A Multi-objective Tabu Search Algorithm for Constrained Optimisation Problems

Daniel Jaeggi, Geoff Parks, Timoleon Kipouros, and John Clarkson

Engineering Design Centre, Department of Engineering,
University of Cambridge, Trumpington Street,
Cambridge CB2 1PZ, United Kingdom

Abstract. Real-world engineering optimisation problems are typically multi-objective and highly constrained, and constraints may be both costly to evaluate and binary in nature. In addition, objective functions may be computationally expensive and, in the commercial design cycle, there is a premium placed on rapid initial progress in the optimisation run. In these circumstances, evolutionary algorithms may not be the best choice; we have developed a multi-objective Tabu Search algorithm, designed to perform well under these conditions. Here we present the algorithm along with the constraint handling approach, and test it on a number of benchmark constrained test problems. In addition, we perform a parametric study on a variety of unconstrained test problems in order to determine the optimal parameter settings. Our algorithm performs well compared to a leading multi-objective Genetic Algorithm, and we find that its performance is robust to parameter settings.

1 Introduction

Real-world optimisation problems have a number of characteristics which must be taken into account when developing optimisation algorithms. Real-world problems are typically multi-objective; trade-offs between risk and reward, and cost and benefit exist at a fundamental level throughout the natural world and are a deep-seated part of human consciousness. These trade-offs carry over directly to the business world, and thus into any form of design activity. Any optimisation method which is to have any serious benefit to the design process must be able to handle multiple objectives.

Real-world problems also tend to be highly constrained. The nature of these constraints and their effect on the optimisation landscape varies from problem to problem. However, optimisation problems in a number of fields have constraints with similar characteristics, and this is discussed further in Section 1.1 below. The optimisation landscape – regions of feasible, highly constrained design space and the variations of objective function values within that space – is strongly influenced by the parameterisation scheme, for any one given problem. Good parameterisation schemes for aerodynamic shape optimisation problems – the particular focus of our work – as shown by Harvey [1], Kellar [2] and Gaiddon

C. A. Coello Coello et al. (Eds.): EMO 2005, LNCS 3410, pp. 490–504, 2005.

et al. [3], tend to produce optimisation landscapes that are highly constrained, have many variables, and many local minima. Thus, the optimisation algorithm must be chosen to perform well in these circumstances [4].

These characteristics quickly rule out the use of traditional gradient-based optimisation methods: notwithstanding their requirement of gradient information (which may be difficult, expensive, or impossible to obtain), these algorithms perform poorly in problems which are highly constrained and contain local minima. Harvey [1] tested a number of meta-heuristic methods on a representative aerodynamic design optimisation problem and found Tabu Search (TS) to be superior to the Genetic Algorithm (GA) and Simulated Annealing (SA) methods.

Numerous of multi-objective GAs exist [5]. Similarly, multi-objective SA methods have been developed [6]. However, despite its popularity in single-objective optimisation problems, very few attempts have been made at developing a multi-objective version of TS. Jones [7] reviewed the literature on multi-objective meta-heuristics and found only 6% of 124 papers concerned with TS.

Given that it may well perform better than a GA or SA method (assuming Harvey's results carry over into multi-objective optimisation) on aerodynamic design optimisation problems, and there is a strong real-world requirement to perform multi-objective optimisation, there appears to be both a need and an opportunity to develop a new multi-objective TS algorithm.

1.1 Constraint Handling

It is important that the constraint handling method of an optimisation algorithm is able to deal with constraints which are binary, as happens in a number of real-world engineering problems. The constraint handling in many multi-objective evolutionary algorithms requires the ability to assign some kind of constraint violation distance to points in design space which violate constraints, and points are then ranked accordingly [5]. In the presence of binary constraints, such an approach cannot be used.

Such constraints occur typically in shape optimisation problems, especially when the cost function is evaluated using a finite difference type of method (including finite element and finite volume methods) which solves a system of equations over a finite mesh. The constraints for these problems arise from three sources, amongst others:

1. *Geometric considerations.* The parameterisation scheme may give rise to shapes which are physically impossible (*i.e.* negative volumes). Conceptually, a distance measure in design space may be formulated by considering an offset vector $\Delta\bar{x}$ which can be added to the design vector $\bar{x}$ to make the design feasible; in practice this may be too costly due to the interdependence between design variables.
2. *Mesh considerations.* Given a valid geometry, it may be impossible to fit a mesh that satisfies certain criteria relevant to the numerical solution of the problem (*i.e.* skewed cells). Again, finding an offset vector $\Delta\bar{x}$ is conceptually possible, but is in practice even more costly than finding a $\Delta\bar{x}$ that satisfies just the geometric considerations.

3. *Numerical solution considerations.* Given a valid mesh and geometry, it may still be impossible to reach a numerical solution of the system of equations for that geometry (*i.e.* lack of convergence in a finite difference method). Yet again, finding a $\Delta\bar{x}$ that makes the solution converge is possible but totally unrealistic, given the massive computational cost of a solution.

The extent to which such constraint violations are encountered is strongly dependent on the parameterisation scheme used. For any problem, it is probably possible to employ a parameterisation scheme that gives a continuous design space and constraint violations for which $\Delta\bar{x}$ may be easily calculated. However, it is the authors' belief that the parameterisation scheme should be chosen first and then the optimisation algorithm, not the other way round. Of course, there is a trade-off: a parameterisation scheme that produces a design space that is almost impossible to search is of virtually no use. Yet we can clearly enhance our choice of parameterisation scheme by easing the restrictions placed on it by the optimiser's constraint handling method.

1.2 Existing Multi-objective Tabu Search Algorithms

There are two approaches to solving a multi-objective optimisation problem. The first reduces the multiple objectives to a single objective by generating a composite objective function, usually from a weighted sum of the objectives. This composite objective function can be optimised using existing single-objective optimisers. However, the weights must be pre-set, and the solution to this problem will be a single vector of design variables rather than the entire Pareto-optimal (PO) set. This can have undesirable consequences: setting the weights implicitly introduces the designer's preconceptions about the relative trade-off between objectives. Real-world problems can produce surprising PO sets which may profoundly affect design decisions, and the potential to generate novel designs is a key benefit of optimisation [6].

The second approach to solving the multi-objective problem is to search directly for the entire PO set. This can be achieved in a number of ways and requires modification to existing single-objective algorithms.

The authors know of only two attempts to produce a multi-objective TS algorithm which finds multiple PO solutions in a single run. Hansen's algorithm [8] is an extension of the composite objective approach: his algorithm performs a number of composite objective Tabu searches in parallel. Each search has a different and dynamically updated set of weights, and in this way the search can be driven to explore the entire Pareto front. This algorithm, although a good implementation of TS, suffers the problems common to all weighted-sum approaches: for problems with concave Pareto fronts, there may be regions of the front that are not defined by a combination of weights and conversely certain combinations of weights represent two points on the front. Thus, this algorithm may not adequately locate the entire PO set.

Baykasoglu *et al.* [9] developed a TS algorithm combining a downhill local search with an *intensification memory* (IM) to store non-dominated points that

were not selected in the search. When the search fails to find a downhill move, a point from the IM is selected instead. When the IM is empty and all search paths exhausted, the algorithm stops. This cannot be considered a true TS algorithm: in restricting the search to only downhill moves its originators reject one of the basic tenets of TS, that "a bad strategic choice can yield more information than a good random choice" [10]. Also, the lack of any diversification strategy renders the algorithm incomplete and merely an elaborate local search algorithm.

The other TS algorithms reviewed by Jones [7] either use a composite objective function or are little more than local search algorithms similar to the algorithm of Baykasoglu *et al.*

2 Multi-objective Tabu Search Adaptation

The single-objective TS implementation of Connor and Tilley [11] is used as a starting point for our multi-objective variant. This uses a Hooke and Jeeves (H&J) local search algorithm (designed for continuous optimisation problems) [12] coupled with short, medium and long term memories to implement search intensification and diversification as prescribed by Glover and Laguna [10].

TS operates in a sequential, iterative manner: the search starts at a given point and the algorithm selects a new point in the search space to be the next current point. The basic search pattern is the H&J search.

Recently visited points are stored in the *short term memory* (STM) and are tabu – the search is not allowed to revisit these points. Optimal or near-optimal points are stored in the *medium term memory* (MTM) and are used for intensification, focusing the search on areas of the search space with good objective function values. The *long term memory* (LTM) records the regions of the search space which have been explored, and is used on diversification, directing the search to regions which are under-explored. This is achieved by dividing each control variable into a certain number of regions and counting the number of solutions evaluated in those regions. A local iteration counter *i_local* is used and reset upon a successful addition to the MTM. When *i_local* reaches user-specified values, the algorithm will diversify or intensify the search, or reduce the search step size and restart the search from the best solution found.

Thus, TS combines an exhaustive, systematic local search with a stochastic element and an intelligent coverage of the entire search space. Our multi-objective TS implementation of [11] is modified in the following areas: search point comparison; the H&J move; optimal point archiving and the MTM; search intensification and restart strategy. These modifications are described briefly below, along with some further improvements.

2.1 Search Point Comparison

In a single-objective optimisation problem, points may be compared using the operators $==$, $>$ and $<$ acting on the objective function values for those points. Similarly, points in a multi-objective problem can be compared in the same way

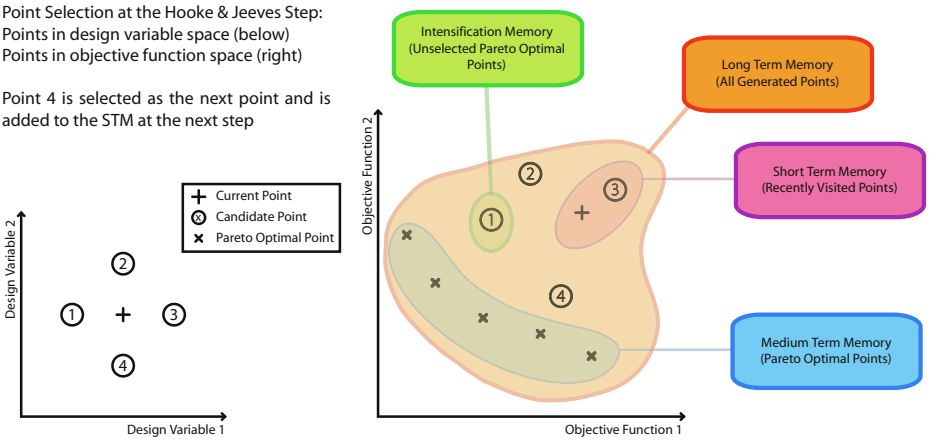


Fig. 1. Point Selection for the Hooke & Jeeves move and Tabu Search Memories

(thus preserving the logic of the single-objective algorithm) by using the concepts of Pareto equivalence (==) and dominance (> and <).

2.2 The Hooke and Jeeves Move

At each iteration, a H&J move is made. $2n\_var$ new points are generated by incrementing and decrementing each design variable by a given step around the current point. The objective functions for each new point are evaluated and, as long as the point is neither tabu (*i.e.* not a member of the STM) nor violates any constraints, it is considered as a candidate for the next point in the search.

In the single-objective TS algorithm, these candidates are sorted and the point with the lowest objective is chosen as the next point. A similar logic can be applied to the multi-objective case: however, the possibility of multiple points being Pareto equivalent (PE) and optimal must be allowed for.

This is achieved by classifying each candidate point according to its domination or Pareto equivalence to the current point. If there is a single dominating point, it is automatically accepted as the next point. If there are multiple dominating points, the dominated points within that group are removed and one is selected at random from those remaining. The other points become candidates for intensification (discussed below). If there are no dominating points, the same procedure is applied to those candidate points which are PE to the current point. If there are no PE points, a dominated point is selected in the same fashion. Thus, our strategy accepts both downhill and uphill moves – the next point is simply the "best" point (or one of the PE best points) selected from the candidate solutions. This logic is shown in Fig. 1 for clarity.

In addition, a *pattern move* strategy is implemented in the same way as Connor and Tilley [11]. Before every second H&J move, the previous move is repeated. This new point is compared to the current point, and, if it dominates

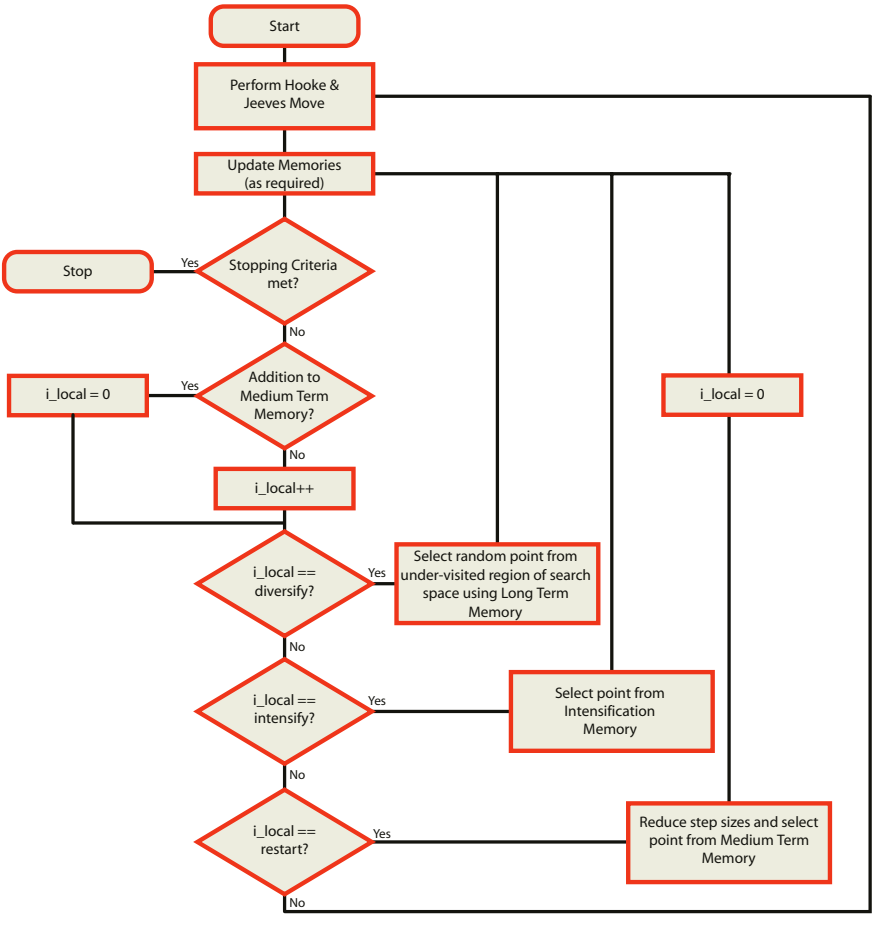


Fig. 2. Flow Diagram of the Multi-objective Tabu Search Algorithm

it, is accepted as the next point; if not, the standard H&J move is made. In this way, the search may be accelerated along known downhill directions.

2.3 Optimal Point Archiving and the Medium Term Memory

In Connor's single-objective TS [11], the MTM is a bounded, sorted list of near-optimal solutions. As the concept of a single optimal point does not exist in multi-objective optimisation (see Section 1.2), we replace the MTM in our multi-objective TS variant by an unbounded set of non-dominated solutions produced by the search. As new points are evaluated, they become candidates for addition to this set. Thus, the MTM represents the PO set for the problem at that stage in the search.

2.4 Intensification and Restart Strategy

The original single-objective TS produced intensification points by using the MTM to generate points in the neighbourhood of good solutions. Although the replacement of the MTM by a PO set of solutions allows us to use a variant of this strategy, a feature of multi-objective optimisation suggests an alternative strategy, similar to that used by Baykasoglu *et al.* [9].

A multi-objective H&J iteration may produce multiple PO points (see Fig. 1). As only one point may be selected as the next point, it seems wasteful to discard the other points. Therefore, we incorporate an intensification memory into our algorithm. This is a set of PE points; at each H&J step, points which dominate the current solution, but are not selected as the next point (of which there can be only one), are considered as candidates for addition to the set. At search intensification, a point is chosen randomly from the IM. The IM is continuously updated and points which become dominated by the addition of a new point are removed. Thus, the IM should always contain points which are on, or near to, the current PO front (stored in the MTM).

The single-objective TS restart strategy returns the search to the current best point in the MTM. As the MTM is now a set of PO points, we simply select one point at random from the set. More intelligent restart strategies are possible [6] and are under investigation. Fig. 1 gives an overview of the various memories used, and Fig. 2 a flow diagram for our multi-objective TS implementation.

2.5 Constraint Handling

We employ a very simple constraint handling strategy: any point which violates any constraint is deemed to be tabu and the search is not allowed to visit that point. Thus, accepted solutions are limited to feasible space. On search diversification, we allow the algorithm to loop until a feasible point has been found. Depending on the problem, it would also be possible to introduce penalty functions to handle certain constraints.

2.6 Improving Local Search Efficiency and Parallelisation Strategy

The H&J local search strategy requires roughly $2n\_var$ solution evaluations (allowing for points that are tabu or violate constraints) at each step, where n_var is the number of design variables. A real-world problem may contain a large number of variables (the shape optimisation of a Boeing 747 wing required 90 variables [13]) and this strategy could become prohibitively expensive. One solution to this is to incorporate an element of random sampling in the H&J step.

We generate the $2n\_var$ new points, remove those that are tabu, and only evaluate $n\_sample \leq 2n\_var$ points from those that remain, selecting randomly to avoid introducing any directional bias. If one of these points dominates the current point, it is automatically accepted as the next point. If more than one point dominates the current point, a non-dominated point from these is randomly selected. If no points dominate the current point, a further n_sample points are sampled and the comparison is repeated. If all the feasible, non-tabu points have

Table 1. Test Functions

Function Name	Number of Variables	Number of Objectives	Constraint Types
SCH	1	2	None
FON	3	2	None
POL	2	2	None
KUR	3	2	None
ZDT1	30	2	None
ZDT2	30	2	None
ZDT3	30	2	None
ZDT4	10	2	None
ZDT6	10	2	None
CONSTR	2	2	2 Inequality
SRN	2	2	2 Inequality
TNK	2	2	2 Inequality
WATER	3	5	7 Inequality

been sampled without finding a point that dominates the current solution, the standard selection procedure is employed.

Any optimisation procedure that forms part of a real-world design cycle must be able to complete in a reasonable time-frame. Parallel processing offers a large potential speed-up; any serious optimisation algorithm should be designed with this in mind. Our multi-objective Tabu Search algorithm is parallelised by means of functional decomposition. At each H&J move, the required objective function evaluations are computed in parallel.

3 Tabu Search Parameter Investigation

The performance of this algorithm has already been tested on nine standard unconstrained test problems and compared to the performance of NSGA-II [14]. Over these nine problems, the algorithm performed comparably with NSGA-II. In that initial study, no attempt was made to find optimal TS parameter settings; they were set to reasonable values, based on experience. Here, we conduct a more systematic parameter setting investigation. The parameters that may be set in the algorithm are shown in Table 2. The parameter settings used in the benchmarking in [14] are also given.

In the studies that follow, the same test conditions as in [14] were used. Parameters not being varied were kept fixed at the values given in Table 2. Performance was assessed using the convergence metric Υ, described by Deb *et al.* [15]; the mean and standard deviation for the results of 45 runs were calculated, and the same set of random number generator seeds used. Each run was stopped and the results calculated after 25000 function evaluations.

Table 2. Tabu Search parameters

Parameter	Initial Value	Description
diversify	10	Diversify search when $i\_local == diversify$
intensify	20	Intensify search when $i\_local == intensify$
reduce	50	Reduce step sizes and restart when $i\_local == reduce$
n_stm	20	STM size – the last *n_stm* visited points are tabu
n_regions	2	In the LTM each variable is divided into *n_regions* regions
SS	8%	Initial H&J step size as percentage of variable range
SSRF	0.5	Factor by which step sizes are reduced on restart
n_sample	6	Number of points randomly sampled at each H&J move

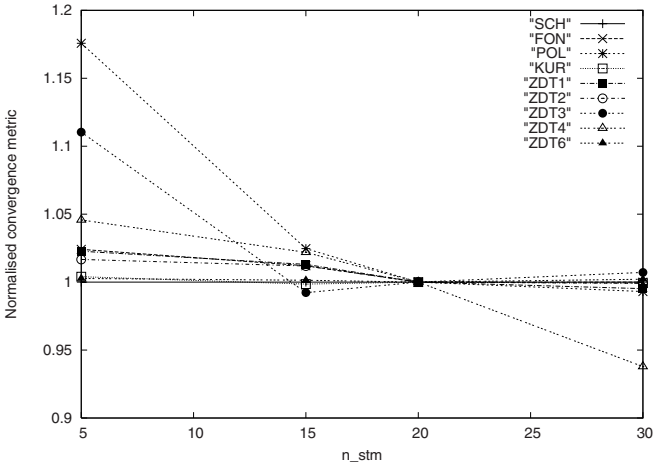


Fig. 3. Normalised convergence metric $\hat{\Upsilon}$ vs *n_stm* over 9 test problems

3.1 Variation in Short Term Memory Size

One of the distinguishing features of TS algorithms in general is the use of a short term memory to define points that are tabu and may not be revisited. This gives the search algorithm a means by which it can climb out of local minima. We might expect that the size of the STM may affect the performance of the algorithm: a STM of zero size would reduce the algorithm to a mere local search algorithm coupled with a global random search; a STM of infinite size would prevent the search from refining solutions in known good regions of the search space. There are also algorithm run time considerations: the computational cost of the algorithm increases with STM size, but for most real-world problems this cost is negligible compared to the cost of function evaluations.

We consider variations in the STM size in the range $5 \leq n\_stm \leq 30$ for the nine test problems used in [14], keeping all other parameter values constant as given in Table 2. The results are shown in Fig. 3; Υ is normalised with respect

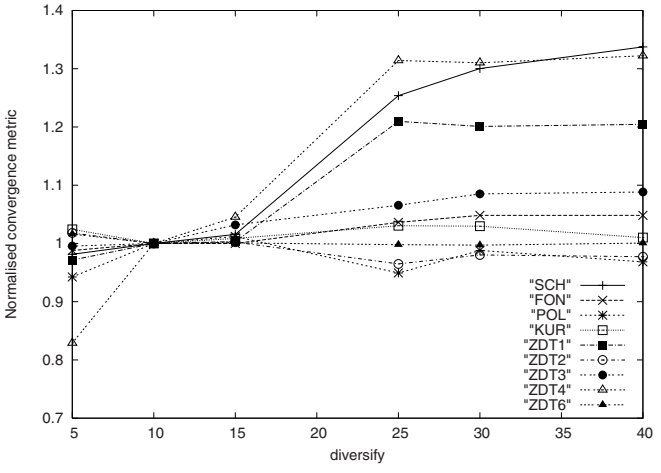


Fig. 4. Normalised convergence metric $\hat{\Upsilon}$ vs *diversify* over 9 test problems

to its value for $n\_stm = 20$, the value used in [14], and is plotted against n_stm. Performance is improved by increasing n_stm from 5 up to around 15 on most test problems. Increases beyond that improve performance but only marginally, and there does not appear to be much benefit in this. On problems with low numbers of design variables, there is hardly any variation in performance; this is most likely due to the lower potential for "cycling" (the search repeating a recent search path) as there are fewer paths that can be taken. The exception is Poloni's problem which shows quite a large improvement in performance between $n\_stm = 5$ and $n\_stm = 15$ despite having only two design variables. Problem ZDT3 also displays a large improvement in performance over this range of n_stm; on this problem, it appears that a larger STM is required to prevent cycling.

The other test problem that exhibits sensitivity to changes in n_stm is problem ZDT4; this shows continually improving performance with increases in n_stm. However, because the presented results have been normalised to show relative performance the incredibly poor absolute performance of our algorithm on ZDT4 has been masked. This is commented on in [14].

3.2 Variations in *Intensify* and *Diversify*

The parameters *intensify* and *diversify* control the balance between a local search of the design space and a global one. Therefore, they are critical in governing performance: certain problems, particularly multi-modal ones, will benefit strongly from a strategy which favours diversification; other problems, such as those with clearly defined local regions containing many near PO points, will be better searched by a strategy favouring intensification.

Fig. 4 shows the effect of varying $diversify$ on the normalised convergence metric $\hat{\Upsilon}$; as in Section 3.1, Υ is normalised against its value for $diversify = 10$, used in the original study [14]. There appear to be two trends: for one group of

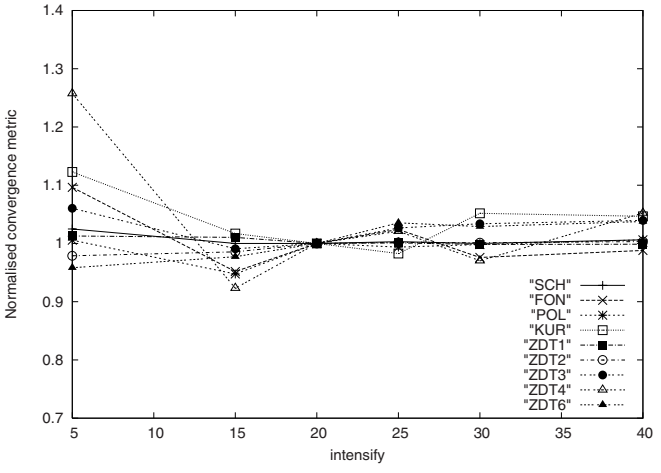


Fig. 5. Normalised convergence metric $\hat{\Upsilon}$ vs *intensify* over 9 test problems

problems, the point at which diversification takes place has little effect on overall performance; another group definitely favours early diversification. In general, early diversification brings performance benefits for these problems.

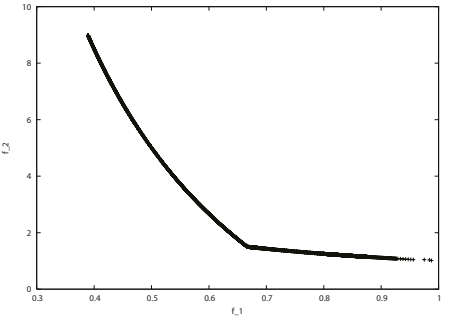
It also is worth noting two points. First, $intensify$ was fixed at 20: thus, over the range of values for $diversify$ used in this study, the order in which intensification and diversification takes place changes. Some of the performance variation is attributable to this. In particular, on problem ZDT4, which contains a large number of false Pareto fronts, intensifying before diversifying traps the optimiser in these false fronts and hinders the location of the true Pareto front.

Second, early diversification tends to increase the total number of diversification moves performed during an optimisation run. This behaviour is beneficial on problems such as SCH; good performance on this problem depends on how fast the region of design space in which the Pareto front is located is found. Thus, more random behaviour in the search speeds its discovery; once this region is found, the local search component in TS effectively finds the rest of the front.

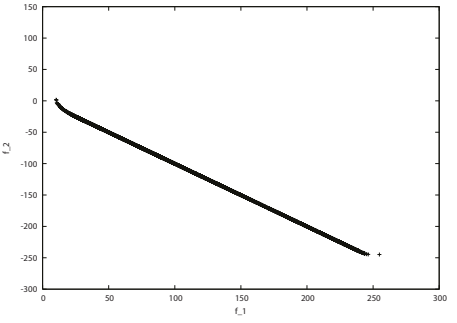
Similarly, Fig. 5 shows the effect on $\hat{\Upsilon}$ of varying $intensify$; Υ is normalised against its value for $intensify = 20$.

For the majority of problems, the absolute value of $intensify$ appears relatively unimportant; of more importance is whether $intensify$ is less or greater than $diversify$. The results suggest that for good performance on these test functions in general, diversification must occur first and its value primarily governs performance. However, the results also show that there is some benefit in reducing the gap between diversification and intensification; a value of $intensify = 15$ gives, on average, better performance. This would prevent the algorithm from needlessly searching poor areas of the design space.

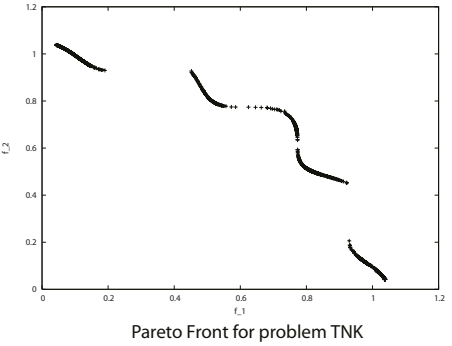
These results raise a concern about the use of test functions in determining algorithm performance on real-world problems. Kipouros *et al.* [16] used a variant of this TS algorithm to perform a multi-objective optimisation of a gas-



Pareto Front for problem CONSTR



Pareto Front for problem SRN



Pareto Front for problem TNK

	CONSTR	SRN	TNK
Mean	0.01523	0.06721	0.48625
Standard Deviation	0.00081	0.00378	0.29218

Mean & Standard Deviation of the convergence metric for the three test problems. Convergence was measured to an ideal Pareto front consisting of 500 points.

Fig. 6. Pareto fronts for constrained test problems CONSTR, SRN & TNK and convergence metric results

turbine compressor blade. For that particular application, intensification was found to be particularly beneficial to overall performance; the optimiser was able to make steady progress (indicated by rate of addition to the MTM) by performing many intensification steps. This suggests that the problem contains regions with many locally PO points and TS is able to effectively search all these through its intensification strategy. In contrast, this characteristic does not appear to be present in this set of test functions. There appears to be an urgent need to devise test problems that accurately reflect characteristics of real-world problems.

For both these sets of results, error bars based on the standard deviation of $\hat{\Upsilon}$ were not plotted, for reasons of clarity. Although there was some variation in the standard deviation with parameter setting, for the most part, these variations were small and the values remained close to the nominal values reported in [14]. There are two main exceptions to this: with $diversify = 5$ seven problems showed an increase in the standard deviation; the standard deviations for ZDT4 varied greatly, which is probably due to the algorithm's poor performance on this problem.

4 Constrained Test Problems

Deb *et al.* [15] also tested NSGA-II on four constrained test problems. Although no quantitative performance results were published, plots showed that NSGA-II was capable of finding a good spread of results along the Pareto front for each. We have tested our algorithm on those four constrained problems: the results for the problems CONSTR, SRN and TNK are shown in Fig. 6; results for problem WATER are presented as the range of values found for all five objectives and are given in Table 3. We used the lower limit of 20000 function evaluations prescribed by Deb *et al.* The parameter settings used were the same as given in Table 2, with the exception that $intensify$ was reduced to 15 as a result of the parameter study presented above.

Table 3. Lower and upper bounds of objective function values on the Pareto front for problem WATER

	f_1	f_2	f_3	f_4	f_5
Multi-objective TS	0.804-0.918	0.022-0.857	0.104-0.962	0.056-1.320	0.129-3.121
NSGA-II	0.798-0.920	0.027-0.900	0.095-0.951	0.031-1.110	0.001-3.124

On problem CONSTR, coverage of the Pareto front is excellent, except in a small region near the tail where f_1 approaches 1.0. The range of solutions found is comparable to NSGA-II and far superior to Ray-Tai-Seow's algorithm presented in [15]. Similar performance is achieved on problem SRN – convergence to and coverage of the Pareto front are good and comparable to NSGA-II.

Problem TNK is slightly harder – the Pareto front is discontinuous and, as shown by Deb *et al.*, some algorithms have difficulty finding the entire central continuous region. Although the spread of solutions that our algorithm finds in this region is slightly worse than on the rest of the Pareto front, it succeeds in locating the continuous region correctly.

Problem WATER is a five-objective problem; due to the difficulty of visualising the Pareto front, figures are presented in Table 3 for the minimum and maximum values of the objective functions found on the Pareto front. Deb *et al.* presented similar figures for NSGA-II on this problem. Of the 10 minimum/maximum values, NSGA-II finds better values on 7, although the differences in most cases are small.

5 Conclusions

In this paper, we have presented a multi-objective TS algorithm with features that make it particularly attractive to real-world optimisation problems, in particular with regard to its constraint handling. In previous work, we benchmarked

this algorithm against NSGA-II on a number of test functions and found that it performed comparably. Here, we performed a study on the effect of varying the TS parameters on the algorithm performance on the same nine test functions.

As regards the STM size, the results suggest that a value of $n\_stm \geq 15$ gives good performance on a range of test functions. Increasing n_stm beyond 25 does not appear to give much performance benefit on the functions tested. Although the computational cost of the algorithm increases with n_stm, this cost is usually negligible for real-world problems where function evaluations are expensive.

Results for varying $intensify$ and $diversify$ show that algorithm performance is, in general, dominated by diversification; performance is improved by increasing the diversification element, and intensification has relatively little effect. This is slightly at odds with experience of using this algorithm on aerodynamic shape optimisation problems, where intensification is a more effective means of advancing the search. This raises a concern about the use of test functions to derive algorithm performance information for use in real-world problems.

Over the majority of test functions, performance is reasonably independent of the TS parameter settings; this suggests that the power of TS comes from its fundamental elements – the combination of a local search algorithm with a variety of search memories – rather than particular, carefully chosen parameter settings. This should ease its application to new problems.

Finally, our TS algorithm was tested on four constrained test problems. On all the problems, the algorithm was able to find a good spread of solutions along the Pareto front, despite our strict constraint handling approach. We believe our multi-objective TS algorithm is well suited to use on real-world optimisation problems where constraints can be binary in nature and such an approach is required to allow the optimiser to run. Indeed the only way to show this is to actually apply it to real-world problems (as we have done in a companion paper [17]); test problems that exist in the literature do not share these characteristics, and it is hard to draw meaningful conclusions from tests using these problems.

However, future work should include a more rigorous performance comparison with other leading MO optimisation algorithms. The performance metrics used in this study are not optimal [18] but were chosen to allow comparison with previously published data [15]. Finally, determining optimal settings for this algorithm requires further more detailed work.

Acknowledgements

This research is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant number GR/R64100/01. The authors would also like to thank Prof. Bill Dawes for his support and encouragement.

References

1. Harvey, S.: The Design Optimisation of Turbomachinery Blade Rows. PhD thesis, Cambridge University Engineering Department (2002)

2. Kellar, W.: Geometry Modelling in Compuational Fluid Dynamics and Design Optimisation. PhD thesis, Cambridge University Engineering Department (2002)
3. Gaiddon, A., Greard, J., Pagan, D., Knight, D.: Automated optimization of supersonic missile performance taking into account design uncertainties. Technical Report AIAA-2003-3879, AIAA (2003)
4. Duvigneau, R., Visonneau, M.: Hybrid genetic algorithms and artificial neural networks for compex design optimization in CFD. International Journal for Numerical Methods in Fluids **44** (2004) 1257–1278
5. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Ltd., Winchester, UK (2001)
6. Suppapitnarm, A., Seffen, K., Parks, G., Clarkson, J.: A simulated annealing algorithm for multiobjective optimization. Engineering Optimization **33** (2000) 59–85
7. Jones, D., Mirrazavi, S., Tamiz, M.: Multi-objective meta-heuristics: An overview of the current state-of-the-art. European Journal of Operational Research **137** (2002) 1–9
8. Hansen, M.: Tabu search for multiobjective optimization: MOTS. In: MCDM, Cape Town, South Africa. (1997)
9. Baykasoglu, A., Owen, S., Gindy, N.: A taboo search based approach to find the pareto optimal set in multiple objective optimization. Engineering Optimization **31** (1999) 731–748
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston, MA (1997)
11. Connor, A., Tilley, D.: A tabu search method for the optimisation of fluid power circuits. IMechE Journal of Systems and Control **212** (1998) 373–381
12. Hooke, R., Jeeves, T.: Direct search solution of numerical and statistical problems. Journal of the ACM **8** (1961) 212–229
13. Jameson, A.: A perspective on computational algorithms for aerodynamic analysis and design. Progress in Aerospace Sciences **37** (2001) 197–243
14. Jaeggi, D., Asselin-Miller, C., Parks, G., Kipouros, T., Bell, T., Clarkson, P.: Multi-objective parallel tabu search. In Yao, X., Burke, E., Lozano, J.A., Smith, J., Merelo-Guervos, J., Bullinaria, J., Rowe, J., Tino, P., Kaban, A., Schwefel, H.P., eds.: Parallel Problem Solving from Nature – PPSN VIII. Lecture Notes in Computer Science, Vol. 3242, Springer-Verlag, Berlin (2004) 732–741
15. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
16. Kipouros, T., Parks, G., Savill, A., Jaeggi, D.: Multi-objective aerodynamic design optimisation. In: ERCOFTAC Design Optimisation: Methods and Applications Conference Proceedings, On CDRom. Paper ERCODO2004_239. (2004)
17. Kipouros, T., Jaeggi, D., Dawes, W., Parks, G., Savill, A.: Multi-objective optimisation of turbomachinery blades using tabu search. In: 3rd Int. Conf. Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science. (2005)
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7** (2003) 117–132