An Algebraic Differential Evolution for the Linear Ordering Problem

Valentino Santucci
Dept. of Math. and Comp. Sc.
University of Perugia, Italy
valentino.santucci@unipg.it

Marco Baioletti
Dept. of Math. and Comp. Sc.
University of Perugia, Italy
marco.baioletti@unipg.it

Alfredo Milani
Dept. of Math. and Comp. Sc.
University of Perugia, Italy
alfredo.milani@unipg.it

ABSTRACT

In this paper we propose a discrete algebraic-based Differential Evolution for the Linear Ordering Problem (LOP). The search space of LOP is composed by permutations of objects, thus it is possible to use some group theoretical concepts and methods. Indeed, the proposed algorithm is a fully discrete Differential Evolution scheme and has been designed by exploiting the group structure of LOP solutions in order to mimic the classical Differential Evolution behavior observed in continuous numerical spaces. The performances have been evaluated over widely known LOP benchmark suites and have been compared to the state-of-the-art results.

1. INTRODUCTION

The Linear Ordering Problem (LOP) is a classical combinatorial optimization problem [3]. LOP has received considerable attention because of its many applications in diverse research fields such as, among the others, economy, graph theory, archeology and computational social choice [3].

The goal of LOP is, given a $n \times n$ matrix $H = (H_{i,j})$, to find a permutation π of the row and column indices $\{1, \ldots, n\}$ which maximizes the objective function

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} H_{\pi(i),\pi(j)}. \quad (1)$$

Since LOP is a NP-hard problem, exact methods are able to find optimal solutions only in small problem instances. Anyway, the permutation structure of the LOP solutions allows to apply a variety of meta-heuristics and evolutionary algorithms specifically designed for permutation-based search spaces.

According to [6] and the more recent [2], MA and ILS are to be considered the state-of-the-art algorithms for LOP.

ILS [6], starting from a random solution, iteratively alternates the two phases of local search (using the insertion neighborhood) and perturbation (shaking the local optimum with a given number of interchange moves).

GECCO '15 July 11-15, 2015, Madrid, Spain

ACM ISBN 978-1-4503-3488-4/15/07.

DOI: http://dx.doi.org/10.1145/2739482.2764693

MA [6] evolves a small population of distinct local optima by applying the same local search as in ILS to the offspring solutions generated by the OBX crossover operator [7].

Further improvements to ILS and MA have been proposed in [2], whose resulting variants of ILS and MA are referred, respectively, as $\mathrm{ILS_R}$ and $\mathrm{MA_R}$.

2. DIFFERENTIAL EVOLUTION FOR PERMUTATIONS

Differential Evolution for Permutations (DEP) algorithm has been successfully proposed in [5] for the Permutation Flowshop Scheduling Problem (PFSP), a combinatorial problem that share with LOP the same representation of solutions as permutations.

DEP mainly consists in a completely discrete variant of the popular numerical Differential Evolution (DE) algorithm [4]. The core component of numerical DE is its differential mutation operator that allows to self-adapt DE population to the objective function landscape at hand by exploiting the population solutions differences.

DEP mimics the same behavior of classical DE in the combinatorial space of permutations. Its key idea resides in the definition of the operations of difference, sum, and truncation on the permutations space. These operations are somehow consistent with their usual definitions in the classical numerical space $\mathbb{R}^n$. This is made possible by exploiting the algebraic structure of the permutations space where its elements, i.e. the permutations, form a group in the algebraic sense.

The Differential Evolution for the LOP Permutations space (DEP) directly evolves a population of NP permutations $\pi_1, \ldots, \pi_{NP}$.

The population is initialized with NP uniformly random permutations obtained by means of the well known Fisher-Yates shuffle.

For each population individual π_i, a mutant permutation ν_i is generated according to

$$\nu_i = \pi_{r_0} \circ \left(F \odot \left(\pi_{r_2}^{-1} \circ \pi_{r_1}\right)\right)$$

where $\pi_{r_0}, \pi_{r_1}, \pi_{r_2}$ are three distinct randomly selected elements of the current population, $\circ$ is the permutation composition operator and $\odot$ is the operator, described in [5], which allows to compute the scaled difference.

The crossover between the population individual π_i and the mutant ν_i is performed according to the order based crossover OBX [7], producing two offsprings $\upsilon_i^{(1)}, \upsilon_i^{(2)}$. OBX has been slightly modified in order to take into account the crossover probability parameter of DE.

Then, the next generation population individual π'_i is the best among $\pi_i, \upsilon_i^{(1)}, \upsilon_i^{(2)}$.

A restart mechanism, introduced to avoid the stagnation of the population, is triggered when all the population elements are the same. Half population is randomly regenerated using the Fisher-Yates method, while the remaining individuals are shuffled each one by a random number of adjacent swaps.

Finally, the population size NP is left free to be set by the user, while the other two parameters, i.e., the scale factor F and the crossover probability CR, are self-adapted using the popular online scheme proposed in [1].

3. EXPERIMENTS

The performances of DEP have been evaluated on a large set of widely known benchmark instances selected from [3]. Namely, we have selected the benchmark suites IO, SGB, MB and XLOLIB. The resulting benchmarks collection is composed by 183 LOP instances and it is quite heterogeneous regarding instance size and how they are generated. Moreover, optimal values are known for IO, SGB and MB instances, while the best known solutions of XLOLIB are reported in [2].

DEP population size has been set to 100 after some preliminary experiments and the algorithm has been run 10 times for each problem instance. Similarly to [2], the termination criterion has been set to $10\,000 \times n^2$ fitness evaluations.

The performance measure employed is the commonly used average relative percentage deviation (ARPD):

$$ARPD = \left(\sum_{i=1}^{10} \frac{(Best - Alg_i) \times 100}{Best} \right) /10 \qquad (2)$$

where Alg_i is the final fitness value found by the algorithm Alg in its i^{th} run, and $Best$ is the best known value for the problem instance at hand.

Owing to space constraints only the results for the XLOLIB instances with $n = 150$ are reported in Table 1 where the ARPDs obtained by DEP are compared with the state-of-the-art ARPDs provided in [2].

However, it worths to note that, on IO, SGB and MB instances, DEP was able to: (i) solve at least in one execution the 82% of the instances, (ii) solve in every execution the 54% of instances, (iii) obtain an overall ARPD lower than the 0.01%.

Regarding XLOLIB and considering that the proposed DEP scheme for LOP is still a preliminary implementation, Table 1 show that, although we were not able to match the performances of the state-of-the-art algorithms $\mathrm{MA_R}$ and $\mathrm{ILS_R}$, DEP results are anyway satisfactory. Indeed, in XLOLIB with $n = 150$, the overall ARPD of DEP is 0.66% and it is less than 0.5% greater than that of $\mathrm{MA_R}$, i.e., the best known algorithm to date. The difference is greater on $n = 250$ instances, but still small, i.e., around the 1%.

Finally, it is worthwhile to note that DEP typically reaches a good enough solution very soon and employs more than 3/4 of the evolution for small refinements. By also considering the satisfactory results on small and easy instances, this aspect clearly reveals that DEP looks to have potentialities for further improvements.

Table 1: Experimental results on XLOLIB benchmarks with $n = 150$

Instance	Best ARPD	DEP ARPD	Instance	Best ARPD	DEP ARPD
be75eec_150	0.13	0.32	t70f11xx_150	0.46	1.35
be75np_150	0.19	0.68	t70l11xx_150	0.04	0.81
be75oi_150	0.12	0.40	t70n11xx_150	0.29	0.85
be75tot_150	0.23	1.18	t74d11xx_150	0.18	0.83
stabu1_150	0.15	0.59	t75d11xx_150	0.19	0.88
stabu2_150	0.09	0.43	t75e11xx_150	0.33	0.72
stabu3_150	0.11	0.46	t75k11xx_150	0.13	0.30
t59b11xx_150	0.28	0.49	t75n11xx_150	0.25	1.04
t59d11xx_150	0.09	0.60	tiw56n54_150	0.14	0.51
t59f11xx_150	0.22	0.75	tiw56n58_150	0.16	0.91
t59n11xx_150	0.11	0.50	tiw56n62_150	0.18	0.68
t65b11xx_150	0.18	0.57	tiw56n66_150	0.24	0.55
t65d11xx_150	0.19	0.87	tiw56n67_150	0.08	0.47
t65f11xx_150	0.19	0.82	tiw56n72_150	0.16	0.51
t65l11xx_150	0.14	0.49	tiw56r54_150	0.06	0.52
t65n11xx_150	0.14	0.54	tiw56r58_150	0.15	0.65
t69r11xx_150	0.24	0.40	tiw56r66_150	0.27	0.77
t70b11xx_150	0.24	0.57	tiw56r67_150	0.18	0.69
t70d11xn_150	0.21	0.59	tiw56r72_150	0.14	0.70
t70d11xx_150	0.33	0.96			

Avg ARPDs: DEP 0.66, $\mathrm{MA_R}$ 0.19, $\mathrm{ILS_R}$ 0.24, MA 0.19, ILS 0.24

4. ACKNOWLEDGMENTS

This work was partially supported by the University of Perugia–DMI project “Mobile Knowledge Agents in Evolutionary Environments” and by the DMI laboratory KITLab (“Knowledge and Information Technology Laboratory”).

5. REFERENCES

[1] J. Brest, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[2] J. Ceberio, A. Mendiburubu, and J. Lozano. The linear ordering problem revisited. *European Journal of Operational Research*, 241(3):686–696, 2015.

[3] R. Martí and G. Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Springer-Verlag, Berlin Heidelberg, 2011.

[4] K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin Heidelberg, 2005.

[5] V. Santucci, M. Baioletti, and A. Milani. A differential evolution algorithm for the permutation flowshop scheduling problem with total flow time criterion. In *Parallel Problem Solving from Nature - PPSN XIII*, Lecture Notes in Computer Science, vol. 8672, pages 161–170, 2014.

[6] T. Schiavinotto and T. Stutzle. The linear ordering problem: instances, search space analysis and algorithms. *Journal of Mathematical Modelling and Algorithms*, 3(4):367–402, 2005.

[7] G. Syswerda. Schedule optimization using genetic algorithms. In *Handbook of Genetic Algorithms*, pages 332–349, New York, 1991. Van Nostrand-Reinhold.