Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 1461–1470

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Evolutionary Algorithm Using Random Immigrants for the Multiobjective Travelling Salesman Problem

Krzysztof Michalak[a]

[a] *Department of Information Technologies, Wrocław University of Economics and Business, Wrocław, Poland*

Abstract

This paper addresses the Multiobjective Travelling Salesman Problem (MoTSP) with the aim to study the effects of including random immigrants in the population of solutions processed by the evolutionary algorithm. Random immigrants are typically used in evolutionary optimization in order to increase the diversity of the population and to allow the algorithm to explore a larger area of the search space. Introducing random immigrants incurs a certain overhead which is especially significant in combinatorial optimization, because local search procedures are usually employed, which, while effective in improving the solutions, are computationally expensive. In this paper several strategies of introducing new specimens are tested with the aim of improving the effectiveness of the optimization process given a limited computation time. In the experiments the proposed approach was tested on kroAB*nnn* instances of the MoTSP. It was found to improve the results of multiobjective optimization in terms of both the hypervolume and the IGD indicators. The most effective immigration strategy turned out to be to decrease the number of immigrants with time.

Keywords: Combinatorial Optimization; Multiobjective Optimization; Random immigrants
2020 MSC: 68W50; 90B06; 90C27; 90C29; 90C59

1. Introduction

In this paper an evolutionary algorithm is used to solve the Multiobjective Travelling Salesman Problem (MoTSP). The MoTSP is one of the wide range of optimization problems connected to transport optimization and the presence of multiple objectives is motivated by the need to minimize, for example, the travel time and the travel cost at the same time. These objectives are clearly in conflict, because faster means of transport are usually more expensive. This situation is typical in multiobjective optimization problems in which each solution x in the search space Ω is evaluated with respect to m, often conflicting, objectives $f_i : \Omega \to \mathbb{R}$, $i = 1, \ldots, m$. Without a loss of generality it can be assumed

E-mail address: krzysztof.michalak@ue.wroc.pl

10.1016/j.procs.2021.08.150

that in a multiobjective optimization problem (MOP) the goal is to jointly minimize $F(x) = (f_1(x), \dots, f_m(x))$ subject to $x \in \Omega$.

Solutions in Ω are often not comparable, because one solution may minimize one of the objectives, while some other solution may minimize some other objective. Therefore, instead of linearly sorting the solutions, the notion of Pareto dominance is used. Denote by $F(x_1) = (f_1(x_1), \dots, f_m(x_1))$ and $F(x_2) = (f_1(x_2), \dots, f_m(x_2))$ the objective vectors of two solutions $x_1, x_2 \in \Omega$. We say that x_1 *Pareto-dominates* x_2 ($x_1 \succ x_2$) when $\forall i \in \{1, \dots, m\} : f_i(x_1) \leq f_i(x_2)$ and $\exists i \in \{1, \dots, m\} : f_i(x_1) < f_i(x_2)$. A solution x is said to be *nondominated* (*Pareto optimal*) when $\neg\exists x' \in \Omega : x' \succ x$ A *Pareto Set (PS)* for a given optimization problem is defined as the set of all nondominated solutions in Ω and a *Pareto Front (PF)* is the set of vectors of objectives of solutions in the Pareto Set. Most often the goal of the multiobjective optimization is to find a good approximation of the PF for a given optimization problem with the assumption that a particular solution can be selected later by a decision maker.

The remaining part of the paper is structured as follows. Section 2 discusses the problem of diversity loss in population-based optimization. Section 3 presents the algorithm used for testing the mechanisms based on random immigrants proposed in this paper for improving the evolutionary optimization for the MoTSP problem. In Section 4 the experiments are described and the results obtained in these experiments are discussed. Section 5 concludes the paper.

2. Loss of Diversity in Population-Based Optimization

One of the problems that arise in population-based optimization methods is the loss of diversity in the population. When optimization progresses, a bigger and bigger part of the population tends to concentrate in a small area in the search space. The quality of the optimization results may suffer if this convergence happens to direct the population towards a suboptimal solution. This issue becomes even more significant in the case of those optimization problems for which it is intrinsically impossible to limit the solution to the problem to just one, the best, point in the search space. Three classes of optimization problems that typically require finding more than one solution are multimodal, multiobjective and dynamical problems.

2.1. Preventing the loss of the diversity in multimodal optimization

In multimodal problems, there often exists a number of solutions that can be considered "good" and which may be located in different areas of the search space. From the decision-making point of view not only the single best solution, but also a diversified set of good solutions might be of interest. For solving multimodal problems multipopulation algorithms can be used in which each subpopulation works in a different area of the search space. Multipopulation algorithms based on the island model were used for evolving spiking neural networks (SNNs) [17], solving combinatorial optimization problems [12] as well as for numerical optimization [2]. In a paper written by Giacobini et al. [9] methods based on small-world topologies are described. Other approaches can be found in the literature, for example species conservation [14].

2.2. Preventing the loss of the diversity in multiobjective optimization

As mentioned in Section 1 multiobjective problems require optimizing several, often conflicting, objectives. Therefore, one has to find trade-offs in which, to make one objective better, it is necessary to allow worse values for at least some of the remaining objectives. The diversity of solutions is important in multiobjective optimization, because they represent alternatives from which the decision maker can select the most suitable one. There are many methods for dealing with the loss of the diversity in multiobjective optimization. Various multipopulation approaches to multiobjective optimization are described by Talbi et al. [21], for example cooperating subpopulations. Other authors [11, 16] applied a multi-start approach in which separate runs of the optimization algorithm were performed starting from different initial solutions or with different values of algorithm parameters. Two well-known groups of algorithms for multiobjective optimization are those that use the notion of Pareto dominance and those based on decomposition. Pareto domination relation can be used for numerical evaluation of specimens as it is done in algorithms such as SPEA [34] and SPEA-2 [33] or for selection in NSGA [19] and NSGA-II [7]. In both cases additional mechanisms are used for increasing the diversity, for example fitness sharing or tournament selection taking into consideration the crowding distance. Multiobjective problem decomposition also prevents the loss of the diversity because different specimens in the population solve different subproblems and thus move towards different parts of the Pareto front. Probably the most notable example of this approach is the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) proposed by Zhang and Li [13, 32]. In MOEA/D each specimen is assigned to a scalar subproblem in which the objective function is an aggregation of the original objectives, but with different weights for each specimen.

2.3. Preventing the loss of the diversity in dynamical optimization

In dynamical problems the need for population diversity stems from the fact that the environment changes in time and the best solution at a certain time instant may not be the best one after an environmental change. If the population converges to a small neighborhood of the best solution for the past time instant it can be hard to find a new optimum, especially, if it happens to be located in a different area in the search space. Similarly as in the case of multiobjective optimization, in dynamical optimization multipopulation algorithms are also used. Some multipopulation approaches to dynamical optimization that can be found in the literature are Self-Adaptive Differential Evolution algorithm (jDE) [3], Shifting Balance GA (SBGA) [4], Forking Genetic Algorithms (FGAs) [25], and the multinational GA (MGA) [26]. Algorithms tackling dynamical problems, which use different methods for keeping the diversity high are also discussed in the literature. One of the examples is the CHC algorithm [8] which uses a conservative selection strategy with a diversity-promoting recombination which only allows the specimens to mate if they are different enough from each other. Another mechanism for increasing the diversity of the population that is used in the CHC algorithm is catastrophic mutation which is triggered if the diversity of the population drops too low and reinitializes the population. This algorithm has been applied, among the others, to the dynamic Travelling Salesman Problem [18]. Another method for increasing the diversity, originally proposed for dynamical problems, is the inclusion of random immigrants [10] discussed in more detail in the next section.

2.4. Random immigrants

To date a number of papers have been published that follow the random immigrants approach originally proposed for dynamical optimization [10]. For example, applications of this approach to dynamical optimization of mobile networks [5] and to dynamical numerical optimization [28] were studied. An empirical analysis of various immigration schemes in the context of dynamical optimization was presented in a work of Yu et al. [31]. The random immigrants mechanism was also combined with other approaches. For example, in an algorithm named SORIGA [24] the RIGA algorithm [10] was combined with the Bak-Sneppen model [1]. Another algorithm named EIGA [29, 30] used elitism-based immigrants. In the context of combinatorial optimization, algorithms involving random immigrants were used, for example, to solve the single-objective Asymmetric Travelling Salesman Problem (ATSP) [20].

In this paper random immigrants, are used to increase diversity in the case of an evolutionary algorithm working on the Multiobjective Travelling Salesman Problem (MoTSP). The motivations for employing this mechanism for the Multiobjective TSP come from two different aspects of this problem and are, therefore, twofold:

1. In combinatorial problems the objective function is not smooth and the algorithms tend to fall into local suboptima. Including random immigrants in the optimization process can help avoid getting stuck in such suboptimum.
2. In multiobjective optimization it is not only important to improve convergence to the Pareto front, but also to find a good range of trade-offs between conflicting objectives. This aspect of multiobjective optimization also motivates the attempt to keep the diversity of the population high.

This paper studies two aspects that influence the immigration process. First, three different strategies of adjusting the number of the immigrants during the runtime of the algorithm are studied and are compared to an algorithm without the immigration. Second, an approach is tested in which the immigrants are allowed to exist for a certain number of generations in a separate subpopulation. This mechanism is proposed to increase the chances of the immigrants to evolve without being instantly eliminated by well-adapted specimens from the main population.

3. The Multiobjective Optimization Algorithm with Random Immigrants

The algorithm used in this paper is an evolutionary algorithm tackling the Multiobjective Travelling Salesman Problem (MoTSP) with the aim at generating a good approximation of the Pareto front for each problem instance. It is based on the NSGA-II [7] elitist selection mechanism and incorporates a Pareto Local Search (PLS) procedure based on the 2-opt operator [6]. For generating new specimens the Inver-Over operator dedicated to the TSP [22, 27] is used instead of separate crossover and mutation operators. An overview of the algorithm is given in Algorithm 1. Presented algorithm uses the following procedures and functions:

- **Evaluate** - calculates values of the objective functions f_i, $i = 1, \ldots, m$ for specimens from a given set.
- **InverOver** - generates the requested number of specimens based on a given population using the Inver-Over operator [22, 27].
- **NewSpecimens** - generates the requested number of randomly generated specimens which, in the case of MoTSP, are random permutations of length N.
- **PLS** - performs the Pareto Local Search. In this paper a best-improvement local search using a neighborhood generated by the 2-opt operator [6] is used.
- **Reduce** - reduces a given population to the requested number of specimens using the elitist selection used in the NSGA-II algorithm.
- **Time** - returns the current time.

The $\mathbf{A} \sqcup \mathbf{B}$ operator selects all nondominated specimens from the union of the sets A and B.

The algorithm uses the **riMode** setting that determines how the number of random immigrants is calculated in each generation. The number of generated random immigrants is set as follows, based on the maximum number of random immigrants M_{imig} and the **riMode** setting:

- **constant** - the number of random immigrants is always M_{imig}.
- **linear_asc** - the number of random immigrants starts at 0 and increases to M_{imig} proportionally to time.
- **linear_desc** - the number of random immigrants starts at M_{imig} and decreases to 0 proportionally to time.

The t_{imig} parameter determines for how long the immigrants are allowed to evolve separately from the main population in order to allow them to improve before they have to compete with the specimens in the main population.

4. Experiments and Results

In the experiments the influence of adding random immigrants on the performance of the algorithm described in Section 3 was tested. The tests were performed on the bi-objective instances of the Travelling Salesman Problem kroAB*nnn*. The objective functions for these instances were calculated as:

$$f_j(\pi) = C_j\,[\pi(N), \pi(1)] + \sum_{i=1}^{N-1} C_j\,[\pi(i), \pi(i+1)] \quad , \tag{1}$$

for $j = 1, 2$ and for two different cost matrices C_1, C_2 taken from two single-objective instances of the TSP. In the experiments MoTSP instances with N = 100, 150, 200 and 300 cities were used with cost matrices from single-objective TSP instances kroA*nnn* and kroB*nnn* made available by Thibaut Lust [23] used as C_1 and C_2 respectively.

The parameters of the algorithm were set as follows. The population size was set to equal to the number of cities in the problem instance solved by the algorithm ($N_{pop} = N$). The random inverse rate parameter used by the inver-over operator was set to $\eta = 0.02$ which is the value used in the literature [22, 27].

Algorithm 1: The algorithm with random immigrants used for tests.

Inputs:

t_{max}	-	The maximum runtime of the algorithm
t_{imig}	-	The time interval between immigrations
N_{pop}	-	Population size
M_{imig}	-	The maximum number of random immigrants

Outputs:

P	-	Optimized population with N_{pop} solutions

```
P := NewSpecimens(N_pop)                  // Population initialization
EP := ∅                                   // Initial archive
Evaluate(P)
EP := EP ⊔ P

t_start := Time()
t_last := Time()
P_imig := ∅
while Time() < t_start + t_max do         // The main loop
    if Time() > t_last + t_imig then      // Addition of random immigrants
        if P_imig ≠ ∅ then
            P := P ∪ P_imig

        δt := (Time() − t_start)/t_max
        switch riMode do
            case constant:      N_imig := M_imig
            case linear_asc:    N_imig := M_imig · δt
            case linear_desc:   N_imig := M_imig · (1 − δt)

        P_imig := NewSpecimens(N_imig)
        Evaluate(P_imig)

        t_last := Time()

    N := |P_imig|                         // Subevolution of random immigrants
    if N > 0 then
        P' := InverOver(P_imig, N)
        Evaluate(P')
        P_imig := P_imig ∪ P'
        PLS(P_imig)
        P_imig := Reduce(P_imig, N)

    P' := InverOver(P, N_pop)             // Genetic operators and local search
    Evaluate(P')
    P := P ∪ P'
    PLS(P)

    EP := EP ⊔ P                          // Archive update

    P := Reduce(P, N_pop)                 // Population reduction
```

To numerically evaluate the performance of the algorithm the hypervolume [35] and the Inverse Generational Distance (IGD) [15] were used. The higher the value of the hypervolume and the lower the value of the IGD the better. The Inverse Generational Distance indicator requires a set of reference points. Such points can be calculated analytically if the optimization problem is formulated as a set of equations or, in the case of a constrained problem, by solving a relaxed version of the original problem. In this paper the set of reference points for the IDG indicator was calculated as a set of nondominated solutions taken from the union of sets containing nondominated solutions produced by each of the algorithms. Because, when different strategies of adding random immigrants are used, the amount of computations per generation can vary significantly the methods were compared on a constant time interval of t_{max} = 1800 seconds.

The experiments were done in two stages. In the first stage four strategies of determining the number of immigrants were tested: “constant”, “linear_asc”, “linear_desc” described in Section 3 and the “none” strategy as a comparison method in which no random immigrants were created. The maximum number of random immigrants created in each generation was set to M_{imig} = 100 and the random immigrants were added every generation which is equivalent to setting t_{imig} = 0 in Algorithm 1. For each strategy and for each test instance the algorithm was run 30 times. Median values of the hypervolume and the IGD from the 30 runs are plotted in Figures 1 and 2 against computation time.

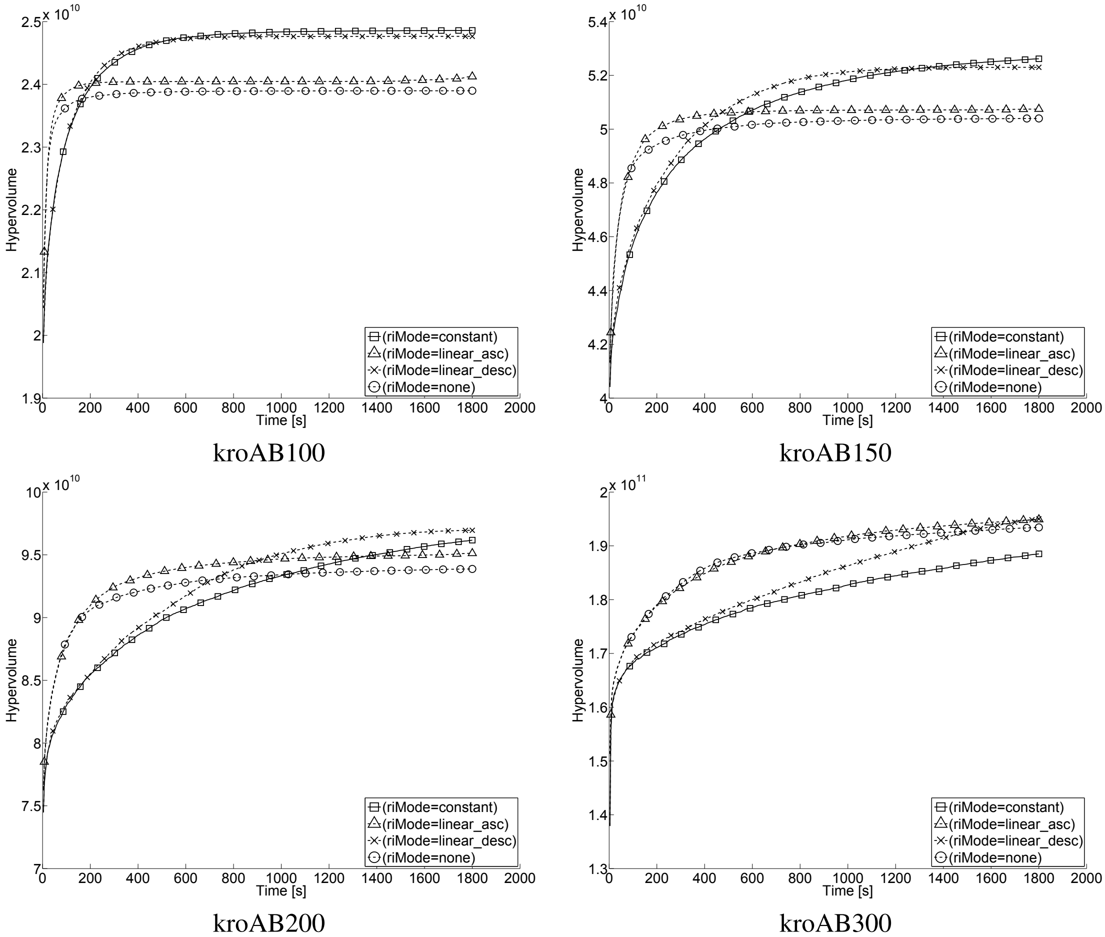


Fig. 1. Values of the hypervolume obtained in the experiments aimed at selecting the best immigration scheme plotted against time.

From Figures 1 and 2 it can be concluded that two the best performing strategies are “linear_desc” which initially adds N_{imig} immigrants to every generation, but linearly reduces this number to zero proportionally to the time of computations and “constant” which always adds M_{imig} immigrants to every generation. The “constant” strategy seems to work better for smaller instances (N = 100 and, in the case of hypervolume, also N = 150). For larger instances the

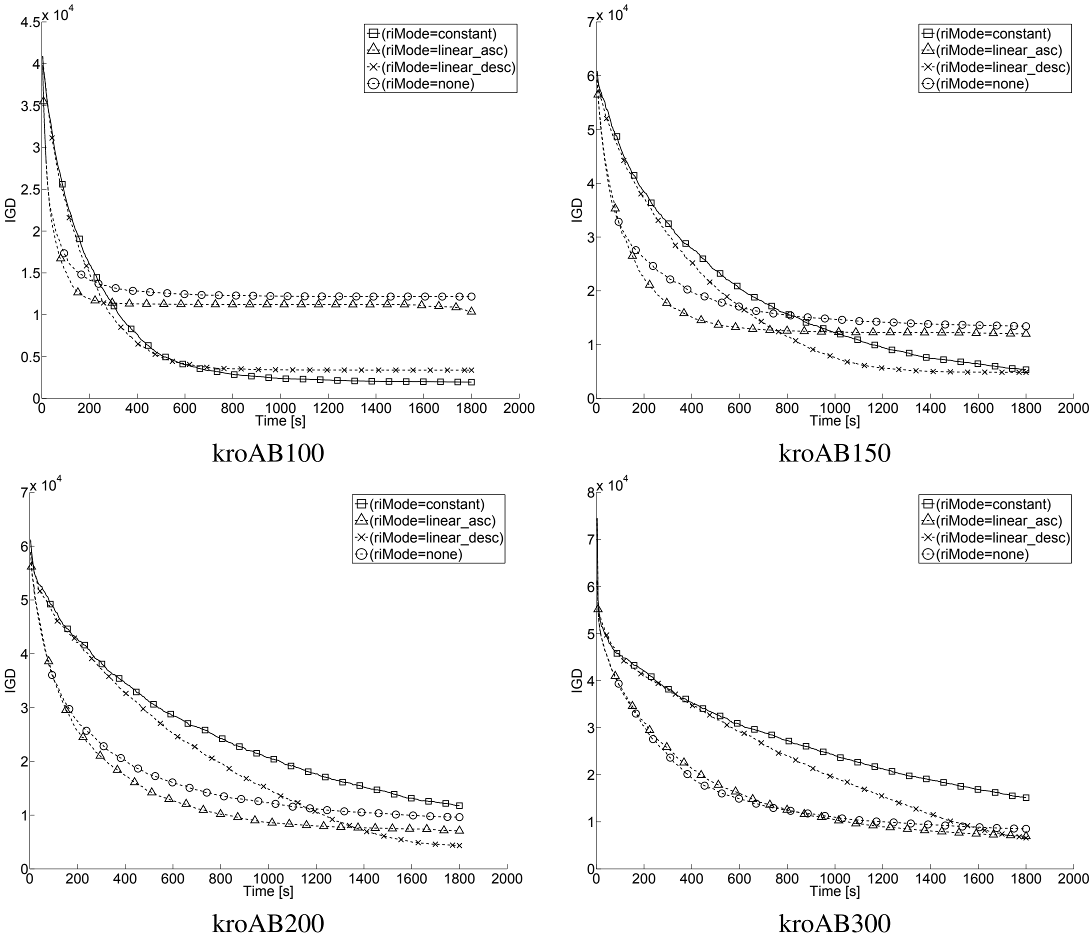


Fig. 2. Values of the IGD obtained in the experiments aimed at selecting the best immigration scheme plotted against time.

"linear_desc" strategy produced better results and it was also observed to produce better results for a short computation time limit of about 500 seconds. For these reasons further experiments were performed using the "linear_desc" strategy.

In the second stage of the experiments random immigrants were allowed to evolve separately for t_{imig} = 10, 20, 30 and 40 seconds. Median values of the hypervolume and the IGD indicators obtained in this stage of the experiments are presented in Figures 3 and 4.

From the obtained results it can be observed that for larger instances, especially if the time available for computations is short, there is some benefit in evolving random immigrants separately before introducing them into the main population. This effect can, for example, be observed in the presented results for computation time of about 500 seconds. For smaller problem instances, however, and for longer optimization times there seems to be no gain in evolving random immigrants separately from the population.

5. Conclusion

This paper studied the effects of adding random immigrants to a population in an evolutionary algorithm that tackled a combinatorial optimization problem. The method of adding random immigrants has been so far successfully used in various optimization tasks, such as dynamical optimization, multimodal optimization and others, because random immigrants introduce diversity to the population and allow the algorithm to avoid getting stuck. In this paper the same approach was used for improving the results of combinatorial optimization in the case of the Multiobjective

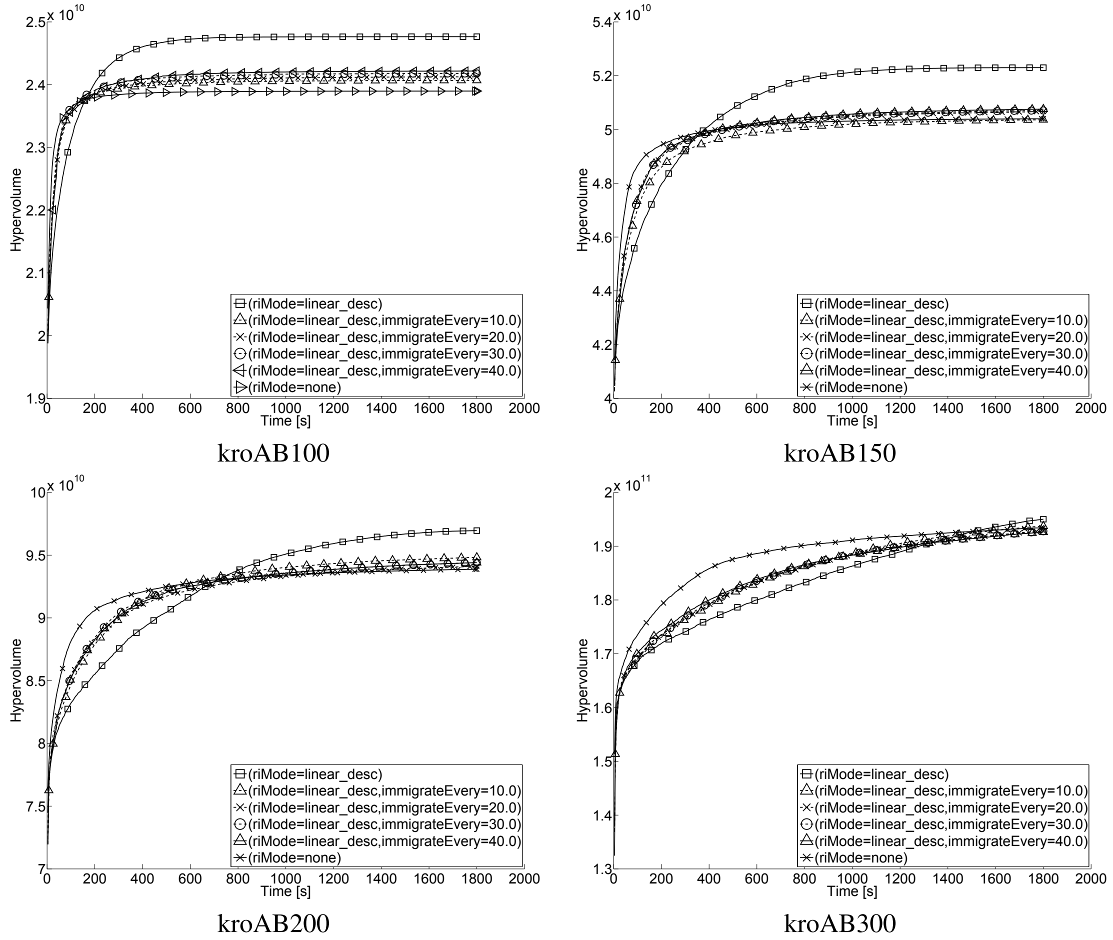


Fig. 3. Values of the hypervolume obtained in the experiments aimed at selecting the best immigration interval plotted against time.

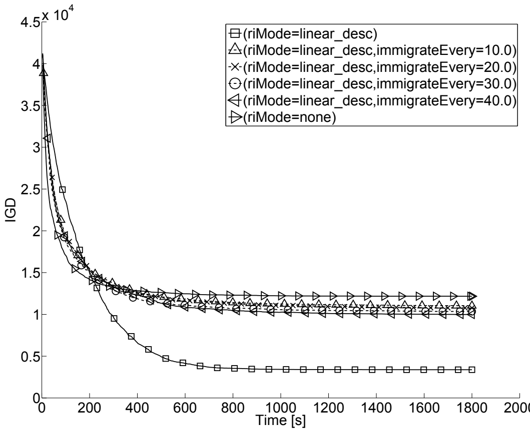
Travelling Salesman Problem (MoTSP). It has been found out that it is beneficial to add more random immigrants at the beginning of the optimization process and then gradually reduce the number of immigrants. This approach most often outperformed other approaches, however keeping a constant immigration rate also seems to work quite well for smaller instances of the optimization problem.

On the contrary, the approach of allowing the immigrants to evolve separately for some time before being added to the main population does not seem particularly beneficial. Only for short computational time limits and large instances there seems to be some benefit from using this approach.

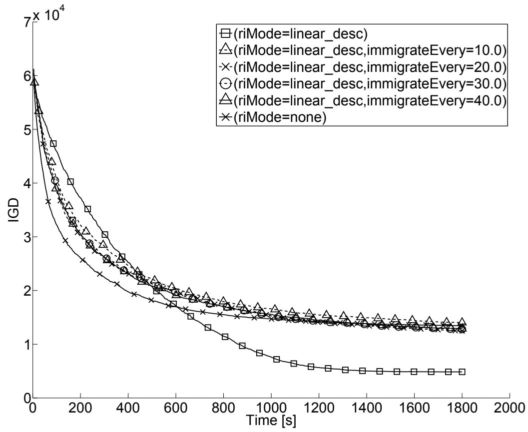
From the results obtained in this paper it seems clear that introducing random immigrants to the population can improve the results of multiobjective combinatorial optimization. However, further work could be done in order to study other strategies of adding random immigrants. For example, an adaptive approach could possibly be developed adjusting the number of immigrants and the time they are allowed to evolve separately based on the effectiveness of the immigrants in improving the results with respect to the current population.

Acknowledgment

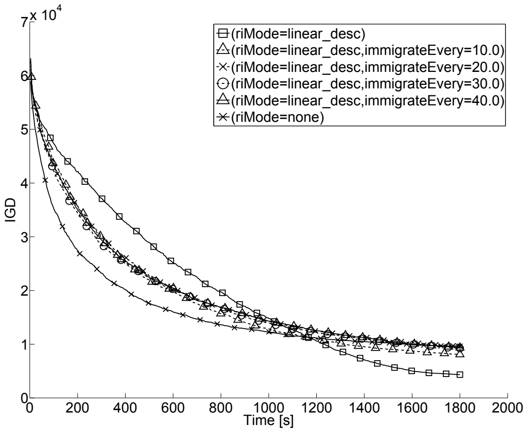
This work was supported by the Polish National Science Centre under grant no. 2015/19/D/HS4/02574. Calculations have been carried out using resources provided by Wroclaw Centre for Networking and Supercomputing (http://wcss.pl), grant No. 407.



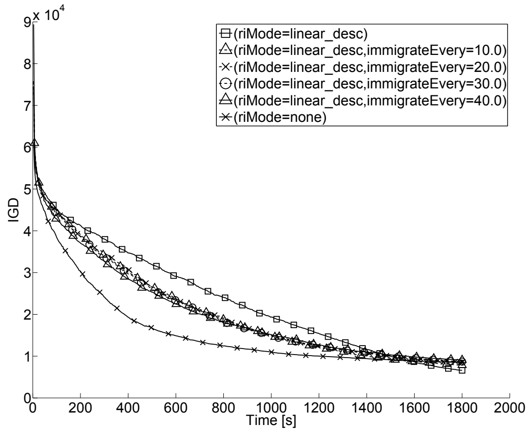
kroAB100



kroAB150



kroAB200



kroAB300

Fig. 4. Values of the IGD obtained in the experiments aimed at selecting the best immigration interval plotted against time.

References

[1] Bak, P., Sneppen, K., 1993. Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters 71, 4083–4086.

[2] Bessaou, M., Pétrowski, A., Siarry, P., 2000. Island model cooperating with speciation for multimodal optimization, in: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.P. (Eds.), Parallel Problem Solving from Nature PPSN VI. Springer, Berlin, Heidelberg. volume 1917 of *Lecture Notes in Computer Science*, pp. 437–446.

[3] Brest, J., Zamuda, A., Boskovic, B., Maucec, M.S., Zumer, V., 2009. Dynamic optimization using self-adaptive differential evolution, in: IEEE Congress on Evolutionary Computation, IEEE. pp. 415–422.

[4] Chen, J., Wineberg, M., 2004. Enhancement of the shifting balance genetic algorithm for highly multimodal problems, in: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, IEEE Press, Portland, Oregon. pp. 744–751.

[5] Cheng, H., Yang, S., 2010. Genetic algorithms with immigrants schemes for dynamic multicast problems in mobile ad hoc networks. Engineering Applications of Artificial Intelligence 23, 806–819.

[6] Chiang, C.W., Lee, W.P., Heh, J.S., 2010. A 2-opt based differential evolution for global optimization. Applied Soft Computing 10, 1200–1207.

[7] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197.

[8] Eshelman, L.J., 1991. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, in: Foundations of Genetic Algorithms. Morgan Kaufmann, pp. 265–283.

[9] Giacobini, M., Preuss, M., Tomassini, M., 2006. Effects of scale-free and small-world topologies on binary coded self-adaptive cea, in: Proceedings of the 6th European Conference on Evolutionary Computation in Combinatorial Optimization, Springer-Verlag, Berlin, Heidelberg. pp. 86–98.

[10] Grefenstette, J., 1992. Genetic algorithms for changing environments, in: Manner, R., Manderick, B. (Eds.), Parallel Problem Solving from Nature 2, Elsevier. pp. 137–144.

[11] Jozefowiez, N., Semet, F., Talbi, E.G., 2007. Target aiming Pareto search and its application to the vehicle routing problem with route balancing.

Journal of Heuristics 13, 455–469.
[12] Lardeux, F., Goëffon, A., 2010. A dynamic island-based genetic algorithms framework, in: Deb, K., Bhattacharya, A., Chakraborti, N., Chakroborty, P., Das, S., Dutta, J., Gupta, S.K., Jain, A., Aggarwal, V., Branke, J., Louis, S.J., Tan, K.C. (Eds.), Simulated Evolution and Learning, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 156–165.
[13] Li, H., Zhang, Q., 2009. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Trans. Evol. Comput. 13, 284–302.
[14] Li, J.P., Balazs, M.E., Parks, G.T., Clarkson, P.J., 2002. A species conserving genetic algorithm for multimodal function optimization. Evol. Comput. 10, 207–234.
[15] Li, M., Zheng, J., 2009. Spread assessment for evolutionary multi-objective optimization, in: Fonseca, C.M., Gandibleux, X., Hao, J.K., Sevaux, M. (Eds.), Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization, EMO '09. Springer-Verlag, Berlin, Heidelberg. volume 5467 of *Lecture Notes in Computer Science*, pp. 216–230.
[16] Mezmaz, M.S., Melab, N., Talbi, E., 2006. Using the multi-start and island models for parallel multi-objective optimization on the computational grid, in: e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on, pp. 112–120.
[17] Schuman, C.D., Plank, J.S., Patton, R.M., Potok, T.E., 2019. Island model for parallel evolutionary optimization of spiking neuromorphic computing, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, New York, NY, USA. pp. 306–307.
[18] Simões, A., Costa, E., 2011. Memory-based CHC algorithms for the dynamic traveling salesman problem, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA. pp. 1037–1044.
[19] Srinivas, N., Deb, K., 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2, 221–248.
[20] Tajani, C., Abdoun, O., Lahjouji, A.I., 2017. Genetic algorithm adopting immigration operator to solve the asymmetric traveling salesman problem. International Journal of Pure and Applied Mathematics 115, 801–812.
[21] Talbi, E.G., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., Coello Coello, C.A., 2008. Parallel approaches for multiobjective optimization, in: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (Eds.), Multiobjective Optimization. Springer Berlin Heidelberg. volume 5252 of *Lecture Notes in Computer Science*, pp. 349–372.
[22] Tao, G., Michalewicz, Z., 1998. Inver-over operator for the TSP, in: Eiben, A.E., et al. (Eds.), Parallel Problem Solving from Nature - PPSN V. Springer Berlin Heidelberg. volume 1498 of *Lecture Notes in Computer Science*, pp. 803–812.
[23] Thibaut Lust, 2021. Multiobjective TSP. https://sites.google.com/site/thibautlust/research/multiobjective-tsp. Online: accessed 2021.05.24.
[24] Tinos, R., Yang, S., 2007. A self-organizing random immigrants genetic algorithm for dynamic optimization problems. Genetic Programming and Evolvable Machines 8, 255–286.
[25] Tsutsui, S., Fujimoto, Y., Ghosh, A., 1997. Forking genetic algorithms: Gas with search space division schemes. Evolutionary Computation 5, 61–80.
[26] Ursem, R.K., 2000. Multinational GA Optimization Techniques in Dynamic Environments, in: Whitley, D., Goldberg, D., Paz, C.E., Spector, L., Parmee, I., Beyer, H.G. (Eds.), Genetic and Evolutionary Computation Conference, Morgan Kaufmann. pp. 19–26.
[27] Wang, Y., Li, J., Gao, K., Pan, Q., 2011. Memetic algorithm based on improved inver-over operator and Lin-Kernighan local search for the Euclidean traveling salesman problem. Computers & Mathematics with Applications 62, 2743–2754.
[28] Wu, Y., Liu, X., 2013. Genetic algorithms with immigrants scheme for dynamic optimization problems. Applied Mechanics and Materials 333-335, 1379–1383.
[29] Yang, S., 2007. Genetic algorithms with elitism-based immigrants for changing optimization problems, in: Giacobini, M. (Ed.), Applications of Evolutionary Computing. Springer Berlin Heidelberg. volume 4448 of *Lecture Notes in Computer Science*, pp. 627–636.
[30] Yang, S., 2008. Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. Evolutionary Computation 16, 385–416.
[31] Yu, X., Tang, K., Chen, T., Yao, X., 2009. Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. Memetic Computing 1, 3–24.
[32] Zhang, Q., Li, H., 2007. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11, 712–731.
[33] Zitzler, E., Laumanns, M., Thiele, L., 2002a. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for multiobjective optimization, in: Giannakoglou, K., et al. (Eds.), Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE). pp. 95–100.
[34] Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3, 257–271.
[35] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G., 2002b. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7, 117–132.