Knowledge Sharing Through Agent Migration with Multi-Population Cultural Algorithm

Andrew W. Hlynka and Ziad Kobti

School of Computer Science, University of Windsor, Windsor, ON, Canada N9B 3P4
hlynkaa@uwindsor.ca and kobti@uwindsor.ca

Abstract

This study presents a new method for knowledge transfer in Multi-Population Cultural Algorithms (MPCA) through agent migration. This agent-based algorithm involves having individual agents using one of multiple pre-defined knowledge algorithms to determine behavior, and using the success of it and other agents to decide on which knowledge algorithms to use next. Two or more subpopulations with their own knowledge algorithm are created. The agents work in the same environment by only communicating with agents within their own subpopulation, and with two global belief spaces monitoring the effectiveness of each subpopulation. Agents transfer between the subpopulations regularly to further improve individual success. We use the "cone's world" problem as testbed. Experimental results reveal the impact of individual knowledge transfer on the target subpopulation's belief space.

1. Introduction

Cultural Algorithms (CA) have been in use for many years. Originally created as an extension of Genetic Algorithms (Reynolds, 1994), they have since grown to incorporate other evolutionary algorithmic structures (Lin, Chen and Lin, 2009; Guo and Liu, 2011) as a basis to their framework. The underlying idea is that evolutionary algorithms generally focus on evolving a number of individuals over time using previous iterations to achieve more successful solutions to various problems, whereas CA add the idea of cultural evolution in order to increase the speed and efficiency of the algorithm. Artificial "belief spaces" help store knowledge gained during the runtime of the algorithm and also influences the guidance of individuals in addition to genetic evolution (Reynolds, 1994). The ways belief spaces are used and the way knowledge is stored within them can vary greatly, but five distinct types of knowledge have been accepted: normative (ranges of better choices), situational (successful and unsuccessful instances), topographic (spatial patterns), historic (past occurrences), and domain (information and relationships regarding domain objects) (Reynolds and Saleem, 2003). CAs have been helpful for various optimization problems and real-world applications (Reynolds, Rychtyckyj, Ostrowshi and Schleis, 2003; Kobti, *et al*., 2006; Diagalakis and Margaritis, 2002; Reynolds and Chung, 1996; Reynolds and Sverdlik, 1994) as well as multi-agent simulations (Reynolds, Kobti and Kohler, 2003; Nakhwal, *et al*., 2010; Alami and Imrani, 2005).

The definition of CA has been further expanded into a new type of algorithm called Multi-Population Cultural Algorithms (MPCA) (Guo, Liu and Cheng, 2012; Guo, Cheng, Cao and Lin, 2010). The general idea is to have a number of sub-populations, each using cultural and genetic evolution to obtain better solutions. Existing implementations have tried various ideas, such as using a global "master" to look over the isolated populations and perform steps to create new populations by taking the most successful individuals from the previous instance (Diagalakis and Margantis, 2002; da Silva and de Oliveira, 2009). Other implementations use a global belief space to extract knowledge from more successful individuals of each subpopulation to share with all individuals, where the individual populations are in charge of their own evolution while being influenced by this knowledge (Guo, Liu and Cheng, 2012; Raessi and Kobti, 2012). As optimization problems are generally used for testing purposes, existing work on MPCA focuses on using a belief space global to all subpopulations in order to communicate and share knowledge quickly. However, work has not been done to discuss the potential outcome of MPCA where individual agents themselves influence the evolution of other sub-populations.

Considering existing work on MPCA, a "Transfer-Agent"-based MPCA (TAMPCA) is suggested. Unlike previous work, individual agents are not taken from each sub-population to update a global belief space itself, but to swap places with each other, such that each subpopulation

and a foreign individual are introduced to each other. This way, knowledge and the current solution the transferred agent knows are communicated with other agents who may not have had access to this information previously. This can potentially alter the evolution process of the sub-population. To best visualize this, sub-populations are each given an unique algorithm defining a singular type of knowledge to use during their evolution process. When transfer-agents are introduced, their types of knowledge used are communicated (as well as their current success), in order to influence other agents accordingly. Each sub-population contains its own global belief space shared with all agents within its respective sub-population, which is used to monitor knowledge types being used and to help influence knowledge adaption to the agents. This proposed algorithm demonstrates a more realistic approach to solve problems, and more importantly, to create more realistic social-based simulations.

The rest of this paper describes the details of the TAMPCA. The details of the problem environment used to test this algorithm are given, followed by the results from the experiments, and finally some concluding remarks.

2. Transfer-Agent Multi-Population Cultural Algorithm

The structure of this TAMPCA is simplified to best examine its properties and reasons of its results. All agents in the algorithm are divided into two populations, each containing an equal number of agents. Note that more than two populations may be considered in other examples. Each agent contains its own knowledge within itself (it own individual "belief space"), collected based on which knowledge type is being used. Each population contains its own global belief space that keeps track of values related to each type of knowledge used in the population along with its effectiveness. A global master program keeps track of all the agents within the populations, and transfers "n" agents between each population when required. The algorithms required to run each defined type of knowledge is stored explicitly within each agent, but a variable defines which knowledge algorithm each individual agent uses during the experiment.

This implementation of the TAMPCA gives the two sub-populations each a specific type of knowledge defined from the five main types of knowledge in CA: topographic and situational knowledge. Topographical knowledge is defined by giving each agent a range of values in the space that are defined as a good potential space for better solutions. A range is initialized as the largest possible range the problem allows at the start. Agents use their own experience and intersections with other ranges saved by other nearby agents in order to further focus their ranges to ideal areas. If a concise range still returns poor results (that is, the poorest known possible result for the problem at hand), then the range is initialized again to be as large as possible before using personal experience and neighbor's ranges again to evolve the range. Situational knowledge has each agent look directly at nearby agents and compare fitness values from them with itself. If a nearby agent is performing better than the current performance of the agent, then the agent will move towards the better agent, otherwise the agent will move randomly for other nearby better solutions. Note that many other types of knowledge algorithms can easily replace these for other implementations.

Since agent communication is important for this implementation in order to influence other agents, an agent communicates with other agents within a certain range of itself in order to further improve its knowledge. The agents do not communicate with agents outside their sub-population, however: the two sub-populations run within the same environment, yet do not influence each other (i.e. the individual agents check the subpopulation each nearby agent belongs to before communicating). This is to enhance the notion of reviewing the effects new knowledge may have on a population when agent-immigration occurs.

Every n iterations (in our case, n = 10) of the experiment, we update the belief space for each sub-population, and also update which knowledge type each agent should use. This is done through a master program, which takes the knowledge type and instance fitness value at that moment from each individual agent, passing it to the global belief spaces for use in each population. Each global belief space contains two real numerical values, each representing the likelihood of each knowledge type being the better for agents to use in future iterations. The two values are between 0 and 1, where both values together add up to 1. When updated, the global belief space takes a numerical value from the fitness values of all agents within a sub-population, and saves their sum representing their respective knowledge types they represent. When all agents have updated the belief space, there should be a large sum for both types of knowledge, representing their success and adaption in the population where greater values suggesting greater adaption or success. The two sums are then divided by their total sum, creating two new values between 0 and 1 that represent the influence ratio of each knowledge type. If all agents within a sub-population are using the same type of knowledge (as they are at the beginning of the algorithm), only the one type of knowledge adds to its numerical influence value in the global belief space, consequently returning an influence value of 1. But if other agents immigrate to this population with different knowledge types, the belief space will be updated with a greater change respective to the fitness value of that agent in comparison to other agents in the population.

During the transfer of agents in TAMPCA, a master program randomly chooses a number of agents from each sub-population to be transferred without any notion as to whether or not the swapping agents were ideal agents with

better fitness in their original populations. Every iteration where the belief space for each population is updated, the agents and the corresponding knowledge type they use are also updated. For the purpose of further improving realism in this algorithm for use in social system models, three potential methods to decide the new knowledge function used are defined, one of which will be randomly chosen when required. First is that if no change occurs, the agent continues the knowledge type that it had been previously using (this represents how many individuals are resistant to any change over time). Second is that each agent will look to the global belief space, a theoretical sub-conscious that helps the agent converge to use a type of knowledge that represents the majority of the population. A "roulette wheel" approach helps decide randomly (with weighted values as dynamically chosen by the two ratio values in the belief space) which single knowledge type is to be used. The third is a similar method to the second, but each agent only looks towards agents close enough for communication to create similar weighted values for a "roulette wheel" approach for choosing a new knowledge type to use. These three different methods for choosing which knowledge to use can be altered in future implementations not to be equally likely to occur, but to be more or less likely based on time passed or past choices.

3. The Problem Domain: Cone's World

In order to test the TAMPCA, a common but simple optimization problem is used in order to visualize effects and performance during this algorithm. The "Cone's World" problem refers to an environment that is made up of several cones or hills, each with varying heights. The algorithm is used to search for a maximum height within the environment. Knowledge specific to the problem domain ("domain knowledge"), such as the fact that hills increase in height towards a local peak, are also utilized in the predefined knowledge algorithms in order to further improve the results to appear more relevant.

Our "cone's world" is a static environment defined as a 100x100 matrix of integers, consisting of numbers from 0 (ground level, worst) to 20 (highest peak, best). These integer numbers represent cone (hill) patterns, where ground levels increase slowly (one unit at a time) towards a hill's peak, before dropping again. 20 cones (of varying heights, only one of which is at maximum height of 20, some overlapping each other) are randomly placed to create the map that is used in this experiment. Each sub-population is given 100 agents, each randomly placed around the environment. The TAMPCA will run for 100 iterations without agent-transfer to see how each population converge to their own local maximum values. Transfer occurs every 100 iterations, such that "n" agents (n being tested for values of 1, 5, 10, 25, and 50) are transferred between the sub-

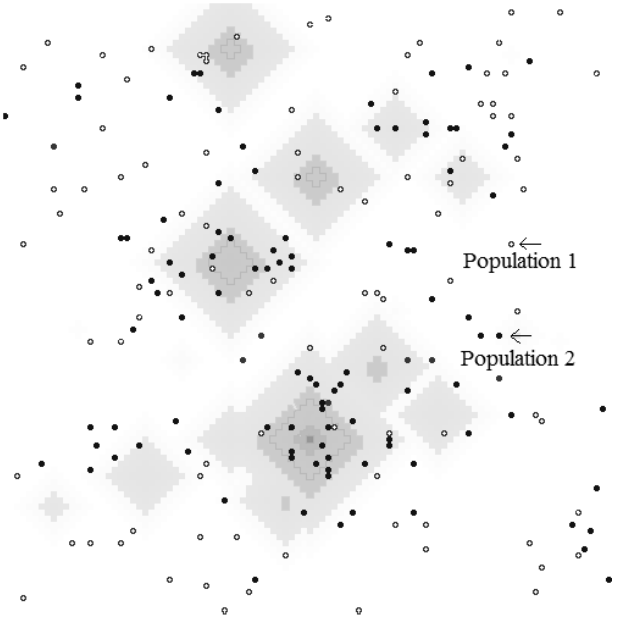

Figure 1 - Domain Map at Beginning of Experiment

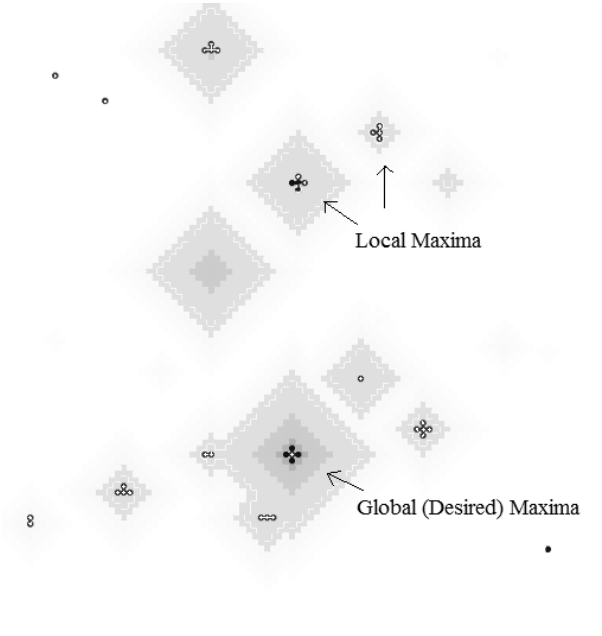

Figure 2 - Domain Map at End of Experiment

populations. This is repeated to examine the effects of continued transfer for 500 iterations as each population tends to converge again before these transfers repeat. Ultimately, results may be expected to show a final convergence where one successful knowledge type becomes dominant over the other. Figure 1 shows a visual representation of the problem domain in this experiment, with an indicator of the highest peak (the darker spots), and Figure 2 shows the

same area by the end of the experiment test, where individual agents have converged to peaks of varying heights.

The results of this experiment will focus on three things: the influence weight values given to each knowledge type in the belief spaces in the populations, the number of individual agents in each population which actually use each type of knowledge (which may be closely related to the belief space weight values), and the fitness values of the populations at each iteration of the algorithm (taken as the average height of all the agents within each population). The results are analyzed to see the effects each population had on each other during the transfer of agents, and whether or not improvement occurs, and if so, to what degree does it occur.

4. Experiments and Analysis

To begin, we run tests with no transfer of agents, allowing a better comparison for future tests where transfer of agents occurs. Figure 3 shows the results of the algorithm without any transfer, from the population using only topographic knowledge, taken from an average of multiple tests. Figure 4 shows the results of the algorithm without any transfer, from the population using only situational knowledge, taken from an average of multiple tests. The average of all the tests is plotted in each graph, as well as the best and worst examples from the tests taken (to give an indication of the range difference in the results). The average heights can reach a maximum of 20, the maximum height in this environment.

As the graphs suggest, situational knowledge is more effective in this problem domain. Situational knowledge converges quickly to a desirable solution, with little difference in the tests. Topographic knowledge was more erratic in its tests, taking longer to converge, and reaching values comparable to situational knowledge in only the best cases. This is a good testing ground for this particular experiment, as knowledge types with varying results is desired to produce more interesting results when transfer of agents occurs. One might predict that situational knowledge will become dominant over time, and that use of topographic knowledge will eventually die out. Also, results after multiple transfers might be expected to reach up to the level achieved by the better of the two knowledge types, in this case, situational knowledge. Of course, knowledge type use in these sub-populations is constant when no transfer of agents occur, and so data regarding that is not recorded here. Some researchers in the field of optimization might cringe at the sight of less than perfect results in a long period, but remember that the entire population is being evaluated here: while some agents do indeed find the ideal solution (the highest peak), others get stuck at lower peaks,

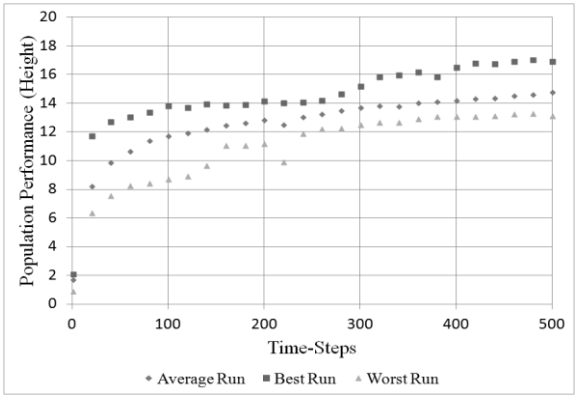

Figure 3 - Population 1 (Topographic) with no agent transfer

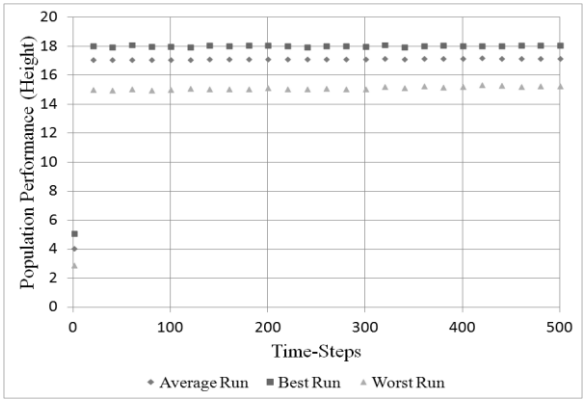

Figure 4 - Population 2 (Situational) with no agent transfer

and are too far from ideal agents to communicate this difference. These experiments are for the success of the population when agents work together, not simply to find the best solutions in the quickest time.

Having results to compare with, the experiment is repeated with a transfer of 1 agent from each sub-population, transfer occurring every 100 iterations during the timeline. For this test, the average run of the algorithm gives an interesting conclusion: Figure 5 shows that the transfer of agents to population 2 results in a slight change to the population's success, despite that the knowledge-agent being migrated is the weaker of the two knowledge types. Population 1 shows a faster increase than without this transfer from the second population. Also, the fair success of sub-population 2's agent being transferred ensures that situation knowledge becomes more popular in population 1, although the original topographic knowledge type is still used more often.

Figure 6 shows results of several tests of the experiment, examining the performance of each population for transfers of 1 agent, 5 agents, 10 agents, 25 agents, and 50 agents.

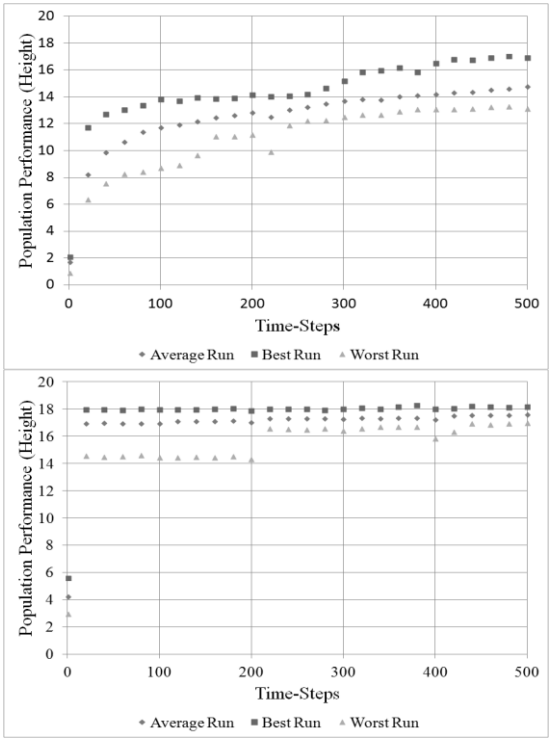

Figure 5 - Population 1 (top) & 2 with transfer of one agent

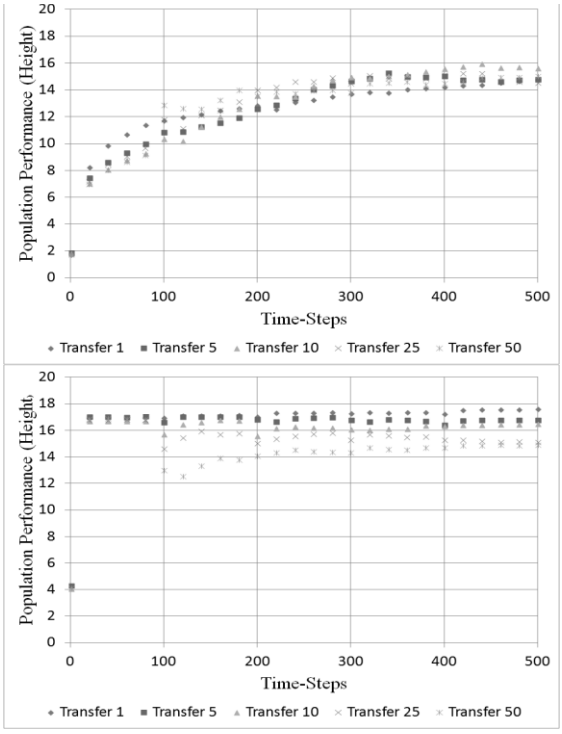

Figure 6 - Population 1 (top) & 2 with transfer of agent(s)

As Figure 6 shows, performance seems to vary based on the number of agents being transferred. With 1 agent transferred every 100 iterations, better improvement does occur in population 1, and even occurs slightly in population 2. During this, the original knowledge types are still dominant in their respective groups. With 5 agents being transferred, population 1 reaches convergence more quickly, but both populations 1 and 2 seem to hover at a point no larger than the best that either reached originally, and even some decrease in performance occurs when knowledge influences come closer together in population 1. With 10, 25 and 50 agents being transferred, convergence in population 1 increases rapidly to meet the level of success of population 2, while population 2 actually decreases each time by a bit, taking in bad influence. Population 1's knowledge influence accepts situational knowledge fairly quickly, whereas population 2 does not have topographic knowledge overcome it, although neither fully accepts one over the other in these last three experiments. However, as the knowledge influence values become similar in both sub-populations, success in both becomes similar as well. The combined success of both populations increases as they both converge to similar success values, but the two populations themselves do not increase together.

5. Conclusion

In these experiments, two sub-populations using different types of knowledge were used in a multi-population cultural algorithm, and then individual agents from both were transferred between each other to examine results. This was tested for transferring 1%, 5%, 10%, 25% and 50% of the populations. Overall performance was evaluated as well as effects on knowledge acceptance.

The test where 1% of each population was transferred appears to be the only situation where increase in success in both sub-populations occurs. Other tests with larger samples from each population transferred offered poorer results, ending with success equal to (or even less than) what the sub-population had originally started with. The weaker of the two knowledge types lost most of its influence in both populations with larger samples of transferred agents, and the population starting with this weaker knowledge would consistently see improvement by accepting the other knowledge. The second population, however, saw some decrease as it was tainted by a lesser knowledge, but not by as large a degree.

The interesting result comes from the test with 1% of the populations transferred, as an increase occurred in performance. The only other noticeable difference is that this test also kept the original knowledge types for each sub-population dominant within their groups, only letting new immigrant individuals have a slight influence within the population's behavior. This might be due to the nature of the solutions found by each sub-population: as the knowledge types were originally different, the solutions

were as well. Thus, agents randomly chosen to be transferred may be the more successful agents in its original population, and their knowledge is lost after the transfer. However, the first test claims that the immigrant individual may offer knowledge or a solution that is better than those existing without taking too much away from the previous population, and through slow evolution the other agents decide whether or not to move towards these new solutions and knowledge methods, or stay at their original place.

The results from this experiment may suggest that social systems, in general, benefit from slow improvement and evolution as opposed to quick and sudden change. Such a bold conclusion isn't especially clear, but this experiment does show that algorithms that are not guaranteed optimal success every time can benefit from other algorithms that happen to perform better during that run. A small transfer of agents among multiple populations may lead to a slight improvement with better consistency than one alone. If true, such ideas would be beneficial towards more complex social simulations to improve evolution, and also towards potential new methods in optimization solutions where multiple appropriate algorithms exist (although time was not a considered factor in these experiments).These results would need to be proven further with testing new knowledge algorithms to compare, and testing within new domains of varying definitions and goals.

Also worth mentioning is the possibility of this method of testing being used to find ideal combinations of algorithms to perform well under certain problems. In this experiment, two defined algorithms were used by individual agents, but not at the same time by any individual. The algorithm performed poorly when the influences of each knowledge type were changed too drastically, but subtle changes may lead to a better combination to use from the start of the algorithm to provide better results. If potential solutions to a problem could be broken up into smaller algorithms that could be assigned to individual agents, an algorithm such as TAMPCA could be used to quickly find a good estimate on how they should best be distributed among the agents for efficiency and accuracy. This is only a hypothesis, however, and requires testing to prove any merit.

Future work will involve testing these experiments on new problem domains with new knowledge types, including testing the effectiveness with using TAMPCA to simulate these situations. This may further lead to studies in behavior in natural societies, and help development in studies regarding complex social systems through artificial modeling.

Acknowledgments

This work was partially supported by a research grant from NSERC Discovery.

References

Reynolds, R. G. 1994. An Introduction To Cultural Algorithms. Wayne State University, Detroit, MI.

Lin, C. J. and Chen, C. H. and Lin, C. T. 2009. A Hybrid of Cooperative Particle Swarm Optimization and Cultural Algorithm for Neural Fuzzy Networks and Its Prediction Applications, In *IEEE Transactions on Systems, Man, and Cybernetics* 55-68, IEEE.

Reynolds, R. G., and Saleem, S. 2003. The Impact of Environmental Dynamics on Cultural Emergence. Oxford University Press.

Reynolds, R. G., and Kobti, Z., and Kohler, T. 2003. Robustness in Coupled Human/Natural Systems in the Northern Prehispanic Southwest, Sante Fe Institute/Oxford Studies in the Sciences of Complexity.

Reynolds, R. G., and Rychitychj, N., and Ostrowshi, D. 2003. Using Cultural Algorithms in Industry, In *Proceedings of the Swarm Intelligence Symposium* 187-192, IEEE.

Kobti, Z., and Snowdon, A. W., and Rahaman, S. and Dunlop, T., and Kent, R. D. 2006. A Cultural Algorithm to Guide Driver Learning in Applying Child Vehicle Safety Restraint, In *IEEE Congress on Evolutionary Computation* 1111-1118, IEEE.

Guo, Y., and Liu, D.., and Cheng, J. 2012. Multi-population Cooperative Cultural Algorithms. Springer-Verlag, Berlin Hiedelberg.

Guo, Y., and Cheng, J., and Cao, Y., and Lin, Y. 2010. A novel multi-population cultural algorithm adopting knowledge migration. Springer-Verlag, Berlin, Hiedelberg.

Diagalakis, J. G., and Margaritis, K. G. 2002. A multipopulation cultural algorithm for the electrical generator scheduling problem. In *Mathematics and Cimputers in Simulation* 293-301. Elsvier Science.

Silva, D. J. A., and Oliveira, R. C. L. 2009. A Multipopulation Cultural Algorithm Based on Genetic Algorithm for the MKP. In *GECCO '09 Proceedings of the 11th Annual conference on Genetic and evolutionary computation* 1815-1816, ACM, New York, NY

Kobti, Z., and Raessi, M. R. 2012. A Knowledge-Migration-Based Multi-Population Cultural Algorithm to Solve Job Shop Scheduling, In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference* 68-73, North America.

Reynolds, R. G., and Chung, C. J. 1996. A Testbed for Solving Optimization Problems Using Cultural Algorithms. Wayne State University, Detroit, MI.

Nakhwal, A. 2010. Agent Modeling in Decision Support System: A Case Study in a Base Hospital System. Unviersity of Windsor, Windsor, ON.

Alami, J. and El Imrani, A. A. 2005. Cultural Algorithms for Air Traffic Conflict Resolution Problem. University of Sciences, Morocco.

Guo, Y., and Liu, D. 2011. Multi-population Cooperative Particle Swarm Cultural Algorithms. In *Seventh International Conference* on *Natural Computation* 1351-1355, Shanghai.

Reynolds, R. G., and Sverdhik, W. 1994. Problem Solving Using Cultural Algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation* 645-650, IEEE.