Parallel Evolutionary Algorithms and High Dimensional Optimization Problems

Nadia Abd-Alsabour*
Cairo University, Cairo, Egypt.

* Corresponding author. Email: nadia.abdalsabour@cu.edu.eg
Manuscript submitted May 28, 2018; accepted July 20, 2018.
doi: 10.17706/jcp.13.11.1265-1271

Abstract: Optimization represents a fundamental part of many fields such as industry whose aim is the result of an optimization problem. The high dimensionality of various optimization problems raises the requirement for developing new appropriate optimization algorithms. A good instance of these algorithms is parallel evolutionary algorithms that can viably exploit the capabilities of various structures of parallel machines. This is investigated in this article through two sorts of experiments using islands algorithms. The acquired results are explained and analyzed in sections six and seven respectively.

Key words: Parallel computing, optimization problems, parallel evolutionary algorithms, high dimensionality.

1. Introduction

High dimensional optimization problems (due to the increasing volumes of data gathered by organizations or individuals) which are common in various applications need a lot of efforts to be tackled in reasonable time. Meanwhile, traditional optimization algorithms experience the computational cost such as the elapsed computational time. A good example of these algorithms is the evolutionary algorithms.

The principle idea of parallel computing is the concurrent utilization of multiple computing resources to tackle the given problem which can be separated into several parts to be solved concurrently and independently (on a multi-core computer or on a cluster of multiple computers). This is to speed up the computations. In other words, the fundamental thought behind most parallel techniques is to partition a task into segments and tackle them concurrently utilizing multiple processors. This can be applied to genetic algorithms (GAs) in an extensive variety of ways [1], [2].

The main idea of genetic algorithms for tackling a given problem is to manipulate a population of candidate solutions which are evaluated to select the best of them to reproduce and mate so as to form the next generation. Over a number of generations, good traits dominate the population resulting in an increase in the quality of the solutions. This is a simulation for Darwinian evolution; bad traits are eliminated from the population because they appear in individuals which do not survive. The good traits survive and are mixed by recombination (mating) to form better individuals [1].

Genetic algorithms are very effective and powerful search techniques that have been utilized successfully for solving practical problems in many disciplines such as engineering, business, and science. Within appropriate time, they generally can get good solutions [1]. However, when they tackle larger and harder problems, the time required for finding adequate solutions grows (their execution time can be a restricting element for utilizing them). This is because a lot of candidate solutions must be evaluated [1], [3].

Consequently, there have been various efforts to quicken GAs which leads to a diversity of options to accomplish this objective [1]. One of the most encouraging choices is to use parallel implementations.

Particularly, it is easy to implement parallel GAs and they then guarantee significant gains in performance [1].

GAs are particularly suitable for parallel computing. This is because they are regarded as embarrassingly parallel algorithms i.e., they require a large number of independent calculations with negligible synchronization and communication costs [2]. Besides, parallelism emerges normally when handling many individuals as each one is an autonomous unit. Therefore, the performance of population-based methods is extraordinarily improved when running in parallel [4].

We utilized them in this paper since they are the oldest and most established evolutionary algorithms. They are able to tackle multi-objectives, dynamic components, and non-linear constraints. Since their advancement, they have been used frequently as alternative techniques to the traditional ones and have tackled successfully a wide range of optimization problems as well [5].

This paper investigates the viability of utilizing parallel evolutionary algorithms (more particularly islands genetic algorithms) for high dimensional subset problems. This is accomplished through two types of experiments; solving the high dimensional knapsack problem utilizing the islands genetic algorithms and the conventional genetic algorithms.

The remainder of this work is organized as follows. Section 2 addresses the high dimensional optimization problems and the used one in the experiments. Section 3 highlights the utilized parallel genetic algorithms. Section 4 shows the carried out experiments and their acquired results. Section 5 analyzes and discusses the acquired results. Section 6 summarizes this paper and points out some work possible to be performed in future.

2. High Dimensional Optimization Problems

Non-trivial problems need great computational resources. Therefore, an assortment of algorithmic enhancements have been advanced such as hybrid and parallel algorithms [4]. The latter is explored in this paper.

Although it is possible for expensive hardware (such as GPU computing and computing disciplines involving natural elements like quantum systems and molecules) to handle these problems, this has a tendency to be unpractical because of many reasons engaged with getting such computation tools such as their costs. Besides, utilizing the GPU computing for example involves re-design and implementation issues in order to use islands models with them which is still investigated in many recent researches [6]-[8] . Thus, it is far more realistic and frugal to utilize computing resources that can be utilized simply in practice and in the same time provide adequate results.

Unfortunately, the performance of the vast majority of the current optimization algorithms degrades as the number of dimensions in the given optimization problem increases (greater than or equals to 100 dimensions like the majority of the real-world and industrial problems and this is the practical situation to have high dimensionality in such optimization problems) and hence using these algorithms for handling these optimization problems become intractable [6], [9]. This is because the solution space of the given optimization problem exponentially increases with the dimensionality which affects the search ability of the utilized optimization algorithm as it cannot fully investigates such a solution space within a reasonable computation time. This is known in machine learning as the curse of dimensionality problem (the amount of the required data increments exponentially with the dimensionality). High dimensionality not only influences the performance of optimization algorithms but also affects numerous predictors such as the nearest neighbors classifier despite the fact that they perform efficiently in low dimensional applications. This is because when the input attributes increase, the spread of the data increments also which results in the difficulty of handling such data) [10]-[13].

In the experiments, we utilized the high dimensional 0-1 knapsack (KP) problem as it is so crucial for many real-world problems by being the basis for such problems. Moreover, it is a good example of subset problems that represent a variety of real-world problems. As problems may be tackled in various ways such as being handled as subset problems, this raises the significance of subset problems. Because of the critical uses of these optimization problems, they have been broadly investigated recently [7], [14], [15].

The 0-1 knapsack problem (known also as the binary KP, the rucksack problem, and the backpack problem) is the problem of selecting a subset out of *n* elements into a knapsack with the main aim to maximize their gained profits and keeping their total weights within the knapsack capacity as depicted in Fig. 1. It is known as 0-1 KP since every component can be included or not in the knapsack i.e., it can be added to the knapsack no more than once.

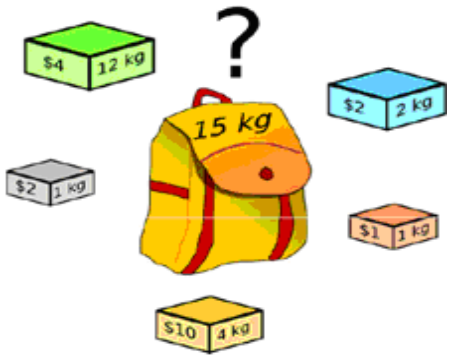

Fig. 1. The KP problem [18].

It is an NP (non-polynomial); it can't be solved in linear amount of time. Recently, various metaheuristics have been broadly used for dealing with hard optimization problems including the knapsack problem. This is on the grounds that they have demonstrated their success in solving a variety of real-world optimization problems in short time [7], [16].

Newly, this optimization problem has been widely investigated due to its enormous practical applicability in a wide range of fields like operations research, management, finance, and computer sciences. Numerous industrial and real-world problems from different fields can be dealt with as a knapsack problem. In addition, it has a variety of direct applications such as shipping organizations whose aim is to pick as much package volume into a transport plane without surpassing the weight limit [7], [16], [17].

The KP's model is:

$$\text{Maximize} \quad \sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{Subject to} \sum_{i=1}^{n} w_i x_i \leq c, \quad x_i \in \{0, 1\} \tag{2}$$

where

c = the most extreme limit of the backpack.

w_i = the weight of element i.

p_i = the profit of element i.

x_i is a binary variable equals to 1 if element i is included in the knapsack, and it equivalents to 0 if not included.

The previous equations are the fitness function to be maximized and the capacity constraint respectively [18].

3. The Islands Model

At present, several parallel approaches have been proposed such as the islands and master–slave ones. These models utilize the merit of parallel computing (the advantage of multitasking characteristics of multi-core machines) by separating an extensive size population into smaller partitions and performing concurrent random searches among them as shown in Fig. 2. These models significantly enhance the issues of premature convergence or convergence to local optimal solutions in the conventional GA techniques, where a single large population is utilized. These models execute parallel computing by allocating unutilized cores with the sub-populations that has not yet been worked. Many parallel genetic algorithms were introduced to avoid the shortcoming of the conventional genetic algorithms [19].

It has been shown that the islands models have preferred performance over the single population ones [19]. They are so sophisticated and it has been noticed that utilizing them frequently prompts speedier algorithms as well as superior performance. This is on the grounds that they can converge quickly because the chromosomes' number of the sub-populations is not as much as the one of the entire population used by the conventional GAs [2]. In addition, each island can seek in varied parts of the whole search space and in this way enhancing the exploratory posture of these algorithms. They are greatly well known [2], [3].

There are alternative names as they are sometimes referred to as distributed GAs since they are commonly executed on distributed PCs. Additionally, they are called islands GAs and multiple-demes as they are like the islands model in population genetics that treats separated demes. Furthermore, they are occasionally known as coarse-grained GAs as the communication ratio is generally large [2], [20].

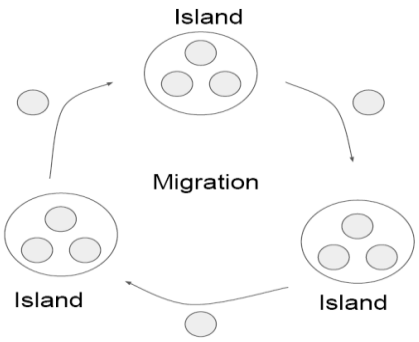

Fig. 2. Multiple-population GA.

The islands GA has multiple populations interconnected in a specific topology (like hyper-cube and ring; a unidirectional ring topology is easy to implement and analyze [19]) performing sparse migrations of information (commonly individuals) between its islands [20]. In other words, the population of chromosomes is separated into a few blocks. Each one has a specific likelihood for individuals' relocation between islands; individuals having larger fitness have larger chances of migration [19]. It is distinguished by a large computation/communication rate [5].

Whilst the master-slave technique does not influence the behavior of the algorithm, the islands model technique alters the way the GA works. For instance, in master-slave GAs, selection considers all the population but in the islands model, it investigates only a division of chromosomes. Additionally, in the master-slave any two individuals can mate (random mating), but in the islands model it is limited to a subset of individuals [2].

4. Computational Experiments

In the experiments, we present a simple case study using traditional hardware resources that shows how the computational time with a parallel GA drops dramatically to approximately the half with the

conventional GA at low cost. The experiments were completed on the 0-1 knapsack problem. So as to gain reliable results, distinctive test instances were utilized where the number of items was varied.

4.1. Method

Two kinds of experiments were carried out:

- Tackling the utilized optimization problem by the islands approach (with two islands each has 100 individuals), and
- Tackling them by traditional genetic algorithm.

In order to guarantee that the experiments are independent of any external factors, we used single machine having multiple cores instead of many machines. This helps maintains a strategic distance from contrasts in the behavior of the used islands. The same setting for the GA parameters is utilized in both of the experiments.

In the experiments, we recorded the elapsed time. Both of these experiments were executed on various instances of high dimensional benchmark 0-1 knapsack problems. These instances are from [21] with the following details appeared in Table 1.

Table 1. The Details of the Utilized Problems

Problem instance	No. of items	Knapsack Capacity
i1	100	2732
i2	250	6536
i3	500	13743.5
i4	750	20351.5

4.2. Results

Table 2 depicts the acquired results that represent the average of ten independent runs (as stochastic methods have to be repeated several times to get reasonable outcomes). The experiments were run on a PC with 2 GHz CPU and 3 GB RAM using R language [22]. The binary format was utilized because it is the most appropriate one for tackling knapsack problems [10], [11]. The obtained results are appeared in the following table.

Table 2. The Obtained Results

Problem instance	Average time (traditional GA)	Average time (Parallel GA)
i1	16.279	16.686
i2	33.634	34.961
i3	62.019	65.48
i4	91.737	95.347

The second and third columns in Table 2 are the average computational time (in seconds) of a traditional GA and a parallel GA respectively which shows how much gain in the computational time with the use of the islands models. The results in the third column represent the average of ten independent runs for getting two solutions (as we used two islands) which are around two times the computational time in the case of the traditional evolutionary algorithms.

5. Discussion

The paper explores the significance of parallel algorithms for high dimensional optimization problems at low cost which was accomplished by performing two kinds of experiments utilizing several high

dimensional knapsack instances using low cost computational recourses. Although there are many recent researches about tackling high dimensional optimization problems using non traditional computational resources (such as GPUs [1], [4]), the experiments were performed using an affordable PC to show that even in this case, we can gain much speed in getting the results.

Using parallel evolutionary algorithms help avoid the main drawback of evolutionary algorithms which is the large computational time. Moreover, this gain in the speed can be greatly increased with the use of more islands.

6. Conclusions and Future Work

The target of this work was examining the importance of parallel computing and more particular islands models for tackling high dimensional optimization problems . This is accomplished by implementing two kinds of experiments (using parallel and traditional optimization algorithms) utilizing diverse instances of high dimensional 0-1 knapsack problems . The main motivation for this work was to answer this question: can we increasing the use of islands models by investigating more their capabilities for tackling high dimensional optimization problems instead of using other unaffordable recent computational tools such as GPUs?

The availability of a variety of parallel hardware capabilities encourages us to move further towards developing software systems to benefit from these capabilities and eventually achieve the crucial aim of numerous users which is the low computational time.

As for the future work, much testing have to be performed with the utilization of various optimization problems. Besides, much optimization for the parameters of the islands models can enhance their performance. Generally speaking, islands models still need much research in order to use them further in practice.

References

[1] Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseauxet Systems Repartis, 10(2),* 141-171.

[2] Luque, G., & Alba, E. (2011). *Parallel Genetic Algorithms: Theory and Real World Applications*. Springer.

[3] Pospichal, P., Schwarz, J., & Jaros, J. (2010). Parallel genetic algorithm solving 0/1 knapsack problem running on the GPU. *Proceedings of 16th International Conference on Soft Computing MENDEL*: *Vol. 2010.* (p. 64).

[4] Luong, T., Melab, N., & Talbi, E. (2010). GPU-based island model for evolutionary algorithms. *Proceedings of GECCO'10,* Portland, Oregon, USA.

[5] Roeva, O., Fidanova, S., & Paprzycki, M. (2013). Influence of the population size on the genetic algorithm performance in case of cultivation process modeling. *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems (FEDCSIS)* (pp. 371-376).

[6] Nananukul, N. (2017). Parallel algorithm for combinatorial optimization problem. *Journal of Telecommunication, Electronic and Computer Engineering.*

[7] Fidanova, S. (2007). Ant colony optimization and multiple knapsack problem. In J. Rennard (Ed.), *Handbook of Research on Nature Inspired Computing for Economics and Management* (pp. 498-509).

[8] Abd-Alsabour, N. (2017). Nature as a source for inspiring new optimization algorithms. *Proceedings of the 9th International Conference on Signal Processing Systems* (pp. 51-56).

[9] Abd-Alsabour, N. (2018). On the role of dimensionality reduction. *Journal of Computers, 13(5),* 2018.

[10] Olariu, S., & Zomaya, A. Y. (2005). *Handbook of Bio-inspired Algorithms and Applications.* London: Chapman & Hall/CRC.

[11] Gorunescu, F. (2011). *Data Mining-Concepts, Models and Techniques.* Springer-Verlag Berlin Heidelberg.
[12] Maimon, O., & Rokach, L. (2005). *Data Mining and Knowledge Discovery Handbook*. Springer.
[13] Yang, P., Tang, K., & Yao, X. (2018). Turning high-dimensional optimization into computationally expensive optimization. *IEEE Transactions on Evolutionary Computation, 22(1),* 143-156.
[14] Abd-Alsabour, N. (2015). Binary ant colony optimization for subset problems. In S. Dehuri, A. K. Jagadev, & M. Panda (Eds.), *Multi-objective Swarm Intelligence- Theoretical Advances and Applications, Studies in Computational Intelligence* (pp. 105-121).
[15] Maniezzo, V., & Roffilli, M. (2008). Very strongly constrained problems: An ant colony optimization approach. *Cybernetics and Systems: An International Journal, 39(4),* 395-424.
[16] Syarif, A., Aristoteles, A. D., & Malinda, R. (2016). Performance evaluation of various genetic algorithm approaches for knapsack problem. *ARPN Journal of Engineering and Applied Sciences*, *11(7),* 4713-4719.
[17] Guler, A., Berberler, M., & Nuriyev, U. G. (2016). A new genetic algorithm for the 0-1 knapsack problem. *APJES IV-III,* 9-14.
[18] Yaghini, M. (2009). *Genetic Algorithms Part 2: The Knapsack Problem.*
[19] Li, C. C., Lin, H., & Liu, J. C. (2017). Parallel genetic algorithms on the graphics processing units using island model and simulated annealing. *Advances in Mechanical Engineering, 9(7).*
[20] Madera, J., Alba, E., & Ochoa, A. (2006). A Parallel island model for estimation of distribution algorithms. In J. A. Lozano, P. Larranaga, I. Inza, & E. Bengoetxea (Eds.), *Towards a New Evolutionary Computation.*
[21] KNAPSACK0-1 — Data for the 0-1 Knapsack problems. Retrieved from http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/
[22] R: A language and environment for statistical computing. Retrieved from http://www.R-project.org. *R Foundation for Statistical Computing*.

Nadia Abd-Alsabour (BSc, MSc, and PhD) is an assistant professor at Cairo University, Cairo, Egypt. Her research interests are swarm intelligence, optimization, data mining, machine learning, artificial intelligence and software engineering. Dr. Abd-Alsabour is a publicity chair, session chair, PC member, guest editor, and a reviewer in many international conferences and journals.