The Trade off Between Diversity and Quality for Multi-objective Workforce Scheduling*

Peter Cowling, Nic Colledge, Keshav Dahal, and Stephen Remde

MOSAIC Research Group, University of Bradford, Great Horton Road, Bradford, BD7 1DP, Great Britain
{p.i.cowling, n.j.colledge, k.p.dahal, s.m.remde}@bradford.ac.uk

Abstract. In this paper we investigate and compare multi-objective and weighted single objective approaches to a real world workforce scheduling problem. For this difficult problem we consider the trade off in solution quality versus population diversity, for different sets of fixed objective weights. Our real-world workforce scheduling problem consists of assigning resources with the appropriate skills to geographically dispersed task locations while satisfying time window constraints. The problem is NP-Hard and contains the Resource Constrained Project Scheduling Problem (RCPSP) as a sub problem. We investigate a genetic algorithm and serial schedule generation scheme together with various multi-objective approaches. We show that multi-objective genetic algorithms can create solutions whose fitness is within 2% of genetic algorithms using weighted sum objectives even though the multi-objective approaches know nothing of the weights. The result is highly significant for complex real-world problems where objective weights are seldom known in advance since it suggests that a multi-objective approach can generate a solution close to the user preferred one without having knowledge of user preferences.

1 Introduction

In collaboration with an industrial partner we have studied a workforce scheduling problem which is a resource constrained scheduling problem similar to but more complex and "messy" than many other well-studied scheduling problems like the RCPSP (Resource Constrained Project Scheduling Problem) and job shop scheduling problem, for which much work has been done [1]. The problem is based on our work with Vidus Ltd. (an @Road company) which has developed scheduling solutions for very large, complex mobile workforce scheduling problems in a variety of industries. Our workforce scheduling problem is concerned with assigning people and other resources to geographically dispersed tasks while respecting time window constraints and is like many scheduling problems NP-Hard [1] because it contains the RCPSP as a sub-problem.

* This work was funded by an EPSRC CASE Studentship in partnership with Vidus Ltd. (an @Road company).

J. Gottlieb and G.R. Raidl (Eds.): EvoCOP 2006, LNCS 3906, pp. 13 – 24, 2006.

This paper is structured as follows: we outline a model of our workforce scheduling problem in section 2, discuss related work in section 3 and propose a multi-objective genetic algorithm for solution of the workforce scheduling problem in section 4. In section 5 we investigate multi-objective genetic algorithms and compare them to other single objective genetic algorithms in terms of solution quality and diversity. We present conclusions in section 6.

2 The Workforce Scheduling Problem

The workforce scheduling problem that we consider consists of four main components: Tasks, Resources, Skills and Locations. A *task* T_i is a job or part of a job that needs to be completed. Each task must start and end at a specified location. Usually the start and end locations are the same but they may be different. Each task has one or more time windows. Some time windows which are an inconvenience for the customer have an associated penalty. We have a set $\{T_1,T_2,\ldots,T_n\}$ of tasks to be completed. Each task is undertaken by one or more resources. We have set of resources $\{R_1,R_2,\ldots,R_m\}$. A task requires resources with the appropriate skills. We have a set $\{S_1,S_2,\ldots,S_k\}$ of skills. Task T_i requires skills $[\mathrm{TS}_1^i,TS_2^i,\ldots,TS_{t(i)}^i]$ with work requirements $[\,w_1^i,w_2^i,\ldots,w_{t(i)}^i]$ where w_q^i is the amount of skill TS_q^i required. Task T_i also has an associated priority $p(T_i)$. Resources are the components that undertake the work and possess skills. Resource R_j possesses skills $[\mathrm{RS}_1^j,RS_2^j,\ldots,RS_{r(j)}^j]$. A function $c(R,S)$ expresses the competence of resource R at skill S, relative to an average competency. Each resource R travels from location to location at speed $v(R)$. For tasks T_1, T_2, $\mathrm{d}(T_1, T_2)$ measures the distance between the end location of T_1 and the start location of T_2. There are three main groups of constraints: task constraints, resource constraints and location constraints and they are described below.

Task constraints:

- Each task can be worked on only within specified time windows.
- Some tasks require other tasks to have been completed before they can begin (*precedence* constraints).
- Some tasks require other tasks to be started at the same time (*assist* constraints).
- Tasks may be split across breaks within a working day. No tasks may take more than one day.
- For a task to be scheduled it must have exactly one resource assigned to it for each of the skills it requires.
- All assigned resources have to be present at a task for its whole duration regardless of their skill competency and task skill work requirement.
- If a task T_i with skill requirements $[\mathrm{TS}_1^i,TS_2^i,\ldots,TS_{t(i)}^i]$ and amounts $[\,w_1^i,w_2^i,\ldots,w_{t(i)}^i]$ is carried out by resources $[\mathrm{R}_1^i,R_2^i,\ldots,R_{t(i)}^i]$ then the time taken is

$$\max_{q\in\{1,2,\ldots,t(i)\}}\left(\frac{w_q^i}{c(R_q^i,TS_q^i)}\right)$$

i.e. the greatest time taken for any single resource to complete a skill requirement

Resource constraints:

- A resource *R* travels from location to location at a fixed speed *v(R)*.
- Resources may only work during specified time windows.
- Resources can only work on one task at once and only apply one skill at a time.

Location constraints:

- Resources must travel to the location of each task they work on, and are unavailable during this travel time.
- Resources must start and end each day at a specified "home" location and must have sufficient time to travel to and from their home location at the start and end of each day.

2.1 Objectives

When building a schedule many different and often contradictory business objectives are possible. In this paper we consider three objectives. The first objective is Schedule Priority (*SP*), given by

$$SP=\sum_{\{i:T_i \text{ is scheduled}\}} p(T_i)$$

Maximising Schedule Priority maximises the importance of the tasks scheduled to the user (and implicitly minimises the importance of tasks unscheduled).

The second objective measures Travel Time (*TT*) across all resources. Define $A=\{(i1,i2,j)$: task T_{i1} comes immediately before T_{i2} in the schedule of resource $R_j\}$. Then,

$$TT=\sum_{(i1,i2,j)\in A}\frac{d(T_{i1},T_{i2})}{v(R_j)}$$

Travel to and from home locations is handled by considering dummy tasks fixed at the start and end of the working day, at the home location of each resource.

The third objective measures the inconvenience associated with completing tasks or using resources at an inconvenient time, which we have labelled Schedule Cost (*SC*). In order to express this accurately we express the time windows for Resource *R* using a function τ where $\tau(R,t)$ is the cost per unit time for resource *R* working at time *t*. We introduce a variable

$$\mathrm{X}(R,t)=\begin{cases}1 & \text{if resource } R \text{ is busy (on a task or traveling) at time } t\\ 0 & \text{otherwise}\end{cases}$$

Similarly we introduce τ' where $\tau'(T,t)$ is the cost per unit time for task *T* being executed at time *t* and

$$\mathrm{X}'(T,t) = \begin{cases} 1 & \text{if task } T \text{ is being executed at time } t \\ 0 & \text{otherwise} \end{cases}$$

Then

$$SC = \sum_{i=1}^{n} \int_t \mathrm{X}'(T_i,t)\tau'(T_i,t)dt + \sum_{j=1}^{m} \int_t \mathrm{X}(R_j,t)\tau(R_j,t)dt$$

Other objectives are possible but these three objectives express most of the primary concerns of the users in this case, at a high level. Considering lower level objectives at regional and resource group level could, however, give in many more objectives for this problem.

3 Related Work

3.1 Similar Problems

The RCPSP is a generalisation of several common scheduling problems including job-shop, open-shop and flow-shop scheduling problems [1]. The RCPSP consists of a set of Tasks to be performed using a set of finite capacity resources. In common with the model in the preceding section, the RCPSP has tasks which are undertaken by finite capacity resources subject to precedence constraints. However, the notion of tasks requiring skills and resources possessing multiple skills is essentially absent, as is the travel aspect. Precedence constraints are a major part of the RCPSP model and in many RCPSP problems the precedence constraints are arguably the most complex constraints with every task involved in a precedence relationship with one or more other tasks [1]. They are less significant for our model with many tasks having no predecessors or successors. Our problem has time-varying resource availability which is rarely considered in the RCPSP, however Brucker [2] discusses an RCPSP with time dependent resource profiles which model the changing availabilities of resources.

The resource constrained multiple project scheduling problem (rc-mPSP) [3] is less widely studied than the RCPSP. An rc-mPSP problem consists of several RCPSP problems that are not connected by precedence constraints, making the overall problem less time constrained than an equivalent RCPSP problem with the same tasks (and the extra precedence constraints). These precedence constraints are closer to the ones observed in our workforce scheduling problem.

A recent review of work on personnel scheduling is presented in [18]. In contrast to the RCPSP, personnel scheduling does include a notion of time-varying resource availability, in some cases resources availability is limited to certain times of day as for our problem. Some personnel scheduling problems such as crew scheduling also include travel between locations. However, personnel scheduling problems generally do not contain any notion of precedence, which is highly significant to both our problem and the RCPSP.

3.2 Solution Techniques for the RCPSP

Since their introduction by Bremermann and Fraser and the seminal work done by Holland [4], genetic algorithms (GAs) have been developed extensively to tackle problems including the travelling salesman problem, bin packing problems and scheduling problems (see for example [5]). [6] reviewed a range of meta-heuristics for solution of the RCPSP and showed genetic algorithms to work consistently well compared with other methods such as tabu search and simulated annealing. Many of the genetic algorithms presented in [6] worked by optimising the order in which tasks are passed to a schedule generation scheme (a greedy constructive heuristic). A serial schedule generation scheme (serial SGS) builds schedules by inserting tasks one by one into the schedule as early as possible. Most of the techniques reviewed in [6] use a serial SGS. Serial SGSs have been shown to be superior in general to parallel SGSs which works by taking a slice of time and inserting as many tasks as possible into it before moving on to another later slice of time. One such genetic algorithm is presented in [5]. Another more complex approach is presented in [7], here a genetic algorithm is used to generate a schedule in a similar fashion to [5], however once the schedule is generated, a forward backward improvement heuristic is used to improve the resulting schedule. The GA in [5] is shown to work well across a set of test RCPSP problems [6], however the genetic algorithm presented in [7] with forward backward improvement is shown to be one of the best approaches presented across the test problem set. None of the aforementioned approaches use real world problems or real problem data and so their performance under such conditions is unknown.

3.3 Multi-objective Scheduling Techniques

The most widely used method for combining multiple objectives in genetic algorithms is the weighted sum method. This can be problematic as practitioners estimated weights are often not a good reflection of the true requirements for a globally good solution. Simply put, a user is not used to being asked to explicitly define the relative importance of different problem goals, and the weights defined may reflect small local effects (since they are easy for the user to understand) rather than more difficult to define global ones.

Multi-objective approaches do not rank solutions directly as weighted sum approaches do; instead they use the notion of dominance and the distribution of solutions in objective space to decide the overall quality of a population of solutions. An important notion is that of a non-dominated solution, which is a solution where there are no other solutions better with respect to all the objectives. Multi-objective approaches try to maintain a set of solutions which are non-dominated and to get a good distribution of these solutions in objective space. This is useful in scheduling as the user no longer has to specify a set of specific weights representing the kind of schedule they think they are looking for, instead they can choose a schedule from a diverse set. A range of multi-objective genetic algorithms are among the most widely used approaches.

The Vector Evaluated Genetic Algorithm (VEGA) was proposed and compared with an adaptive random search technique in [8], which showed that VEGA out performed adaptive random search in terms of solution quality. VEGA finds a set of

non-dominated solutions and works by splitting the population randomly in to a number of subpopulations (the number of subpopulations being equal to the number of objectives to be considered). The population evolves with a mating pool created using a proportion of individuals from each sub-population with different objective functions. The technique is simple, however, each solution is only evaluated with one objective at any time and because of this eventually all solutions converge towards one solution with respect to a single objective [9]. In real world problems this convergence towards one objectives best solution is unworkable where we seek a trade off between objectives.

The Non-dominated Sorting Genetic Algorithm (NSGA) [10], and its enhancement NSGA-II [11] sort the population into non-dominated fronts. This is done by first identifying non-dominated individuals in the population, these are on the first front, the first front is then removed from the population and then non-dominated individuals are identified again, these comprise the second front. This process is repeated until all fronts are identified. The computational complexity of the algorithm has been reduced from O(mN^3) for NSGA to O(mN^2) for NSGA-II per generation, where m is the number of objectives and N is the population size; however the memory requirements have increased from O(N) to O(N^2). Elitism has been introduced in NSGA-II to force the algorithm to keep the extreme maximum and minimum solutions for all objectives, as has a new algorithm for crowding distance calculation. Crowding distance is a representation of the density of neighbouring individuals on any given front. Mating selection in NSGA-II is performed by binary tournament selection using the crowded comparison operator. If both solutions are on the same front, then the crowding distance is used as a tie breaker. The parent and child populations are combined and this new population (of size 2N) is ranked according to front. Fronts are then added to the next generation's population (starting with the non-dominated front) until the size exceeds N. Once this has been done, crowding distance assignment is applied to the last front that has been added and the crowded comparison operator is used to sort this final front and the worst individuals are removed to give the next generation's population.

The Strength Pareto Evolutionary Algorithm (SPEA) was introduced in [12], and further improved to SPEA2 in [13]. SPEA2 uses Pareto based ideas in both population selection and mating selection. Fitness assignment in SPEA2 works in two parts: domination, and distance to the kth nearest neighbour. The domination of an individual *i,* is calculated by counting the individuals dominated by *i*, this is the individuals *strength* value. Then, the *raw fitness R(i)* is calculated for each individual *i* by adding all the *strength* values of individuals dominated by *i*. The *distance* value σ_i^k is created for each individual *i* by finding the Euclidean distance to all other individuals in objective space and sorting them in ascending order. The k^{th} closest is taken where *k* is a user defined parameter ([13] recommends *k* parameter to be the square root of the combined size of the population and the archive population). Then a *density* value *D(i)* is calculated by the equation $D(i) = \frac{1}{\left(\sigma_i^k + 2\right)}$ ensuring that (in the final calculation of fitness) the raw fitness value takes precedence over it. Finally, the total fitness for each individual (*i*) can be calculated using $F(i) = R(i) + D(i)$.

SPEA2 uses an archive to allow individuals to survive from one generation to the next. At the end of a generation all non-dominated individuals from the archive and the population are copied to a new archive. If the archive is not filled by this process the best non-dominated individuals (according to the SPEA2 fitness function) from the population can also be copied into it. If however, the archive is overfilled by non-dominated individuals it is truncated by removing individuals based on their σ_i^k distance to other individuals in the archive.

NSGA-II and SPEA2 have been compared with SPEA and Pareto Envelope-Based Selection Algorithm (PESA) [14] on a set of test problems including the knapsack problem in [13]. NSGA-II and SPEA2 are shown to perform the best out of all tested. SPEA2 is said to be better in higher dimensional objective spaces and the solutions generated by SPEA2 are shown to dominate those generated by NSGA-II 80% of the time (on average).

4 Application of Multi-objective Genetic Algorithms to our Workforce Scheduling Problem

As the review of [6] showed genetic algorithms such as [7] and [5] perform well across a set of test problem instances, we have chosen to adopt a similar approach to our workforce scheduling problem. We use a serial schedule generation scheme to encapsulate the assignment of tasks to resources and constraint handling for our problem. A permutation based genetic algorithm optimises the order in which tasks are passed to this serial schedule generation scheme. We consider two multi-objective GAs to a single, weighted sum objective GA which uses binary tournament selection and elitist replacement. The multi-objective genetic algorithms will use NSGA-II and SPEA2 multi-objective approaches for the mating and environmental selection phases.

4.1 Serial Schedule Generation Scheme (SGS)

The serial schedule generation scheme we have built uses a permutation of tasks to generate a schedule. As our problem is more complex than traditional RCPSP problems (due the way skills, resources and task durations are defined) the task scheduling process within the SGS is split into two sub processes, resource selection and task insertion. This is also where many of the similarities between the RCPSP serial SGS methods defined in [15] and our method end. This is unfortunate as for the RCPSP solved with an activity list serial SGS there is always an order of tasks which will generate an optimal schedule when a regular performance measure is considered [15]. However, this is not the case with our SGS that because of the complexity of allocating resources having appropriate skills to a task requiring multiple skills, and time window constraints. The schedules generated will however always be feasible.

The motivation for divorcing resource selection from task insertion in this way is mainly due to the fact that until resources are selected for a task, there is no way of knowing exactly how long the task will take. This uncertainty causes many problems in the scheduling process, and is often the cause of tasks going unscheduled. Resources are selected by finding the intersection of the periods of time they have

available in common with the task time windows and the other resources already selected (here a larger amount of available time is preferable). Due to the consideration of travel and competency in resource selection ties rarely occur, however when they do they are broken randomly. Once all resources are chosen the task is then inserted as early as possible in the schedule.

5 The Competitiveness of Multi-objective Genetic Algorithms

Real world scheduling problems often have many and contradictory objectives making them an interesting testbed for comparing the performance of multi-objective genetic algorithms and weighted sum genetic algorithms. In our experiments we intend to investigate the competitiveness of multi-objective genetic algorithms when compared with weighted sum genetic algorithms in terms of both solution quality and diversity.

In our experiments diversity will be measured using the "Max Spread" measure from [9] and the "Morrison and De Jong" measure which is based around ideas taken from the moment of inertia in mechanical engineering [16]. Solution quality will be assessed for each population (whether from a multi-objective or weighted sum run) by several weighted sum objective functions. We do not expect that multi-objective methods will outperform all weighted sum methods when assessed by weighted sum objective functions but we are interested to see how close they come. Equally, we do not expect weighted sum methods to outperform multi-objective methods in terms of diversity but are interested to see how the diversity of the two respective populations compares. In order to reduce the effects of randomness, each method (7 weighted sum and 2 multi-objective) will be run ten times each on a set of three different problem instances with a stopping criterion of 250 generations of evolution. The problem instances were generated using the problem generator we have developed in collaboration with Vidus Ltd. (an @Road company). The test problems we used had 100 tasks, 10 resources and 6 skills, considered over a 3-day scheduling period, and correspond to a "small to medium sized" problem in practice.

These experiments were run using a master-slave approach to parallelise and thus speed up the GA runs. We used 26 2.8Ghz Pentium 4 machines (25 slaves and one master) and each of the GAs has a population size of 100 and was allowed the same number of schedule evaluations over 250 generations. Each run took around 1 (single-machine) CPU hour on this parallel architecture. Since we consider 9 solution approaches and 10 runs for each approach for 3 problem instances, a total of 270 runs were carried out.

Values for mutation and crossover rates will be taken from previous parameter tuning experiments for the weighted sum objective method with binary tournament selection and elitist replacement, these being a crossover rate of 25% and a mutation rate of 1%. The reader should note that these parameters have not been specially tuned for the multi-objective methods, particularly since such tuning is much more difficult when objective weights are not assumed.

The weighted sum methods will use the objective functions shown in Table 1 where SP is the sum of all the priority values of all scheduled tasks, SC is the total

schedule cost, and TT is the total travel time on the schedule. The values in Table 1 have been chosen to provide a trade off between objectives. Here we are considering a diverse range of weights to see if we can get close to a schedule which is good for a human scheduler without knowing what the human scheduler's ideal weighting scheme is, since the human scheduler is unlikely to know these weights in practise.

Table 1. Objective functions for weighted sum methods

Formula for Calculation	Name on Graphs
f = SP	SP
f = SP – (SC * 6)	SP_6SC
f = SP – (TT * 6)	SP_6TT
f = SP – (SC * 2) – (TT * 4)	SP_2SC_4TT
f = SP – (SC * 4) – (TT * 2)	SP_4SC_2TT
f = SP – (SC * 2) – (TT * 6)	SP_2SC_6TT
f = SP – (SC * 6) – (TT * 2)	SP_6SC_2TT

The genetic algorithm was run ten times with each weighted sum objective function shown in Table 1, as well as ten times with both NSGA-II and SPEA2 multi-objective methods. Each of these runs were assessed by each of the objective functions in Table 1 as well as the "Max Spread" [9] and "Morrison and De Jong" [16] diversity measures. Averages over the ten runs of each have been taken and plotted as a percentage of the best average found.

5.1 Experimental Results

Figure 1 shows the average fitness value of the best individual in the final population over thirty runs (three problem instances run ten times each). Each group of results along the x-axis represents the performance of all the different types of GA when assessed using fixed objective weights as given in table 1. For example, the first group (of 9 bars) in Figure 1 shows the performance of the multi-objective methods NSGA-II and SPEA2 as well as the single objective methods (with different objective weights) when assessed using the SP objective function.

Figure 1 shows that multi-objective algorithms find solutions that are within 2% of the best solution found by weighted sum objectives (when assessed by the weighted sum objectives) without knowing what the weights are and often find a solution within 1% of the best. This cannot be said of the weighted sum objective approaches which use the "wrong" weights whose performance is much less consistent. Figure 1 illustrates the possible effect of a poorly defined set of weights on resulting solution quality, for example if our actual global objective function (not known to the user) is SP_6SC and the user defines SP as the objective function then the solution they ended up with would be much worse in global terms than if they had used a multi-objective approach. The NSGA-II approach yielded results as good or slightly better than SPEA2 on average in all cases.

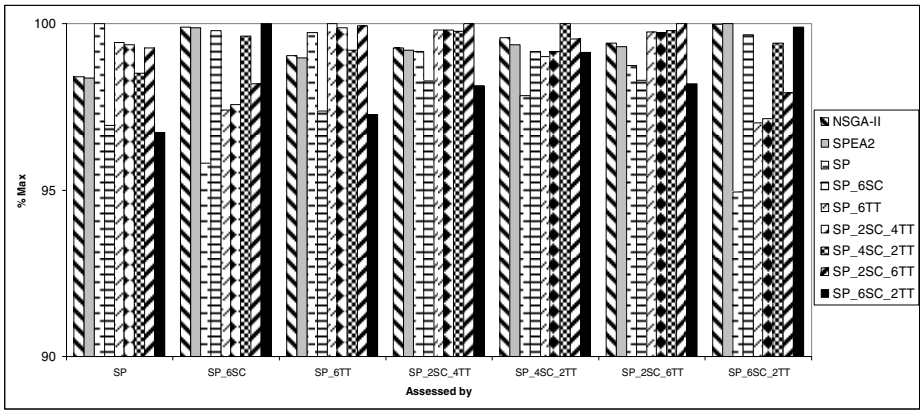

Fig. 1. The relative performance of multi-objective and single objective weighted sum genetic algorithms when assessed by different weighted sum objective functions

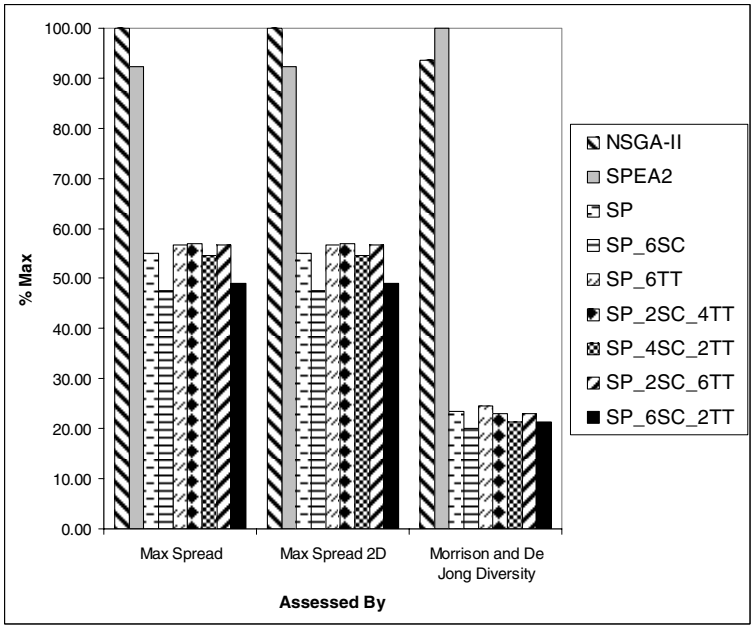

Fig. 2. The relative performance of multi-objective and single objective weighted sum genetic algorithms when assessed by diversity metrics

An interesting observation is that sometimes the "best" result is not found by the GA that is running the same objective function weights as is being used to assess the runs, an example of this is shown in Figure 1 when assessing the runs with the SP_6SC_2TT objective, here the GA that was run with the SP_6SC_2TT objective is outperformed by both NSGA-II and SPEA2 albeit by a small amount. In a case like

this we believe that there may be a mismatch between the schedule generation scheme‘s task-resource allocation heuristic and the objective weights chosen, but this flaw is not exposed by the NSGA-II and SPEA2 methods which have a diverse population.

Figure 2 shows the average values of the diversity measures for the different GA populations. Here NSGA-II and SPEA2 yield a much more diverse population than weighted sum approaches.

The results of figures 1 and 2 show that where it may not be possible for a user to define weights for each objective, still at least one of the population generated using a multi-objective approach is likely to be close to the users preference, even though this preference is not known in advance. A new problem arises in this case for the user, that of selecting a good solution from a population. However, this is a problem that the user is more familiar with and which is likely to yield better results than choosing objective weights in many cases. Ways in which the users ability to choose between schedules for the RCPSP such as those considered in Shackleford and Corne [17] and there approaches could be useful here.

6 Conclusions

In this paper multi-objective genetic algorithms have been shown to be an effective approach allowing the user to avoid having to define weights for a set of objectives in exchange for a small decrease in solution quality. This is useful as expressing their knowledge as a set of weights for a real world problem is usually difficult for human users who are not used to being asked to explicitly define the relative importance of different problem goals.

One problem with this approach is that it replaces the problem of defining a set of weights with the problem of selecting a solution from the Pareto optimal set which is an interesting research question in itself. However, analysing schedules and choosing a good one is a much more familiar activity for a human scheduler.

In future work it will be interesting to consider problems with a larger number of objectives, and methods which integrate the users’ ability to effectively select a solution into the solution process.

References

1. Pinedo, M., Chao, X.: Operations scheduling with applications in manufacturing and services. McGraw-Hill, New York (1999)
2. Brucker, P., Knust, S.: Resource-Constrained Project Scheduling and Timetabling. In: Burke, E. Erben, W. (eds.): Practice and Theory of Automated Timetabling III (PATAT). Springer-Verlag Berlin Hidelberg New York (2000) 277-293
3. Lova, A., Maroto, C., Thomas, P.: A multicriteria heuristic method to improve resource allocation in multiproject scheduling. Eur. J. Op. Res. Vol. 127. Elsevier (2000) 408-424
4. Holland, J. H.: Adaptation in Natural and Artificial Systems. University of Michigan Press. Michigan. (1975)
5. Hartmann, S.: A Competitive Genetic Algorithm for Resource-Constrained Project Schedul-ing. Naval Research Logistics. Vol. 45. Wiley (1998) 733-750.

6. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. Article to appear in the Eur. J. Op. Res. (2005)
7. Valls, V., Ballestin, F., Quintanilla, S.: A hybrid genetic algorithm for the RCPSP. Technical Report, Department of Statistics and Operations Research, University of Valencia.. (2005)
8. Schaffer J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proceedings of an International Conference on Genetic Algorithms and their Applications, (1985) 93-100
9. Deb, K.: Multi-objective Optimisation using Evolutionary Algorithms. John Wiley and Sons, New York.(2001)
10. Srinivas, N., Deb, K.: Multiobjective Optimisation Using Nondominated Sorting in Genetic Algorithms. J. Evolutionary Computation, Vol. 2 No. 3. (1994) 221-248
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multi-Objective Genetic Algorithm-NSGA-II. IEEE Trans. Evolutionary Computation, Volume 6. (2002) 849-858
12. Zitzler, E., Thiele, L.: An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland. (1998)
13. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland. (2001)
14. Corne, D. W., Knowles, J. D., Oates, M. J.: The pareto envelope-based selection algorithm for multiobjective optimisation. In M. S et al. (Ed.) Parallel Problem Solving from Nature. PPSN VI, Berlin (2000) 839-848
15. Kolisch, R., Hartmann, S.: Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In Weglarz, J. (ed): Project scheduling: Recent models, algorithms and applications. Kluwer, Amsterdam, the Netherlands (1999) 147-178
16. Landa Silva, J.D., Burke, E.K.: Using Diversity to Guide the Search in Multi-Objective Optimization. In: Coello Coello C.A., Lamont G.B. (eds.): Applications of Multi-Objective Evolutionary Algorithms, Advances in Natural Computation, Vol. 1, World Scientific. (2004) 727-751
17. Shackelford, M., Corne, D.: Collaborative Evolutionary Multi-Project Resource Scheduling. In Proceedings of the 2001 Congress on Evolutionary Computation, IEEE Press (2001) 1131-1138
18. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research. Vol. 153. Elsevier (2004) 3-27