developing world can be made difficult by the fact that, though an effective policy exists, the actual implementation poses many obstacles. For instance, situations may occur where regional imbalances arise: an example would be a treatment site with sufficient diagnostic kits for a disease, but with no drugs to treat the diagnosed cases.

Recent work [11, 12] described a prototype system resource management system, based upon a heuristic plan builder, an indirect [1] solution encoding and a range of evolutionary/meta-heuristic techniques that optimised the sequence of relief sites to be considered by the plan builder. The work also evaluated the feasibility of this system to recommend a shipment plan to minimise such shortages in real-world situations.

The initial results obtained, on real data, were encouraging; though more work needs to be undertaken before the system can be deployed in the real-world. This paper describes some ongoing work to address some of these issues.

2 Recap of Earlier Work

NOTE: as space is limited, a full description of the initial system cannot be given here. Therefore this paper is written assuming that the reader is familiar with the material in [11, 12]. These papers available from the author's WWW page: `http://www.dai.ed.ac.uk/students/andrewt/`.

A brief description of the problem now 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)'. The aim is then to maximise the number of resource targets met (referred to as resource-sites), whilst minimising the number of shipments. This is subject to constraints on feasibility, for example, a site cannot supply more of a resource than it has.

The hardware limitations are also severe — 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 generate plans for regions with at least 250 relief centres within several hours.

3 A Post-Mortem of the EA

The initial study considered a range of optimisers alongside an evolutionary algorithm (EA), including tabu search (TS) [2], simulated annealing (SA) [4], and stochastic hillclimbing (SHC). Perhaps surprisingly, the EA performed particularly poorly, which may be of concern if the final problem is cast as a Pareto-optimisation problem, or if multiple solutions are desired (areas were EAs are thought to possess advantages).

One possibility is that the crossover operator (Modified PMX [7]) used in the initial study may have been unsuitable as it did not respect the correct type of building block. This idea was advanced by Goldberg as the 'Principle of Meaningful Building Blocks' [3]. Modified PMX is a *Position-based* crossover operator — it respects [8] building blocks such as 1 * * 4 5. Crossover operators have been devised that respect *edge* and *precedence* building blocks. In fact, recent work has proposed such a taxonomy of sequencing operators [6].

An argument can be made for assuming that precedences are sensible building blocks for this problem. The argument is as follows: consider two shortages of a given resource in (a possibly larger) problem. Let us label these sites A and B. There are 2 sources nearby (called X and Y), each of which can only supply one of the sites. However, though X

can supply A or B, Y can only supply B. If A is processed before B by the plan builder all is well. The problem arises when B is processed before A as it is possible that the plan builder will supply B from X, thus leaving A unsupplied. Therefore what is important is that A is before B in the sequence (a precedence), and not their absolute positions in the sequence, or that A and B are adjacent (an edge).

This is supported by empirical evidence: for the EA, the shift operator gave slightly better quality solutions than the swap operator (for the other SHC-based optimisers, no significant difference was found). An analysis of these operators shows that the shift operator is less disruptive of precedences than the swap operator (though this is reversed for position building blocks).

To this end, two additional crossover operators were considered: Enhanced Edge Crossover (EEX) [10], and Precedence Preserving Crossover (PPX) [6]. The EA was run 20 times for each crossover operator over a range of crossover probabilities (0.0 to 1.0) and population sizes (10 to 100).

3.1 The Effect of the Crossover Operator

The effect of crossover probability (with mutation applied otherwise) when using a rank-based, steady-state GA, with population 100 was investigated. Figure 1 shows a plot of the average (over 50 runs) fitness obtained in 2000 evaluations against a range of crossover rates (no significant impact on the speed of search was observed).

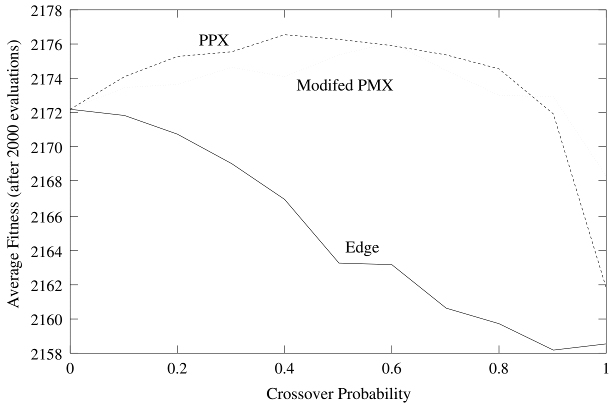

Figure 1: A Plot of the Effect of Crossover Operators and Probabilities

As is quite evident upon inspection, PPX does produce a consistent, though modest, improvement over Modified PMX, for all but the highest crossover probabilities (though this would be expected if precedences were important as the disruptive effect of Modified PMX on the precedences would be analogous to mutation). Therefore the hypothesis that precedences are suitable building blocks for this problem appears to be correct. Unfortunately, the performance of the GA still lags behind the stochastic hillclimber; which was for SHC: $\mu_{fitness} = 2178.18$, $\sigma_{fitness} = 0.82$, $\mu_{evals} = 958.58$, $\sigma_{evals} = 523.87$ — significantly better than $\mu_{fitness} = 2176.54$, $\sigma_{fitness} = 2.06$, $\mu_{evals} = 1697.74$, $\sigma_{eval} = 196.72$ for the EA. This may be a sign that the problem lies with the population-based nature of the search itself, rather than with the crossover operator.

3.2 The Effect of Population Size

To check if the suspicion above was valid, the population size of the EA was progressively reduced from 100 to 10. If the gains obtainable by searching with a population, such as recombination and the use of population statistics to guide the search, are being outweighed by the costs then it would be expected that a smaller population would lead to improved performance. Figure 2 shows plots of the average (over 50 runs) fitness and number of evaluations to final solution obtained in 2000 evaluations against population size for each of the crossover operators (the crossover probability was re-tuned for each population size).

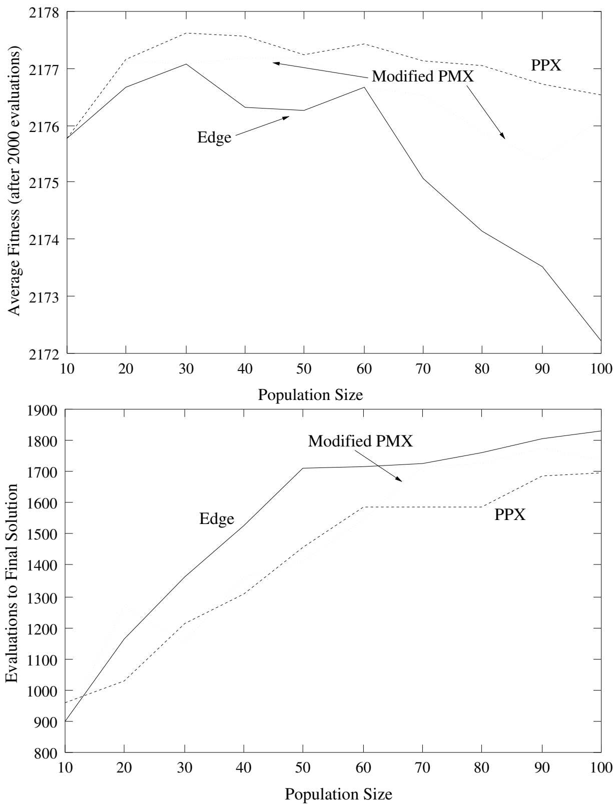

Figure 2: Plots of the Effect of Population Size

The plots show that, as far as fitness is concerned, a small population size (about 30-50 for PPX) is the best choice, though the fitness obtained in the best case is not as high as obtained by SHC. However when the speed of search is considered, the results indicate that the presence of a population affects search adversely — only with a population size of 10 was the search speed comparable to that of SHC. It would appear that the benefits population-based search can bring do not exhibit themselves sufficiently to be of use in this problem.

4 An Upper Bound Procedure

In the initial study, it was found that the sites with shortages were heavily concentrated (spatially) at one end of the dataset. When this observation was combined with the maximum shipment distance constraint, it became immediately apparent that, in the large, about 30% of the resource-sites in the data set were inaccessible to any of the sites with surpluses. To get a more accurate figure as to the number of sites that could be supplied, the following procedure for estimating an upper bound of the number of resource sites that can be supplied was devised.

Each site is considered in turn, then each resource, i, is considered by the following procedure:

1. If the current site has a shortage of the resource under consideration, and a supply can be found, then this site is a *supplyable shortage* of that resource (num_short_i denotes the number of sites that this applies to).

2. If the current site has a shortage of the resource under consideration, but NO supply can be found, then this site is considered a *unsupplyable shortage* of that resource (un_short_i).

3. If the current site has a surplus of the resource under consideration, and a site that has a shortage of that resource can be found that the current site can supply, then this site is a *supplyable surplus* of that resource (where the total is given by ok_i).

4. If the current site has a surplus of the resource under consideration, and NO site that has a shortage of that resource can be found that the current site can supply, then this site is a *unsupplyable surplus* (un_i).

5. Otherwise, the current site has a *satisfactory* level of that resource (sat_i).

A tally is kept up of the the total amounts of the supplyable surpluses and shortages ($surplus_i$ and $short_i$) of each resource and the number of sites in each category. The upper bound can then be calculated as follows, where res is the number of resources under consideration:

$$bound = sup + \sum_{i=1}^{res} sat_i + un_i + ok_i$$
$$sup = \sum_{i=1}^{res} min\{1.0, surplus_i/short_i\} \times num\_short_i$$

The upper bound is effectively the number of resource-sites that are already satisfied, plus the number of resource-sites with supplyable shortages, weighted by the amount of resources available to supply those shortages. This approach does have its weaknesses — it ignores the possibility that shortages of a given resource may be in clusters, some of which may be in contact with a source of supply that does not have enough of a surplus to supply all of the sites in the cluster, although at the same time a supply site for another cluster may have a massive surplus. In these situations, the bounds procedure described here would over-estimate.

The analysis of the two test problems by the procedure is shown in Table 1. It indicates that it is not possible to deal with all of the shortages, and that the distribution of supplies across the data is very uneven. In addition, the upper bound allows us to evaluate the performance attained by this system in a more positive light as we are comparing the performance of the system against a more attainable target.

5 Exploiting Constraints

Most of the domain knowledge is held in the plan builder and therefore it is one of the most important aspects of the system to get right (especially as most of the CPU time is spent in there). Two approaches to improving the plan builder are now investigated. For the remainder of the investigations in this report, stochastic hillclimbing will be used. The reasons are three-fold: first, SHC was the best performer in the initial study; second, for the large dataset it takes a very long time for the local optima to be located, usually much longer than the optimiser has to run, thus the additional machinery of the EA is not required; third, the arguments that are used in the design of the enhancements below also apply to the EA, — in fact this has been observed empirically [9].

Needless to say, if sites that cannot be supplied are a regular occurance then there would be no point in considering them in the search. Therefore we could exploit these constraints to reduce both the search space and the amount of effort involved in constructing a shipment plan.

To reduce the search space, the following reasoning was employed: as the search space is the order in which the sites have their shortages dealt with, the only sites we need deal with are those with shortages. Furthermore using the information obtained by the bound calculation procedure, we can further restrict this to the sites that can be somehow supplied. Therefore each site was considered in turn, and if it contained no shortages which could be supplied, it was removed from the search space. Apart from reducing the search space, we are reducing the number of redundant moves which should speed up search. To see why, consider a site X that has no supplyable shortages, any move that involves X would result in no change in the final shipment plan, as no resources will be moved by the shipment plan builder.

In addition, the bound calculation procedure shows us which sites are supplies for which resources. This information could be used to prevent the plan builder having to examine sites that we already know are not sources of supply for that resource; this was incorporated into the plan builder.

	Small Dataset	Large Dataset
Upper Bound:	242 (cf. 252)	2623 (cf. 3396)
Supplyable Surpluses:	82	605
Unsupplyable Surpluses:	0	13
Satisfactory Resource Levels:	0	42
Supplyable Shortages:	166	1963
Unsupplyable Shortages:	4	773

Table 1: Results of the Analysis by the Bound Calculation Procedure

5.1 Results Obtained

When the constraints were exploited by the system, no reduction in the search space was produced for the small dataset, and a modest reduction was obtained for the large dataset (from 283! to 278!). A closer examination of the datasets showed that shortages of different resources occur in different regions of the datasets, and thus very few sites are without some kind of a shortage. It should be noted that although the datasets were drawn from real examples, they were selected so to be challenging — most situations are not as difficult as this. Fortunately, the reduction in the number of resource-sites that have to be considered by the plan builder was more encouraging: from 252 to 82 for the small dataset, and from 3396 to 605 for the large dataset.

To see whether this resulted in an actual improvement in performance, stochastic hillclimbing with a shift-neighbourhood was run 20 times for 2000 evaluations. The following performance indicators were recorded: actual (wall-clock) time to perform 2000 evaluations on a SPARC 5, the number of evaluations until the best solution was found, the overall solution quality, and the number of resource-sites that met their targets. The results obtained are shown in Table 2, with standard deviations given in brackets. Where differences in performance have been reported to be significant, this has been ascertained using a t-test.

It was found that this approach produced a significant improvement in the wall-clock time. Search space reduction was also found to be effective in increasing the quality of the final solution, both in terms of overall fitness, and the number of resource-sites that met their targets (1975 resource-sites is just over 75% of the upper bound).

6 Intelligent Initialisation

The reasoning that sites with the most problems should be earlier in the sequence, may be exploited in the initialisation procedure. If we can identify which sites have the most shortages and place then at the start of the initial solution, we may obtain a better quality solution.

Therefore the following procedure was used: generate a random solution, calculate a heuristic value for for each site, then apply a sorting procedure (in this case quick-sort) to place sites with the most supplyable shortages at the start of the sequence. The following two heuristics were used:

1. Using the bounds procedure, obtain the number of shortages at each site that are supplyable and use this as the heuristic.
2. Evaluate a randomly generated string using the plan builder and then, with the help of the bounds procedure, finding the number of supplyable shortages at each site assuming that the plan has been implemented.

Both methods employ the belief that the sites that have shortages should be placed earlier in the sequence. However, the second method aims to overcome the possibility that the first heuristic could be mislead by virtue that, in the initial situation, although a site may have many shortages these shortages are easily resolved by the plan builder, whereas a site with fewer but more difficult to supply shortages would be placed later in the sequence by the first heuristic. Of course the second heuristic overcomes this problem at the cost of an extra evaluation, but this may be worth the effort.

The first experiment was to ascertain if the initialisation procedures do, in fact, produce a better quality initial solution. Therefore each of the 2 initialisation methods were used 1000 times and the quality of their solutions measured and compared against 1000 randomly-generated solutions.

Table 3 summarises the results obtained for the large dataset. There appears to be some increase in the quality of solutions generated by the 2 heuristics but a t-test indicated that this difference was not significant. In addition, it also appeared that using the plan builder to initialise did not result in any improvement above using the bounds procedure alone, despite its higher computational cost.

Attention was then turned to whether these initialisation methods would result in any gains when the optimiser was run. Again stochastic hillclimbing and the large dataset were used in this evaluation. In each case, stochastic hillclimbing was run 20 times for 2000 evaluations. Figures 3 and 4 summarise the results obtained for the shift neighbourhood.

For both neighbourhoods, it was evident from an inspection of the t-value plots that initialisation using the bounds procedure was not effective — presumably due to the concerns about this method discussed earlier. On the other hand, initialisation using the evaluation function was effective in improving solution quality. This effect seemed to diminish in the later stages of the run which would not be unexpected as the rate of progress of the hillclimber also diminishes with time, thus allowing the randomly initial-

	Time	Evaluations	Quality	Resource-Sites
No Reduction	704s	1975.65 (21.65)	18619.45 (56.30)	1958.30 (5.95)
Reduction	671s	1966.10 (27.92)	18796.40 (50.42)	1975.10 (5.36)

Table 2: Results Obtained when the Plan Builder was Modified

	Random	Using Bounds	Using Plan Builder
Quality	16988.93 (123.31)	16995.66 (124.73)	16993.86 (126.05)
Resource-Sites	1793.03 (13.08)	1793.67 (13.18)	1793.56 (13.32)

Table 3: Comparison of Initialisation Methods over 1000 Initialisations

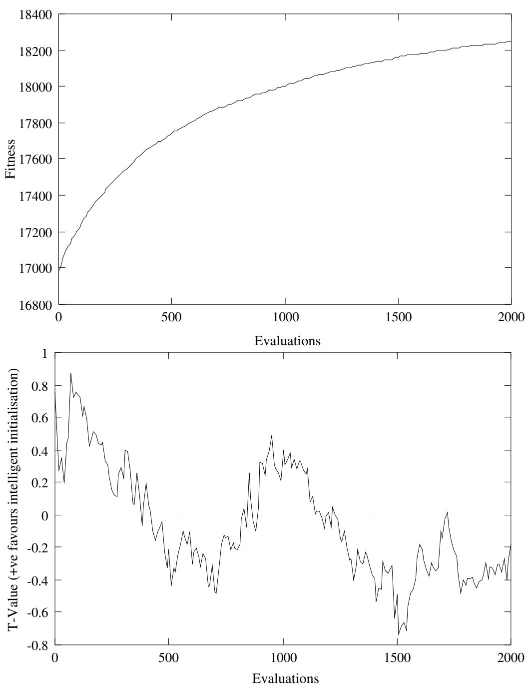

Figure 3: Plots of the Effect of Initialisation using the Bounds Procedure (Shift Neighbourhood)

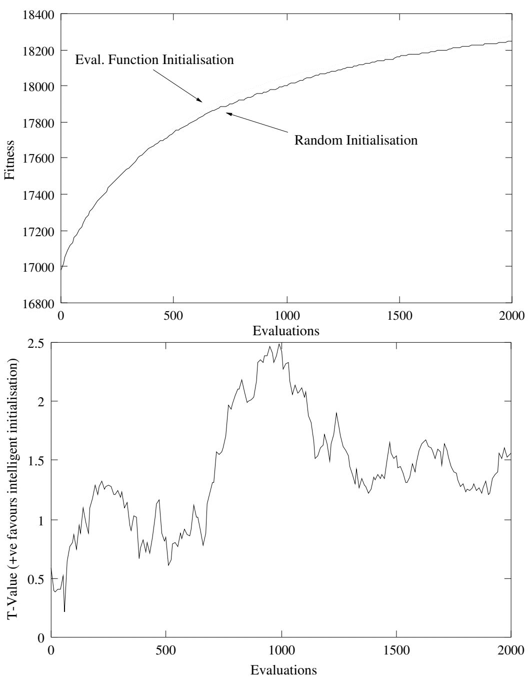

Figure 4: Plots of the Effect of Initialisation using the Evaluation Function (Shift Neighbourhood)

ised hillclimber to ‘catch up’ on the intelligently initialised hillclimber.

Furthermore, it would appear, on the basis of these results that improvements (or lack of them) in initial solution quality is a poor predictor of initialisation method effectiveness when the optimiser is run for some period of time.

7 Interactive Scheduling

In the real world, constraints are not set in stone. However it is not a trivial task to represent which constraints to relax to the optimiser. Some constraints may only be active in certain situations, one way around this is to allow the user to interactively decide which constraints to relax. In addition, a problem may have an underlying structure that allows assumptions to be made that the system would not know about. One example of this is how to decompose a larger problem into smaller subproblems in an acceptable manner. Finally, there may be sources of supply that the system does not know about (for instance airlifts).

However, in order for this to be achieved, the user has to have a picture of what the problem currently is, and where the bottlenecks and dependences are. We propose that the plan builder and bound calculation procedure can be used to assist in this process.

How can this be performed? The first step is to run the bounds procedure. This allows us to identify which sites cannot be supplied with which resources. Then the plan builder is run with a random sequence, as even without optimisation the builder produces a feasible plan, which gives us some idea of the distribution of shipments, and of which parts of the plan are bottlenecks. Displaying this information to the user, then allows the user to make informed strategic decisions about supply policy and logistics.

If the user is the given the facility to relax constraints, suggest shipments, and to partition the problem into subproblems, then he can explore the effect of these changes by re-running the bounds procedure and plan builder with the new situation. One example is if there is a large area that has a shortage of a particular resource, then the user could order a special shipment or airlift into the area, and tell the system that he has done so before the plan builder is rerun.

In addition, the user can change the order of sites of the sequence, thus only applying the optimiser if desired, and allowing the user to find a good initial sequence for the optimiser to improve. Once the user is satisfied with the overall situation, the optimiser can then be run to 'smooth the rough edges' off of the plan. Direct representations, which often allow infeasible plans, would have some disadvantages in this regard as the user needs to have a sensible starting plan to work on, which such an approach may not be able to guarantee.

This approach thus delegates decisions between the user and the system according to their strengths. Human schedulers have a far better idea of what can and cannot be done, and of the wider issues and goals of the logistical problem which are difficult, if not impossible, to represent to a computer system — therefore they are better at making strategic decisions. On the other hand, drawing such plans by hand is slow and laborious and thus leaves little room for further optimisation — human schedulers usually take the first feasible schedule that they can find [13]. Thus an automated system allows a wider space of possible plans to be considered. Finally, and most importantly, plans that have had a human contribution to their creation are more likely to be adopted [5], an issue that the authors feel is essential to address if such a system is to be successfully used in practice.

8 Conclusion

This paper has described ongoing work into producing a system for the redistribution of resources in the developing world. Two main points have been identified and addressed in this paper: first, the need for domain knowledge to be fully exploited in all parts of the system in a principled manner; second, that it is not feasible, or even desirable, for the system to do all of the work, and therefore such a system should be designed to allow human intervention.

9 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), and Keith Anderson for final proofreading.

References

[1] 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.

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

[3] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading: Addison Wesley, 1989.

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

[5] B. MacCarthy, S. Crawford, C. Vernon, and J. Wilson. How do Humans Plan and Schedule? In *The Third Workshop on Models and Algorithms for Planning and Scheduling Problems*, 1997.

[6] D. Mattfeld, C. Bierwirth, and H. Kopfer. On Permutation Representations for Scheduling Problems. In H.-M. Voigt et al, editor, *Parallel Problem-solving from Nature - PPSN IV*, LNCS, pages 310–318. Springer-Verlag, 1996.

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

[8] N. J. Radcliffe. Equivalence Class analysis of genetic Algorithms. *Complex Systems*, 5(2):183–205, 1991.

[9] P. Ross and A. Tuson. Directing the Search of Evolutionary and Neighbourhood-Search Optimisers for the Flowshop Sequencing Problem with an Idle-Time Heuristic. In *The 1997 AISB Workshop on Evolutionary Computing*, 1997.

[10] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A comparison of genetic sequencing operators. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76. San Mateo: Morgan Kaufmann, 1991.

[11] A. Tuson, R. Wheeler, and P. Ross. A Prototype Emergency Resource Redistribution System For Disease Control Programmes. In *The 15th Workshop of the UK Planning and Scheduling Special Interest Group*, 1996.

[12] A. Tuson, R. Wheeler, and P. Ross. An Evolutionary/Meta-Heursitic Approach To Emergency Resource Redistribution In The Developing World. In *The Third International Conference On Artificial Neural Networks And Genetic Algorithms*, 1997.

[13] V. Wiers and K. McKay. Task Allocation: Human Computer Interaction in Intelligent Scheduling. In *The 15th Workshop of the UK Planning and Scheduling SIG*, 1996.