

Evolutionary Multiobjective Optimisation Through Spatially-Structured Non-Dominated Sorting: A Preliminary Study

Grant Dick

¹Department of Information Science,
University of Otago,
Dunedin, NEW ZEALAND
Email: gdick@infoscience.otago.ac.nz

Presented at SIRC 2006 - The 18th Annual Colloquium of the Spatial Information Research Centre
University of Otago, Dunedin, New Zealand
November 6th-7th 2006

ABSTRACT

Multiobjective evolutionary algorithms (MOEAs) are useful tools capable of searching problems that contain several conflicting criteria. Although MOEAs have been shown to be capable of finding a wide spread of Pareto-optimal solutions for a given problem, they are still hindered by the requirement for significant computation. This paper investigates a new MOEA that incorporates spatial structure into the population. The introduction of space into the algorithm alters the behaviour of the algorithm so that computational complexity increases linearly with population size. In addition, the paper suggests paths that could be taken to improve the algorithm's ability to successfully converge upon the global Pareto-optimal front of a given problem.

Keywords and phrases: multiobjective optimisation, evolutionary algorithms, non-dominated sorting, population structure, computational complexity

1 Introduction

Evolutionary algorithms (EAs) are heuristic search methods inspired by the concepts of Mendelian genetics and Darwinian selection. With suitable modification they are able to effectively search problems that consist of several, often conflicting, objectives (Deb 2001). However, this ability to search multiobjective problems typically comes at the expense of high computational complexity. Although some inroads into reducing complexity have been made, a fast multiobjective evolutionary algorithms (MOEAs) will still have an average-case complexity of $O(MN^2)$ (where N is the population size and M is the number of objectives). Obviously this quadratic growth in complexity limits the applicability of these methods to problems that require large population sizes.

A primary contributor to the complexity of MOEAs is the need for the population to maintain sufficient diversity of solutions. One way to encourage a population to maintain diversity is to impose a spatial structure on the population (Dick & Whigham 2005). The population structure restricts mating within the population and promotes local genetic divergence. This paper investigates the use of population structure within an MOEA to reduce the number of diversity-preserving computations required for a given generation. The result of incorporating spatial structure within the population is an algorithm capable of executing with $O(MN)$ complexity.

The remainder of this paper is structured as follows; §2 introduces the concepts relevant to exploring multi-objective problems with evolutionary algorithms, §3 introduces the proposed *TorusNDS* algorithm which is then compared with an existing MOEA in §4. Finally, a discussion of the findings of this paper, and a suggested path for future work, is presented in §5.

2 Evolutionary Multiobjective Optimisation

Many real-world problems often have many components that often conflict each other. For example, in engineering problems, there is often a trade-off between the cost of manufacturing an item and its mean time to failure. Instead of a single optimal solution, these *multiobjective* problems possess a set of optimal solutions along a *Pareto-optimal* front.

2.1 Non-Dominated Sorting – NSGA-II

A recent development in the field of MOEAs is the Non-Dominated Sorting Genetic Algorithm II (*NSGA-II*) (Deb, Agrawal, Pratap & Meyarivan 2002). This method relies on two principles – *dominance* and diversity preservation through *crowding*. *Dominance* is defined as the case in which an individual a is better than individual b in at least one objective, and *no worse* than b on all remaining objectives. This dominance relationship can be used to sort a population of solution, and an individual’s fitness is based upon its rank within a sorted population. Crowding comes into play in the event that two individuals have the same rank and rewards the individual that occupies the individual that resides in the least occupied region of the Pareto front.

3 Proposed Algorithm

The *NSGA-II* was developed in response to a number of criticisms to previous MOEAs. While a significant improvement over these existing MOEAs, *NSGA-II* is still bound by quadratic computational complexity. The reason for this complexity is that, in order to maintain suitable diversity, each individual must be compared with the entire population. The methods that *NSGA-II* uses to maintain diversity are not the only possible solutions. Another mechanism that helps to promote diversity is to use a spatially-structured population. This method reduces the connectivity of the population by placing each individual at a specific location in space and limiting its interactions (selection, mating) to *demes* of geographically “close” individuals. The proposed method, referred to here as *TorusNDS*, uses a torus as its population structure. The actual algorithm for this method is shown in Algorithm 1. *TorusNDS* is essentially an asynchronous spatially-structured evolutionary algorithm (SSEA) using the “Uniform Choice” update policy (Tomassini 2005). The primary difference between *TorusNDS* and a typical SSEA is in the replacement strategy; a typical SSEA will use an elitist “replace if better” approach that relies solely on fitness. *TorusNDS*, however, uses operators from the *NSGA-II* method to determine the individuals that will be replaced at each time step.

```
input : A population of individuals of size  $N$   
output: The same population with reproduced individuals  
1 population ← generateInitialPopulation;  
2 while not done do  
3    $\{r, c\}$  ← random coordinates in population;  
4   deme ← constructDeme ( $r, c$ );  
5    $m$  ← population[ $r, c$ ];  
6    $f$  ← random element from deme;  
7    $\{d, s\}$  ← reproduce ( $m, f$ );  
8   pool ← deme  $\cup$   $\{m, f\}$ ;  
9   fastNonDominatedSort (pool);  
10  calculateCrowdedDistance (pool);  
11   $\{die_1, die_2\}$  ← last 2 elements of pool;  
12  replace ( $die_1, d$ );  
13  replace ( $die_2, s$ );  
14 end  
15 return population;
```

Algorithm 1: The proposed *TorusNDS* algorithm

The method presented here is not the first attempt to incorporate spatial population structure into an MOEA. Previous studies used a population structure to implement an environmental gradient that was used to weight individual objectives of the overall problem (Murata, Ishibuchi & Gen 2000, Kirley 2001). Environmental gradients will not be covered in this paper and will be explored at a later date.

3.1 Complexity

The *TorusNDS* method builds upon the operators developed for *NSGA-II*. However, in *TorusNDS* the typically global operators (*fastNonDominatedSort* and *calculateCrowdedDistance*) are now applied locally within demes. Given a deme size of d individuals, the complexity of *TorusNDS* will be $O(MNd)$. The fact that d is constant and is typically $\ll N$, the overall complexity of *TorusNDS* should increase linearly with respect to population size.

4 Experimental Results

The *TorusNDS* method was compared with *NSGA-II* on six benchmark multiobjective problems from previous studies (Zitzler, Deb & Thiele 2000, Deb et al. 2002). The problems are designed to test an algorithm's ability to converge upon the global Pareto-optimal front under varying levels on multimodality, deception and convexity. All experiments were run with the parameters as specified in (Deb et al. 2002). Additional experiments with larger population sizes $N = 225$ and $N = 400$ were also performed. Each configuration was repeated 30 times to gather the necessary statistics. Three performance measures were considered; the ability of an algorithm to converge upon the global Pareto-optimal front, the spread of solutions along that front, and the amount of CPU time required to complete a run. The first two measures were taken by visual inspection of the non-dominated solutions produced by the algorithm. Results from typical runs are shown in Figures 1–6. A comparison of CPU time for *TorusNDS* and *NSGA-II* is shown in Figure 7.

4.1 Discussion of Results

The results suggest that *TorusNDS* typically requires more generations to converge upon the global Pareto-optimal front, regardless of population size. This was true for five out of six of the test cases, with the exception of the fifth test case. This is not surprising for two reasons; first, it is well known that SSEAs demonstrated weaker selection pressures than their traditional counterparts (Sarma 1998). Second, while *NSGA-II* applies a selection pressure when choosing parents, *TorusNDS* picks parents at random from within a deme. These two factors mean that *TorusNDS* should exhibit much less selection pressure on individuals than *NSGA-II* and hence will require more time for convergence.

In terms of diversity, it appears that *TorusNDS* is not able to cover as much of the global Pareto-optimal front as *NSGA-II* when using smaller population sizes. This may be in part due to the above convergence problem – fewer individuals have had a chance to converge upon the optimal front, therefore there is less coverage. Another reason may be that the spatial effect increases as the ratio between population radius and deme radius increases (Sarma 1998). With the larger populations, the local effect, and hence local genetic divergence is greater, which permits more thorough coverage of the global Pareto-optimal front.

Finally, as predicted, the computational resources required for *TorusNDS* are significantly lower than that for *NSGA-II*. By a population size of ~ 1000 , the difference is approaching an order of magnitude. This suggests that extra effort can be invested in *TorusNDS* to increase the selection pressure and the end result will still be less processor-intensive than *NSGA-II*.

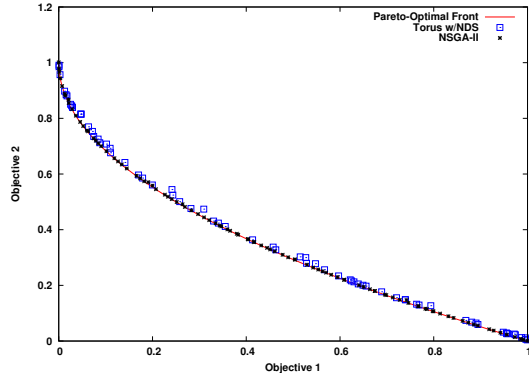
5 Conclusion

This paper has proposed a new MOEA based around the concept of spatially-structured populations. Initial testing suggests that, in terms of processor time, it is significantly faster than *NSGA-II*, an existing MOEA known for its below average complexity. However, although able to discover a diverse set of solutions along the global Pareto-optimal front of several benchmark functions, the overall quality of the search was lower than that of *NSGA-II*. This may be in part due to the lower selection pressure exerted by *TorusNDS* on the population.

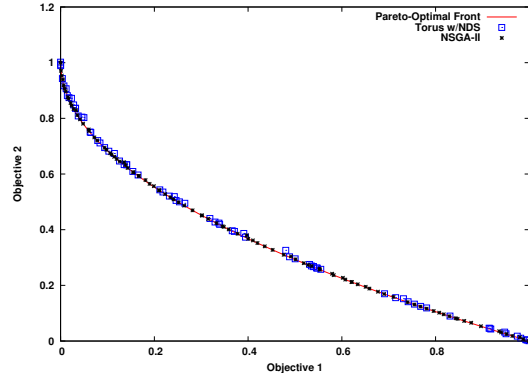
Future work should investigate the possibility of increasing the selection pressure within demes. One possible way to do this would be to perform the ranking of the deme prior to selection of parents and updating this ranking once offspring have been produced. Doing so would increase the computational complexity somewhat, although the overall behaviour should still be linear with respect to population size.

References

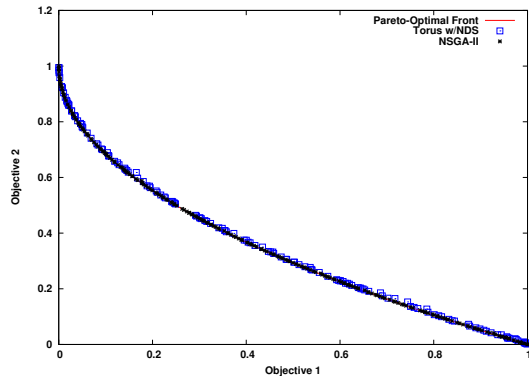
- Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons. Chichester, UK.
- Deb, K., Agrawal, S., Pratap, A. & Meyarivan, T. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II" *IEEE Trans. Evolutionary Computation*. **6**(2).
- Dick, G. & Whigham, P. A. (2005). "The behaviour of genetic drift in a spatially-structured evolutionary algorithm" In D. Corne, Z. Michalewicz, B. McKay, G. Eiben, D. Fogel, C. Fonseca, G. Greenwood, G. Raidl, K. C. Tan & A. Zalzala (eds), *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*. Vol. 2 IEEE Press Edinburgh, Scotland, UK pp. 1855–1860.
- Kirley, M. (2001). "MEA: A metapopulation evolutionary algorithm for multi-objective optimisation problems" *Proceedings of the 2001 IEEE Conference on Evolutionary Computation*. IEEE Press Seoul, Korea.



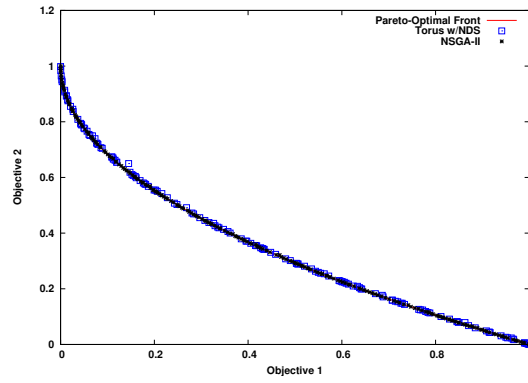
(a) $N = 100, G = 250$



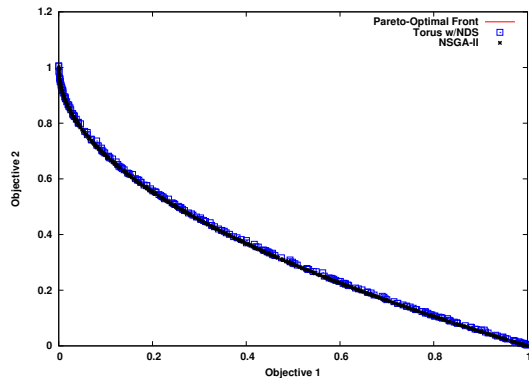
(b) $N = 100, G = 500$



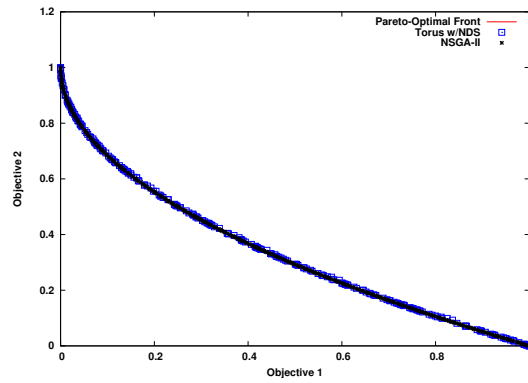
(c) $N = 225, G = 250$



(d) $N = 225, G = 500$

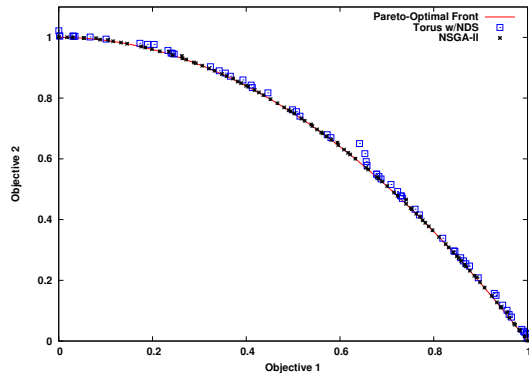


(e) $N = 400, G = 250$

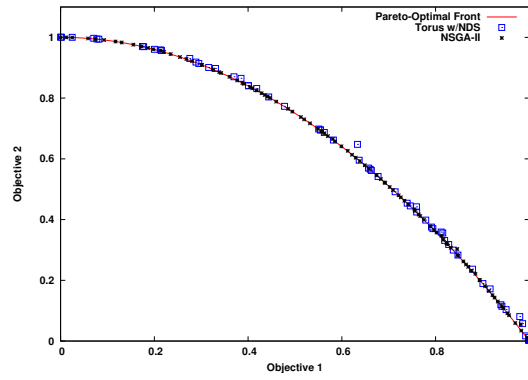


(f) $N = 400, G = 500$

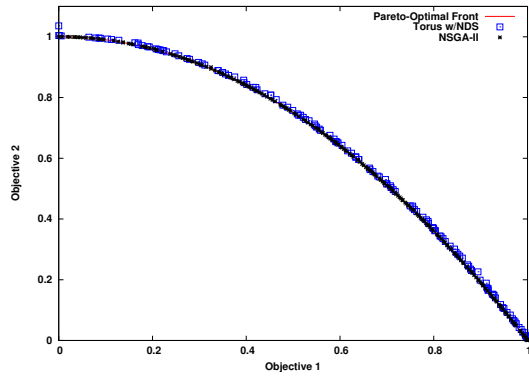
Figure 1: Comparison of *NSGA-II* and *TorusNDS* on test problem *ZDT1*.



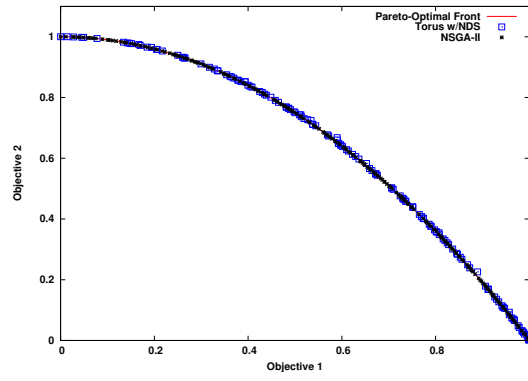
(a) $N = 100, G = 250$



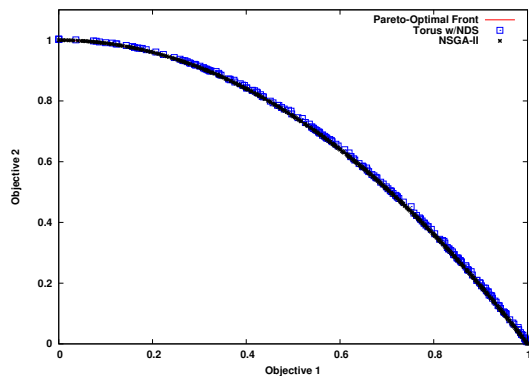
(b) $N = 100, G = 500$



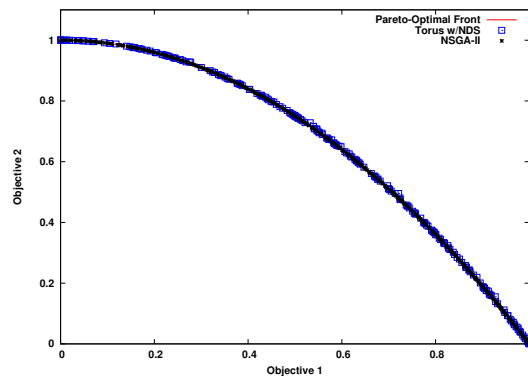
(c) $N = 225, G = 250$



(d) $N = 225, G = 500$

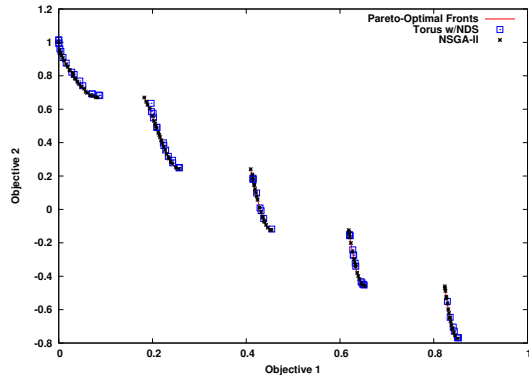


(e) $N = 400, G = 250$

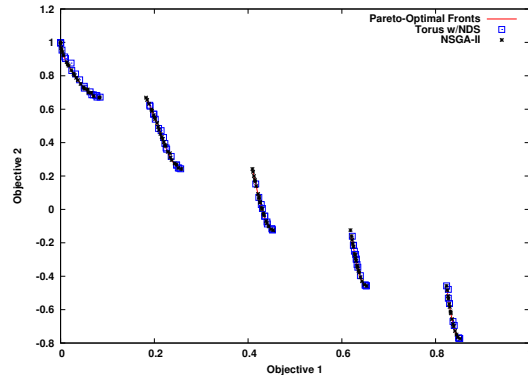


(f) $N = 400, G = 500$

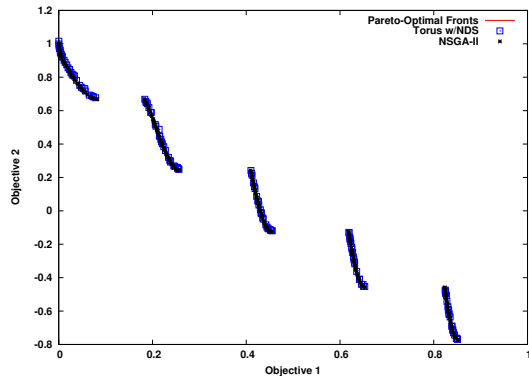
Figure 2: Comparison of *NSGA-II* and *TorusNDS* on test problem *ZDT2*.



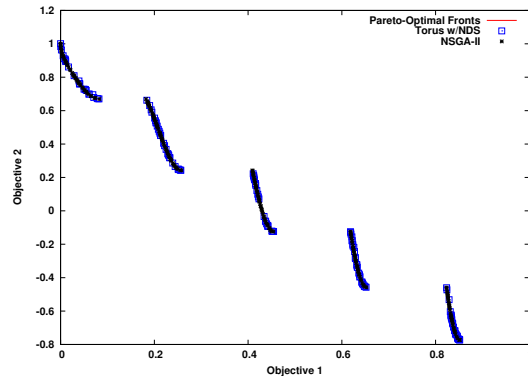
(a) $N = 100, G = 250$



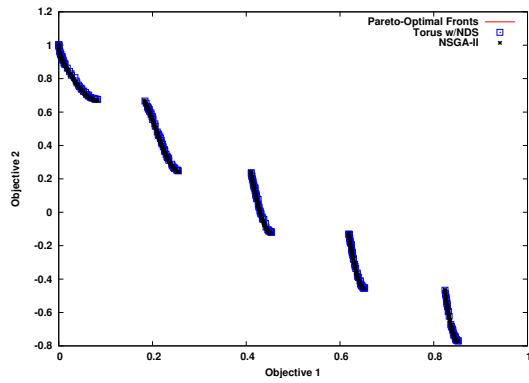
(b) $N = 100, G = 500$



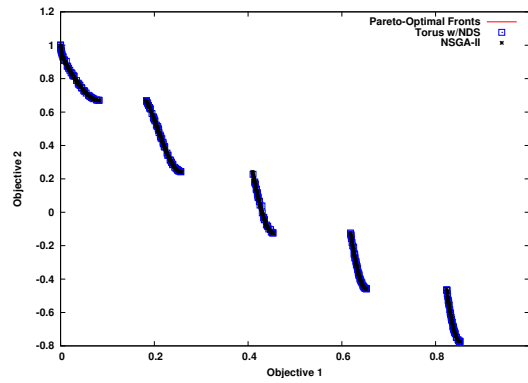
(c) $N = 225, G = 250$



(d) $N = 225, G = 500$

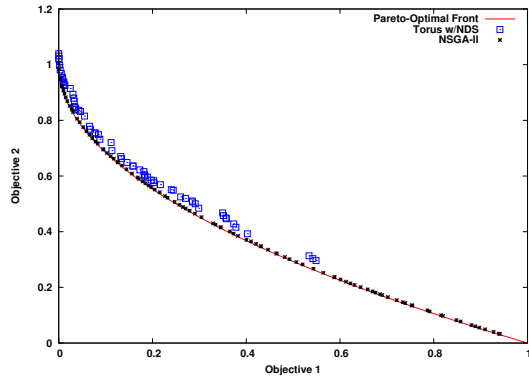


(e) $N = 400, G = 250$

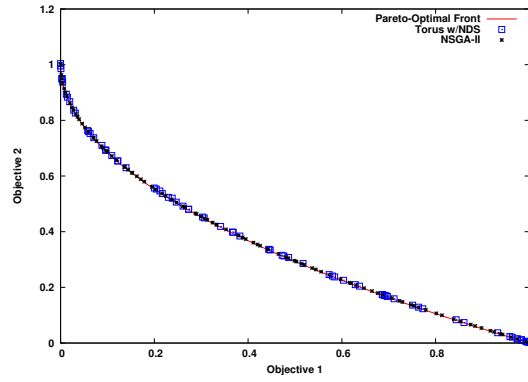


(f) $N = 400, G = 500$

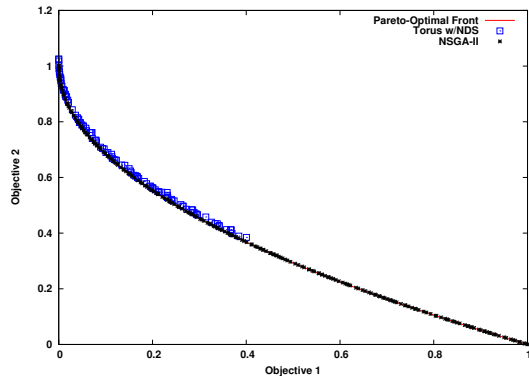
Figure 3: Comparison of *NSGA-II* and *TorusNDS* on test problem *ZDT3*.



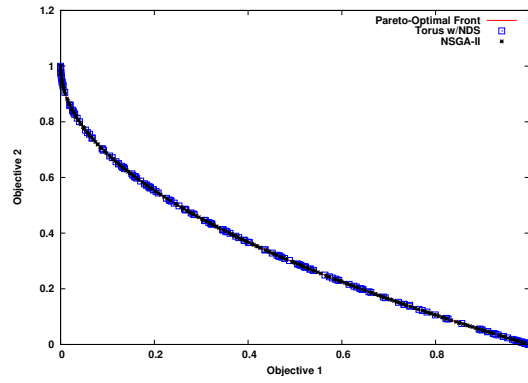
(a) $N = 100, G = 250$



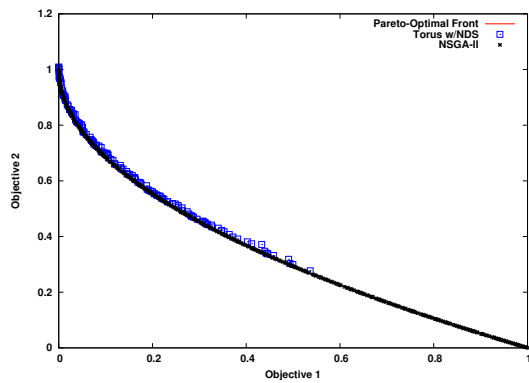
(b) $N = 100, G = 500$



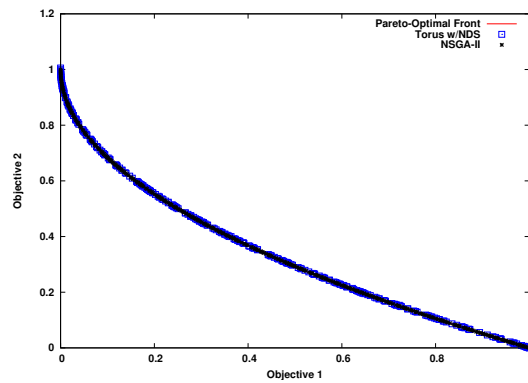
(c) $N = 225, G = 250$



(d) $N = 225, G = 500$

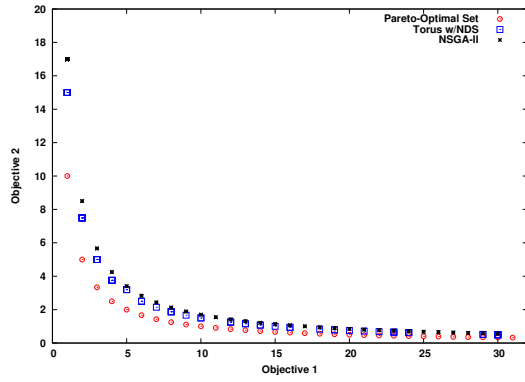


(e) $N = 400, G = 250$

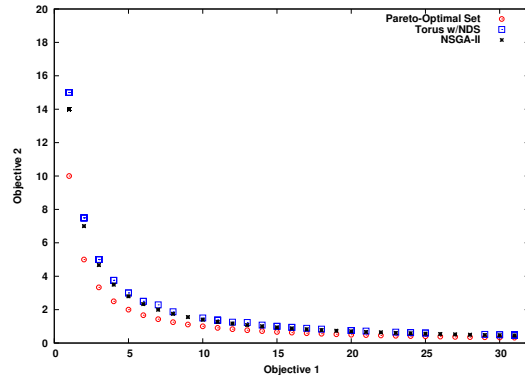


(f) $N = 400, G = 500$

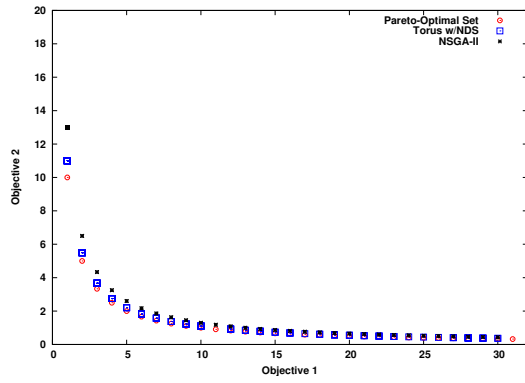
Figure 4: Comparison of *NSGA-II* and *TorusNDS* on test problem *ZDT4*.



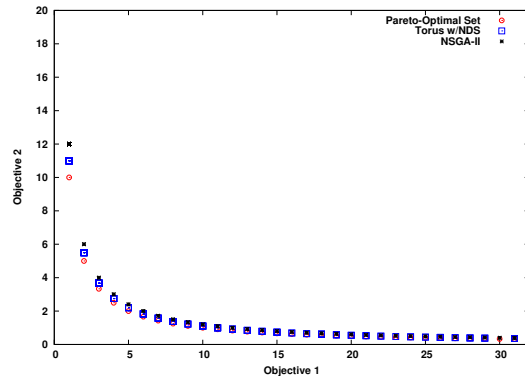
(a) $N = 100, G = 250$



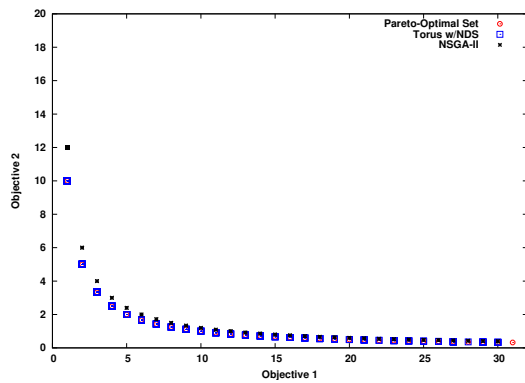
(b) $N = 100, G = 500$



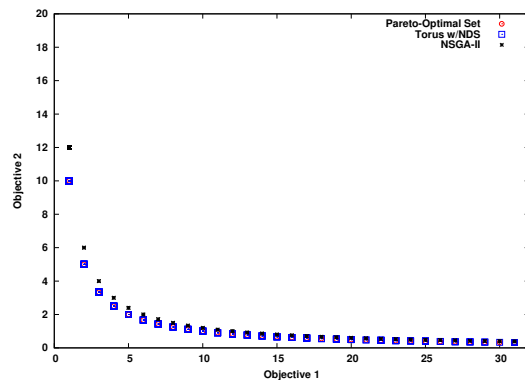
(c) $N = 225, G = 250$



(d) $N = 225, G = 500$

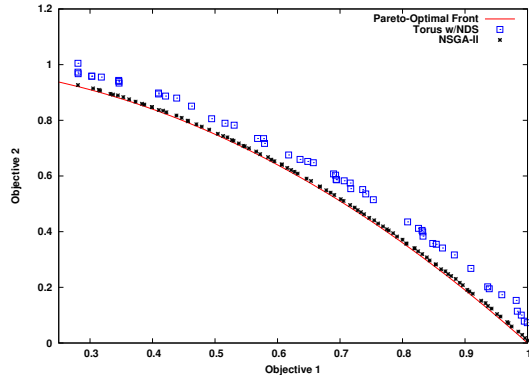


(e) $N = 400, G = 250$

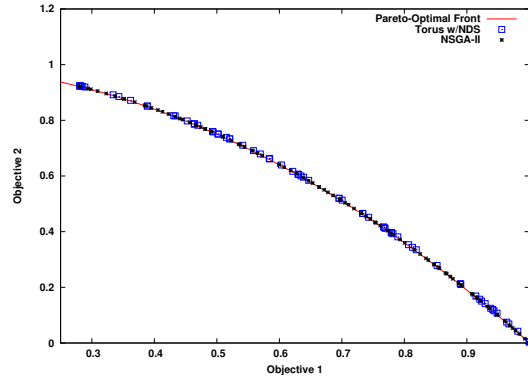


(f) $N = 400, G = 500$

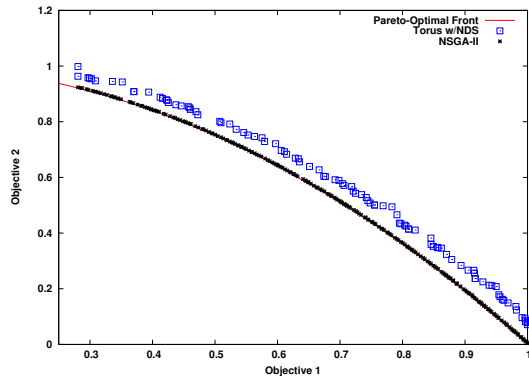
Figure 5: Comparison of *NSGA-II* and *TorusNDS* on test problem *ZDT5*.



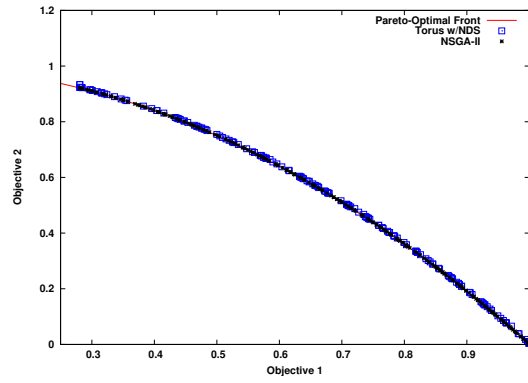
(a) $N = 100, G = 250$



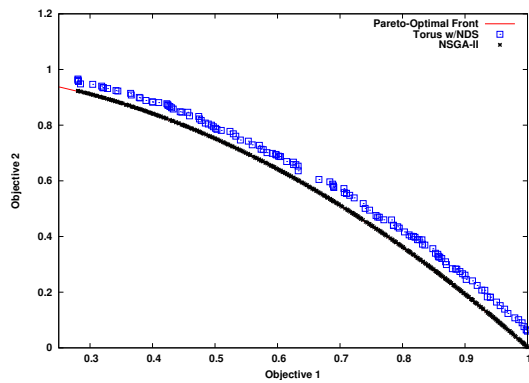
(b) $N = 100, G = 500$



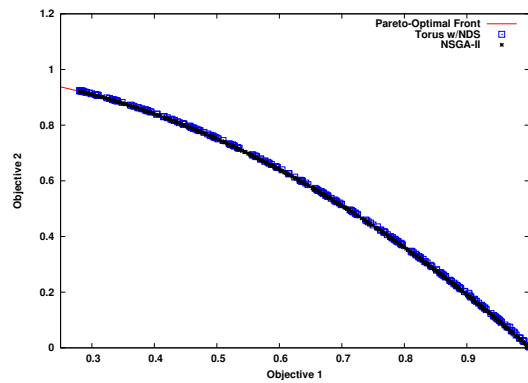
(c) $N = 225, G = 250$



(d) $N = 225, G = 500$



(e) $N = 400, G = 250$



(f) $N = 400, G = 500$

Figure 6: Comparison of *NSGA-II* and *TorusNDS* on test problem *ZDT6*.

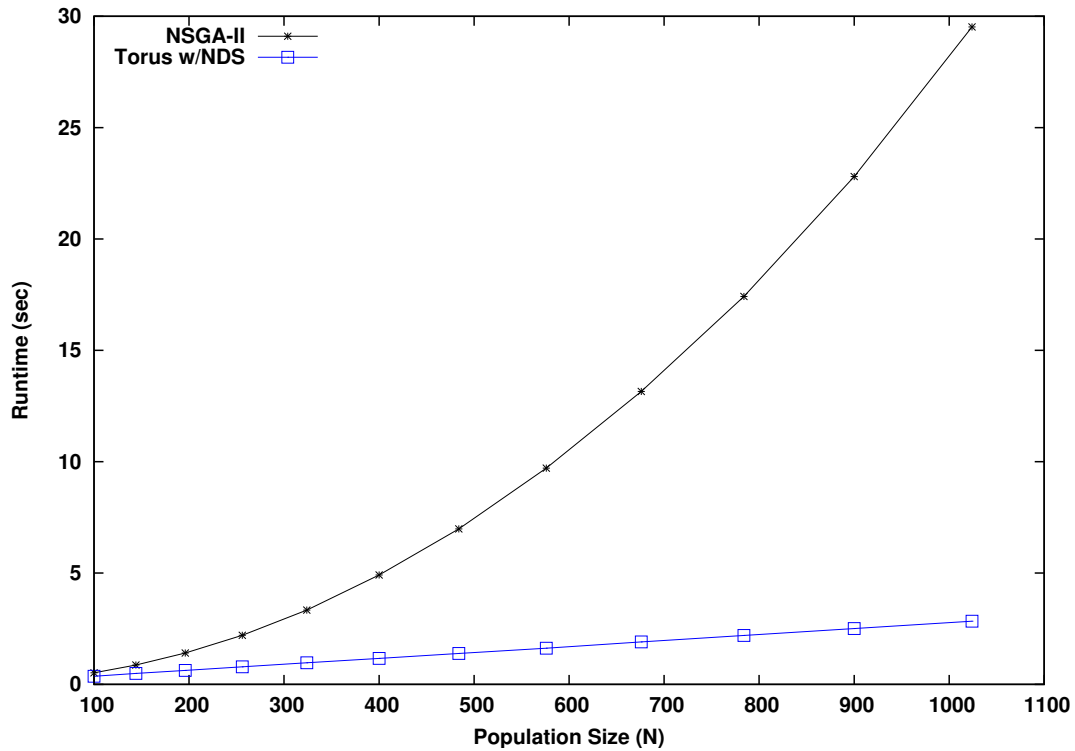


Figure 7: Runtime performance of *NSGA-II* and *TorusNDS* on test problem *ZDT6*.

- Murata, T., Ishibuchi, H. & Gen, M. (2000). “Cellular Genetic Local Search for Multi-Objective Optimization” In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee & H.-G. Beyer (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. Morgan Kaufmann Las Vegas, Nevada, USA pp. 307–314.
- Sarma, J. (1998). *An Analysis of Decentralized and Spatially Distributed Genetic Algorithms*. PhD thesis George Mason University Fairfax VA, USA.
- Tomassini, M. (2005). *Spatially structured evolutionary algorithms*. Springer.
- Zitzler, E., Deb, K. & Thiele, L. (2000). “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results” *Evolutionary Computation*. **8**(2): 173–195.