Study on the Impact of the NS in the Performance of Meta-Heuristics in the TSP

A. S. Santos*, A. M. Madureira**, M. L. R. Varela*,
*University of Minho / School of Engineering - Dept. of Production and Systems, Guimarães, Portugal
**Polytechnic Institute of Porto / School of Engineering - GECAD Research Group, Porto, Portugal
abg@isep.ipp.pt, amd@isep.ipp.pt, leonilde@dps.uminho.pt

Abstract — **Meta-heuristics have been applied for a long time to the Travelling Salesman Problem (TSP) but information is still lacking in the determination of the parameters with the best performance. This paper examines the impact of the Simulated Annealing (SA) and Discrete Artificial Bee Colony (DABC) parameters in the TSP. One special consideration of this paper is how the Neighborhood Structure (NS) interact with the other parameters and impacts the performance of the meta-heuristics. NS performance has been the topic of much research, with NS proposed for the best-known problems, which seem to imply that the NS influences the performance of meta-heuristics, more that other parameters. Moreover, a comparative analysis of distinct meta-heuristics is carried out to demonstrate a non-proportional increase in the performance of the NS.**

Keywords - Meta-heuristics, Simulated Annealing, Discrete Artificial Bee Colony, Neighborhood Structures, TSP.

I. INTRODUCTION

Complex Combinatorial Problems (COPs) such as the Travelling Salesman Problem (TSP) is a classic combinatorial optimization problem, which has been applied in, logistics, transportation, networking and commercial domains [1-7]. It is certainly one of the most well-known and widely studied problem in the domain of COPs that can be defined as the problem of finding a minimum distance tour of N cities, starting and ending at the same city and visiting other city exactly once [1]. In this paper, a Simulated Annealing (SA) and Artificial Bee Colony algorithm (DABC) are used to approach TSP and to determine how much the chosen NS impacts the performance of either meta-heuristics.

Computational tests were performed to demonstrate that the proposed meta-heuristics, for instance, the SA algorithm, and the DABC algorithm are suitable for being used in the Euclidian-TSP. Moreover, in this paper the authors intend to show that NS should be treated as another parameter, that influences the performance of the optimization techniques. Therefore, in this paper four well-known NS are examined with SA and DABC in the Symmetric Euclidian TSP. Furthermore, a statistical analysis of the performance of the NS is presented.

The structure of this paper is the following: section II overviews the developments in meta-heuristics, and presents in more detail Simulated Annealing (SA) and Discrete Artificial Bee Colony algorithm (DABC). Section III briefly describes the neighbourhood structure (NS) examined in the computational study. Section IV details the instances used in the computational study, the parameters for both meta-heuristics before the results are presented and explored and finally the sections ends with the statistical study to demonstrate how the NS influences the behaviour of the meta-heuristics. Section V presents the conclusions.

II. META-HEURISTICS AND DECISION MAKING

Meta-heuristics are the main techniques to address complex computational problems, such as, combinatorial optimization problems, which have shown to be the approach techniques with better performance [8]. They can be defined as an iterative procedure inspired by nature that guides a subordinate heuristic in the exploration of the solution space, [9,10]. Unlike the Neighborhood Search methods that analyze solutions within the solutions space belonging to a particular neighborhood, meta-heuristics have mechanisms to move beyond the local optima solutions, making them particularly interesting in addressing Complex Combinatorial Problems (COPs). As an approximation method, a meta-heuristic, cannot guarantee optimal solutions, but usually outperform other less evolved approximation optimization techniques. In general, meta-heuristics parameters compromise between the technique effectiveness and efficiency [9].

There are two distinct classes of meta-heuristics: the single solution based meta-heuristics and population based meta-heuristics. Meta-heuristics are then divided by the number of solutions they use. Single solution meta-heuristics are inspired by Neighborhood Search techniques, and the population-based meta-heuristics are based on Genetic Algorithms (GA). Usually the single solution based meta-heuristics have higher intensity than population meta-heuristics, since population based meta-heuristics simultaneously use several solutions to explore a larger portion of the solution space, which makes them also more time consuming [9,10].

A. Simulated Annealling

Simulated Annealing (SA) is one of the most efficient meta-heuristic to approach complex optimization problems, through adaptation of the procedure of neighborhood search techniques, which would enable it to overtake local optimums [11,12]. It is a particularly appealing meta-heuristic to address Complex Combinatorial Problems (COPs), since it can find near-optimal solutions, which, in a real-world environment are usually suitable, without much computational effort [1,7]. Further we discuss the parameters of SA to show how easily they can be tuned to optimize the performance of the meta-heuristic in a specific problem. Results in literature demonstrate how suitable the SA meta-heuristics is in the Symmetric Euclidean-TSP problem.

Simulated Annealing selects one solution between a finite number of possible solutions, but unlike neighbourhood search techniques it allows movements that worsen the current solution. It can find near-optimal solutions for combinatorial problems with low computational effort, and it is relatively simple to implement and manipulate its parameters.

978-1-5090-1897-0/16/$31.00 ©2016 IEEE

In the implemented software, SA provides alternative solutions as well as the evolution path of the meta-heuristics optimization procedure. Such implementation allows the user to visualize the movement through the solution space with advantages and disadvantages of the parameters. With such information the decision maker understands how the parameters impact the optimization technique. The motivation for using SA is four fold. First, we can solve complex combinatorial optimisation problems. Second, it can be easily understood since its procedure is based on the annealing process [11,12] and it does not include too many parameters. Third, since it is a guided-random search algorithm [11,12], it allows us to control its path through the solutions pace with the parameters of the meta-heuristic [11,12]. Finally, the algorithm is simple to implement and it does not require too much computational time to find decent solutions, since the SA search procedure is efficient and robust [11,12].

B. Discrete Artificial Bee Colony

Artificial Bee Colony (ABC) is one of the most recent optimization techniques inspired by bee behaviour, the Queen Bee Evolution (QBE), based on colony structure, Marriage in Honey Bees Optimization (MBO) in the reproduction process, and Bee Colony Optimization (BCO) or Virtual Bee Algorithm (VBA), in search of food [20], which was developed by Karaboga, in 2005 [13] and Pham et al., in 2005 [14] and was inspired by the behaviour of a colony of bees in search for food. Through this meta-heuristic the search within the solution space is performed by three types of bees: worker bees, opportunistic bees, and scout bees. ABC as been developed to approach continuous problems, not discrete problems, such as, scheduling problems. Moreover, recently there have been proposed versions of the ABC for discrete problems, the Discrete Artificial Bee Colony (DABC) procedure, including a version proposed by Karaboga & Gorkemli to address the Symmetric Euclidian TSP problem [15].

The DABC is similar to ABC, just changing the way the bees move in the neighbourhood of a source of food, which is carried out by manipulating the solutions, by transforming them into neighbourhood solutions.

The DABC works through the interaction of three phases, until the interruption criterion: the phase of the worker bees, the phase of opportunistic bees and the phase of the scout bees. Initially it is determined a food source (x_i) for each working bee, usually randomly or partially randomly with a heuristic component. At the phase of working bees, each one will explore a solution in the neighbourhood (v_i) of its food supply, and if it yields a superior performance than the current food source, the new one should replace that food source. Next occurs the phase of opportunistic bees, which wait for the performance of each food source to determine which food source they will explore. This means that the opportunistic bees wait in the colony for the information provided by the worker bees and probabilistically selects a food source. Once it chose a food source, the opportunist bee will explore a solution in its neighbourhood and if a solutions with a better performance than the current source of food, then it replaces it. The phase of scout bees, occurs once a food source has been abandoned. A food source is abandoned once l iterations did occur without improving the performance. Karaboga & Gorkemli in [15] present values that indicate the number of iterations without improvement that should occur before a food source can be abandoned, knowing that the higher l conduct to a more intensive search while the lower l conduct to a search with higher diversity. Once a food source is abandoned, the working bee will become a scout bee and move to a new food source.

III. NEIGHBOURHOOD STRUCTURES

One parameter common to all meta-heuristics, and even other optimization techniques, is the neighborhood structure. NS manipulate solutions to turn one solution into one another in one basic movement. In SA, the NS determines how a solution is turned into another, in each iteration, while in DABC, the NS determines how worker and outlooker bees move around their food source. Mechanisms that manipulate several solutions, as the crossover mechanism in the Genetic Algorithms, are also NS that cross two parental solutions into a child solution. Mutation operators that manipulate, some or all, of the created solutions, can be understood as part of the NS to introduce diversity to the optimization procedure. There are dozens of simple NS that are adequately broad-scoped to be applied in a vast number of problems, other more advanced NS are often developed to be used in a specific problem.

In the computational study, four NS will be reviewed. NS1, 2-swap or 2-Exchange, is a notorious neighbourhood structure for scheduling problems, it selectes two random operations and swaps their positions [16]. For example a execution sequence of [1,2,3,4,5,6] where the 2nd and 4th operations were randomly selected would become [1,4,3,2,5,6] as demostrated in figure 3. Since TSP is a sequence problem, the NS1 can also be used. NS2, or 2-opt, is a more tradional choice for TSP problems, it selects two paths at random, disconnects and reconnects them in another manner [6]. For example the tour [1,2,3,4,5,6], where the path [1,2] and [4,5] were selected woud become [1,4,3,2,5,6], as demostrated in figure 3. NS3 and NS4 combine NS1 and NS2 into a more ample and diverse NS. NS3 uses NS2 followed by NS1, or does a 2-opt movement followed by 2-swap, NS4 uses NS2 twice, or uses 2-opt twice.

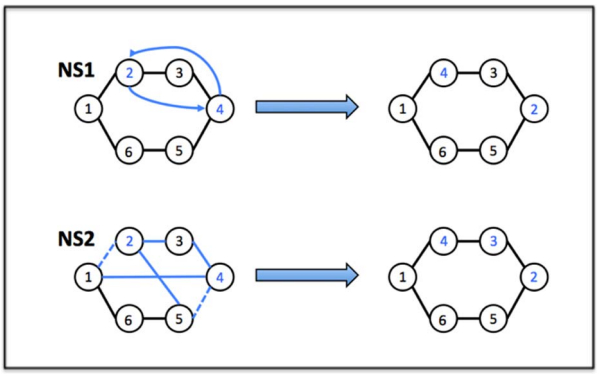

Figure 1. NS1 and NS2

Studies about the performance of NS have examined how the NS influences the behavior of local-search techniques. In. [17], Ahuja et al presents an overview of well-known NS for the Euclidian-TSP problem that demonstrate the need to balance the size of the NS and the available time. In meta-heuristics, the performance of NS has been the topic of much research, with several NS proposed and studied for the best-known problems, however this seems to infer that the NS is more important than the other parameters. Since meta-heuristics are bestowed with

mechanisms to move from the local-optimums, the NS should be presented as another parameter, which interacts with the other parameters to determine the overall balance between the intensity/diversity of the meta-heuristic search procedure.

IV. Results Analysis

SA and DABC, with the neighborhood structures (NS) previously defined, were implemented in c in Microsoft Visual Studio 2012. The computational tests were performed on a MacBook Pro with a 3GHz Intel® Core i7 processor, 16GB of 1600MHz RAM and Windows 10 64-bit.

Both meta-heuristics were tested with all the implemented NS in 5 academic benchmark instances of the Symmetric Euclidian TSP [6], KroA100, KroB100, KroC100, KroD100 and KroE100, with 100 nodes each, which correspond to $9.33262_{E}157$ possible solutions. All the problems are available in the TSPLIB, which also presents the optimal solutions for each problem, for KroA100 is 21282, for KroB100 is 22141, for KroC100 is 20749, for KroD100 is 21294 and for KroE100 is 22068, which allows a better comparison of the results of both meta-heuristics with all the NS across the 5 instances, since solutions can be normalized around the optimal solution in each instance of the problem.

A. Parameterization

Meta-heuristics parameterization has concentrated the attention of researchers, as an appropriate definition of the parameters can improve the technique performance in a specific application. However the parameterization can be time intensive, often harder than the implementation of the meta-heuristics [18]. Hunter et al. [19] presents a formal definition of the parameterization, as: "*Given a parameterised target algorithm A, a set of problem instances I and a performance metric c, the goal is to find parameter settings of A that optimize c in I*".

In this paper the relation between the NS and the rest of the parameters will be studied, instead of the overall performance of the meta-heuristics. Moreover each meta-heuristics will be tested with parameters that find solutions in less that 1 and 10 seconds, to examine how the performance of each NS varies with the increase of the computational time, to examine the evolution of the solutions. Since all the defined NS have distinctive levels of intensity/diversity, it is expected that the evolution of the solutions will depend on the characteristics of each of the NS.

In SA the parameterization process is simpler than in other meta-Heuristics, since with enough time SA performs well, even without a detailed parameterization procedure [20, 21]. However, with limited computational time the parameters need to be precisely determined, in order to conduct an efficient search of the solution space. SA parameters include the Initial Temperature, the Cooling Ratio, the Epoch Length, the Neighborhood Structure (NS) and the Stoppage Condition. Park & Kim in [20] examined the impact of each parameter in SA and presented some parameterization rules: the initial temperature (T_i) should accept almost all movements, which means the initial acceptance probability (P_i) should be near 1; the cooling ratio (α) should result in a slow decrease of the temperature, which means that the cooling ration with a proportional cooling function should be between [0.80;0.90]; finally, the epoch length (L) should be related to the instance size, which means that it should be close to the dimension of the problem or the number of neighborhood solutions.

Since this paper will study the performance of each NS bellow 1 and 10 seconds of computational time, the Stoppage Condition will be the number of iterations, while the NS will be the one presented in the previous point. In table I are the rest of the parameters for both 1 and 10 seconds. It is important to notice that the overall performance of SA is not the main focus of this paper; instead it will examine how the performance of each NS varies, under different time limitations. It is expected that when the computational time increases, the NS that provides more diversity will result in a better performance.

In DABC the parameterization is more complex than SA, as the parameters have a tremendous impact in the performance of DABC, more pronounced than in SA, with less parameters. DABC parameters include the Colony Size, the Limit, the Neighborhood Structure (NS) and the Stoppage Condition. Akay & Karaboga [22] examined the impact of each parameter in ABC and presented some parameterization rules: the colony size (L) does not need an increase of the population sizes to solve large optimization problems since this parameter has an enormous impact in the computational time; the limit (l) is related to the colony size, larger colonies can have smaller limits, since the lack of diversity from smaller limits are balanced by the number of individuals in the population. In [15] the authors presented some rules to determine the limit, based on the number of bees and the dimension of the problem. Other parameters considered in some implementations of ABC and DABC, some authors have used a fixed number of scout bees to increase the diversity of the Meta-Heuristic.

Like in SA, the stop criterion will be the number of iterations, since the purpose is to examine the performance of each NS bellow 1 and 10 seconds of computational time. DABC requires more computational time per iteration than SA, so there will be a difference in the number of iterations for SA and DABC. In table I are the rest of the parameters for both 1 and 10 seconds. Once more, the focus on this paper is not the overall performance of DABC; it will examine the performance of each NS under different time limitations and compare how the performance of each Meta-Heuristic is affected.

TABLE I. Parameters of SA and DABC

	Parameter	1s	10s
SA	N. of Iterations	200000	2000000
	Initial Temperature (T_i)	2000	20000
	Cooling Ratio (α)	0.95	0.95
	Epoch Length (L)	500	500
DABC	N. of Iterations	10000	100000
	Colony Size (L)	20	20
	Limit (l)	1000	10000

SA and DABC parameters were determined so that the procedure would conclude in less than 1 second, and then adapted to conclude in less than 10 seconds, with an increased of the number of iterations and the initial temperature in SA and the limit in DABC. Some parameters were used for both 1 and 10 seconds and did not impact the run-time of either Meta-Heuristic.

B. Computational Results

Both meta-heuristics were run five times for each problem and achieved computational results close to the optimum. In figure 4 the top row represents the computational result of SA, and the bottom row represents the computational results of DABC, for 1 and 10 seconds, with each neighborhood structure results represented in a different color.

In SA with 1 second, NS2 found the best solutions (21459, 22552, 20872, 21508, 22122), close to the optimal solutions, represented in blue. NS4 found the second best solutions (24070, 24991, 24140, 24321, 24794) and NS3 (25038, 25730, 25526, 24587, 25662) the third best. NS3 and NS4 solutions are similar, with a small lead in the performance of NS4. NS1 found the worst solutions (25527, 27718, 26510, 26357, 29442) and also appeared to have more variability across the problems, for example, in KroA and KroE. In SA with 10 seconds, NS2 found the best solutions (21353, 22284, 20812, 21398, 22216). NS4 found the second best solutions, in KroB, KroC, KroD and KroE, (21601, 22535, 20905, 21700, 22717), NS3 the third best solutions in all problems but KroA (21357, 22888, 21007, 22128, 23078) and NS1 the worst solutions (24031, 24566, 22888, 22814, 24311) in all problems.

All the NS performed better with the 10 seconds limit and, other than NS2 that already found near optimal solutions with the 1 second limit, all NS appeared to improve identically. NS1 and NS2 intense searches did not appear to be less competitive than either NS3 or NS4, with their more diverse searches, even with the increase of the available computational time.

In DABC with 1 second, NS2 found the best solutions, (22225, 23650, 21981, 22552, 23161), but unlike SA the difference to the other NS is more pronounced. NS4 found the second best solution (31739, 32341, 31690, 32302, 31868), followed by NS3 (34693, 35482, 34406, 35290, 36253) and NS1 (36383, 35157, 37416, 36775, 36202). In DABC with 10 seconds, NS2 found the best solutions (21504, 22544, 21089, 21938, 22654) and NS4 found the second (22586, 24332, 22335, 22712, 23772), improving the solutions until they were almost optimal. NS3 found the third best solutions (23989, 24814, 23624, 23793, 24834) and NS1 found the worst solutions (31445, 32701, 33473, 32953, 34060), but this time the difference in performance between NS1 and the other NS is noticeable, this is evident in figure 1 that shows that NS1 solutions are almost 50% worse than the other NS.

Once more, all the NS improved their performance in the 10 seconds limit, nevertheless, in opposition to the SA, there appears to be a difference between the improvement of the NS1 and NS2 more intense searches and the NS3 or NS4 more diverse searches. NS3 and NS4 appeared to improve much more that NS2 and specially NS1, which barely improved and had the worst result from the computational test.

Overall SA performed better than the DABC, which is much more noticeable in the 1 second experiment. In the 10 seconds experiment the performance of both meta-heuristics improved, but there appears to be a difference in improvement of NS1 and NS2, and NS3 and NS4 in DABC, where NS1 and NS2 did not improve as the NS with more diverse searches.

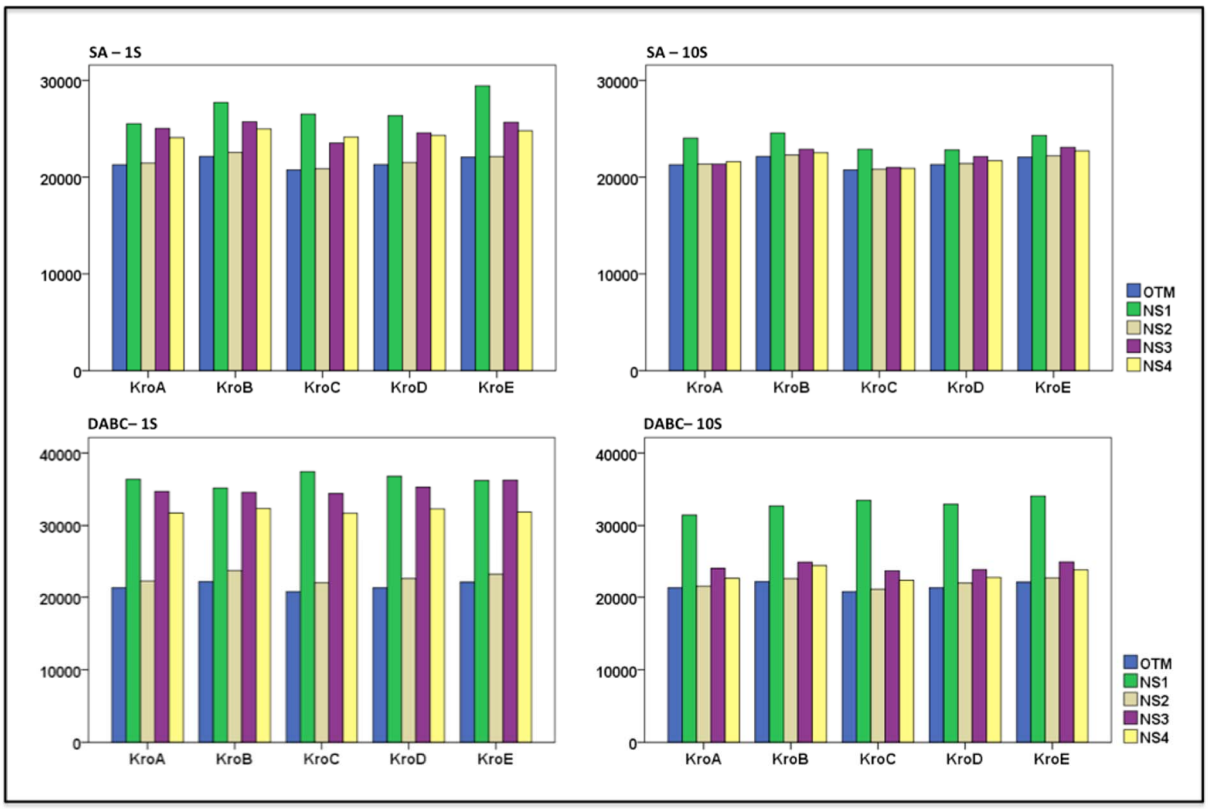

Figure 2. Results of SA and DABC

C. Statistical Results

In the computational trials, there appears to be a difference in the evolution in the NS performance, however to examine this difference in more detail the computational results need to be normalized to compare the results across all instances. Since the optimal solutions KroA, KroB, KroC, KroD and KroE, are available in TSPLIB, this document will use mean percent devotion from the optimal solution, which is the best know metric to compare different meta-heuristics [23]. Once the results are normalized, the improvement of each of the NS performance with the available time will be examined.

SA normalized results are presented in figure 3. In KroA, the deviations for 1 second are 19.946%, 0.832%, 17.649% and 13.100% for NS1, NS2, NS3 and NS4, for 10 seconds the deviations are 12.917%, 0.334%, 0.352% and 1.499%. In KroB, the deviations for 1 second are 25.189%, 1.856%, 16,210% and 12.872%, for 10 seconds the deviations are 10.953%, 0.646%, 3.374% and 1.780%. In KroC, the deviations for 1 second are 27.765%, 0.593%, 23.023% and 16.343%, for 10 the deviations are 10.309%, 0.304%, 1.243% and 0.752%. In KroD, the deviations for 1 second are 23.777%, 1.005%, 15.464% and 14.215%, for 10 the deviations are 7.138%, 0.488%, 3.917% and 1.907%. In KroE, the deviations for 1 second are 33.415%, 0.245%, 16.286% and 12.353%, and for 10 seconds 10.164%, 0.671%, 4.577% and 2.941%.

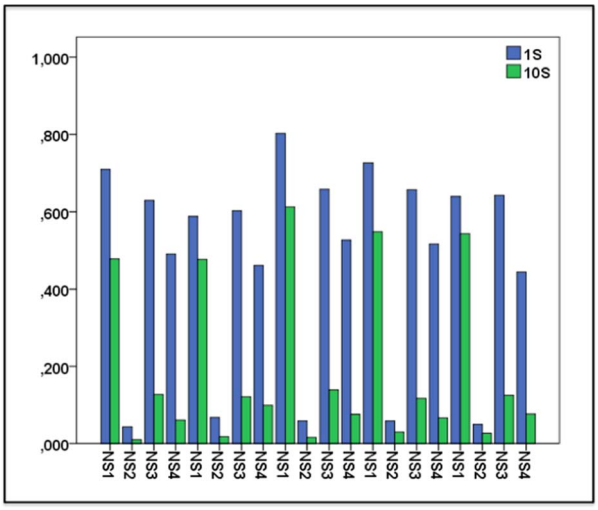

Figure 3. Results from SA with 1 and 10 Seconds

DABC results are presented in figure 6. In KroA, the deviations for 1 second are 70.957%, 4.431%, 63.016% and 49.135%, for 10 seconds the deviations are 47.754%, 1.043%, 12.720% and 6.127%. In KroB, the deviations for 1 second are 58,787%, 6.815%, 60.255% and 46.068%, for 10 seconds the deviations are 47.694%, 1.820%, 12.073% and 9.896%. In KroC, the deviations for 1 second are 80.327%, 5.908%, 65.820% and 52.730%, for 10 the deviations are 61.323%, 1.639%, 13.856% and 7.644%. In KroD, the deviations for 1 second are 72.701%, 5.908%, 65.727% and 51.695%, for 10 the deviations are 54.753%, 3.024%, 11.736% and 6.659%. In KroE, the deviations for 1 second are 64.047%, 4.953%, 64.279% and 44.408%, for 10 seconds the deviations improve to 54.341%, 2.655%, 12.534% and 7.722%.

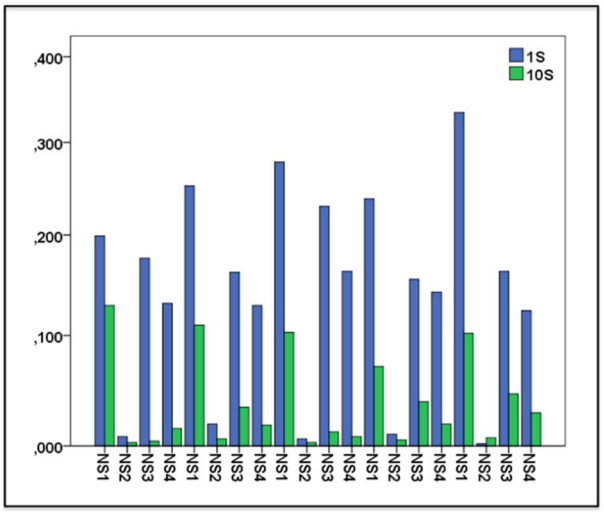

Figure 4. Results from DABC with 1 and 10 Seconds

Overall, the improvement of NS1 and NS2 appeared less pronounced than the improvements of NS3 and NS4. In SA the difference in the improvement of NS1 and NS2, and NS3 and NS4 is not as prominent as in DABC. In SA, NS1 and NS2 had a mean improvement of 8.069%, and NS3 and NS4 an improvement of 13.517%, while in DABC, the NS1 and NS2 improved 9.881% and NS3 and NS4 improved 46.216%. However, to compare the differences between the mean in improvement of NS1/2 and NS3/4 in SA and DABC, require the Wilcoxon Mann-Whitney test, with the hypothesis:

$H_0: \eta_{NS1/2} = \eta_{NS3/4}$

$H_1: \eta_{NS1/2} \neq \eta_{NS3/4}$

The Wilcoxon test results, in table II, demonstrate that there are statistical differences between the mean improvement of NS1/2 and NS3/4 in DABC (0.000) but not in SA (0.239).

TABLE II. WILCOXON MANN-WHITNEY TEST

Parameter	SA	DABC
Mann-Whitney U	34.000	0.000
Wilcoxon W	89.000	55.000
Z	-1.210	-3.780
Asymp. Sig. (2-tailed)	0.226	0.000
Exact Sig. [2*(1-tailed Sig.)]	0.247	0.000
Exact Sig. (2-tailed)	0.239	0.000
Exact Sig. (1-tailed)	0.120	0.000

SA less impactful parameterization procedure can explain the lack of variance in the performance of NS1/2 and NS3/4 with the increase of the available time. Park & Kim in [25] state that SA do not require a detailed parameterization procedure in order to perform well, if there is enough computational time, which explain the SA results in the 10s tests. In opposition, in DABC the results demonstrate that NS3/4 improved more than NS1/NS2 with the increase of the available time, since DABC requires a detail definition of the meta-heuristics parameters.

V. CONCLUSIONS

In this paper we demonstrated that the behavior of NS varies in accordance with the rest of the parameters of the meta-heuristics. For that purpose, four simple NS, with distinctive levels of intensity/diversity, were studied, NS1 or 2-swap, which swaps the position of two destinations of the tour, NS2 or 2-opt, which disconnects and reconnects two paths between destinations, and NS3 and NS4 mix NS1 and NS2. In the computational study, it was demonstrated that the performance of each NS depends on the rest of the parameters in DABC, in the case, the duration of the procedure. In all the conducted exams, the improvement of NS1/2 appeared less pronounced than the improvements of NS3/4. The Wilcoxon Mann-Whitney test for DABC showed that NS1/2 improved the performance of the meta-heuristic less than NS3/4 (0.239), which cannot be demonstrated in the SA test (0.000).

The computational results appear to demonstrate that NS should be treated as another parameter, which contribute to the overall balance between the intensity/diversity of the meta-heuristics as much as the other parameters. Studies that demonstrate the benefits of one NS, should take into account how that NS interacts with the other parameters. One important consideration is how the NS would perform under other time constraints. SA does not require an exhaustive parameterization procedure, since the performance of the meta-heuristics scales with the available computational time, while DABC need an attentive determination of the parameters to perform well. That distinction, between the two meta-heuristics, can be examined in the computational study, where SA result to scale uniformly with the computational time, independently of the values of the parameters, which does not appear to happen in DABC.

Further work will focus on additional computational tests with other NS and more time intervals, which should allow a more in-depth revision of the evolution of the behavior of both meta-heuristics. Other, more advanced NS, can also be used, to examine their behavior with different time constrains.

ACKNOWLEDGMENT

This work is supported by FEDER Funds through the "Programa Operacional Factores de Competitividade - COMPETE" program and by National Funds through FCT "Fundação para a Ciência e a Tecnologia" under the project: FCOMP-01-0124-FEDER-PEst-OE/EEI/UI0760/2011, PEst-OE/EEI/UI0760/2014, and PEst2015-2020.

REFERENCES

[1] C. S. Jeong, and M. H. Kim, "*Fast parallel simulated annealing for traveling salesman problem on SIMD machines with linear interconnections*," Parallel Computing, 1991, 17(2-3), 221-228.

[2] G. A. Croes, ''*A method for solving traveling salesman problems*,'' Operations Research, 1958, 6(6), 791-812.

[3] J. C. Creput and A. Koukam, "*A Memetic neural network for the Euclidean traveling salesman problem*," Neurocomputing, 2009, 72(4-6), 1250-1264.

[4] M. Dorigo and L. M. Gambardella, "*Ant colonies for the traveling salesman problem*," Biosystems, 1997, 43, 73-81.

[5] S. Lin, ''*Computer solutions of the traveling salesman problem*," Bell System Technical Journal, 1965, 44(10), 2245-2269.

[6] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu and Q. X. Wang, "*Particle swarm optimization-based algorithms for TSP and generalized TSP*," Information Processing Letters, 2007, 103(5), 169-176.

[7] Y. Chen and P. Zhang, "*Optimized annealing of traveling salesman problem from the nth-nearest-neighbor distribution*," Physica: A Statistical and Theoretical Physics, 2006, 371(2), 627-632.

[8] T. Ghosh, S. Sengupta, M. Chattopadhyay, and P. K. Dan, "*Meta-heuristics in cellular manufacturing: A state-of-the-art review*," International Journal of Industrial Engineering Computations, 2011, 2, 87-122.

[9] H. I. Osman and J. P. Kelly, "*Meta-Heuristics: An Overview. Meta-Heuristics Theory and Applications*," Kluwer Academic Publishers, 1996.

[10] I. Pereira, A. Madureira, P.M. Oliveira and A. Abraham, "*Tuning Meta-heuristics Using Multi-agent Learning in a Scheduling System*," LNCS Transactions on Computational Science, Springer Verlag, 2013.

[11] M. L. Pinedo, "*Scheduling Theory, Algorithms, and Systems (Fourth Edition)*," Springer, 2012.

[12] L. R. Varela and R. A. Ribeiro, "*Evaluation of Simulated Annealing to solve fuzzy optimization problems*," Journal of Intelligent & Fuzzy Systems, 2003, 14(2), 59-71.

[13] D. Karaboga, "*An Idea Based On Honey Bee Swarm For Numerical Optimization*," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[14] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. M. Zaidi, "*The Bees Algorithm. Technical Note*," Cardiff University, 2005.

[15] D. Karaboga and B. Gorkemli, "*A Combinatorial Artificial Bee Colony Algorithm for Traveling Salesman Problem*," Proc. of the International Symposium on Innovations in Intelligent Systems and Applications (INISTA), 2011, 50-53.

[16] M. Sevkli and M. Emin Aydin, "*Variable Neighbourhood Search for Job Shop Scheduling Problems*," Journal of Software, 2006, 1(2), 34-39.

[17] R. K. Ahuja, O. Ergun, J. B. Orlin and A. P. Punnen, "*A Survey of Very Large-Scale Neighborhood Search Techniques*," Discrete Applied Mathematics, 2002, 123(1-3), 75-102.

[18] B. Adenso-Diaz and M. Laguna, "*Fine-Tuning of Algorithms using Fractional Experimental Design and Local Search*," Operations Research, 2009, 54(1), 99-114.

[19] F. Hutter, H. Hoos, K. Leyton-Brown and K. P. Murphy, "*An Experimental Investigation of Model-Based Parameters Optimisation: SPO and Beyond*," Proc. of the 11th Annual Conference on Genetic and Evolutionary Computation, 2009, 271-278.

[20] M. W. Park and Y. D. Kim, "*A Systematic Procedure for Setting Parameters in Simulated Annealing Algorithm*," Computers & Operations Reasearch, 1998, 25(3), 207-217.

[21] S. Anily and A. Federgruen, "*Simulated Annealing Methods with General Acceptance Probabilities*," Jounal of Applied Probability, 1987, 24, 657-667.

[22] B. Akay and D. Karaboga, "*Parameter Tuning for the Artificial Bee Colony Algorithm*, " Computational Collective Intelligence, Semantic Web, Social Network and Multiagent Systems, Lecture Notes in Computer Science, 2009, 5796, 608-619.

[23] J. Silberholz and B. Golden, "*Comparison of Metaheuristics*", in Handbook of Metaheuristics, M. Gendreau and J. Potvin, Eds. Springer, 2010, 625-640.