Local search for parallel optimization algorithms for high diminsional optimization problems

Nadia Abd-Alsabour[1,*]

[1]Cairo University, Cairo, Egypt

Abstract—Local search algorithms perform an important role when being employed with optimization algorithms tackling numerous optimization problems since they lead to getting better solutions. However, this is not practical in many applications as they do not contribute to the search process. This was not much studied previously for traditional optimization algorithms or for parallel optimization algorithms. This paper investigates this issue for parallel optimization algorithms when tackling high dimensional subset problems. The acquired results show impressive recommendations.

Keywords—***local search; optimization algorithms; paralell algorithms; high dimensional subset problems***

1 Introduction

Optimization algorithms provide a crucial role since many real-world applications are optimization problems that can be tackled using these algorithms. They involve searching for the best configuration of the problem factors (the most ideal solution) to achieve particular objectives [1].

Many fields involve dealing with high dimensional optimization problems nowadays. An example of these optimization problems is subset problems. They require choosing as opposed to ordering or sequencing [2]. This is because the order between the solution parts in a partial solution is not significant. This raises the need for advancing suitable algorithms to tackle them especially when they are high dimensional.

The reason for their importance is that a variety of real-world applications can be tackled as subset problems besides their significant applications [3]-[4]. However, these applications can't be tackled utilizing traditional optimization algorithms within reasonable time. In addition, these algorithms may not be able to provide high quality solutions. This has motivated researchers to advance appropriate optimization algorithms such as hybrid and parallel algorithms. The latter have proven their success when handling complex optimization problems such as high dimensional optimization problems since dimensionality reduction is not applicable in all of these optimization problems.

Genetic algorithms are inherently parallelizable search methods since they need a big no. of independent calculations with insignificant synchronization and communication costs [5]. Besides, parallelism emerges normally when utilizing populations' algorithms as every member in these populations is an autonomous unit. Therefore, when running in parallel, the performance of these algorithms is enhanced extraordinarily [6]. Can these algorithms be enhanced by employing local search algorithms with them?

This paper investigates this question i.e., the role that local search algorithms may provide if they are employed with parallel optimization algorithms when tackling high dimensional subset problems. This is accomplished through 2 sorts of experiments; with utilizing the optimization algorithm individually, and with including to them a variety of local search algorithms.

The rest of this paper is structured as follows. The next section addresses the local search. Section 3 demonstrates the parallel optimization algorithms. The experiments are demonstrated in Section 4. Section 5 details the acquired results. The conclusions and the future work are given in the last section.

2 local search and optimization algorithms

* Corresponding author: nadia.abdalsabour@cu.edu.eg

Recently, hybridizing more than one optimization algorithms together to handle a given problem is commonly used and often provides better results than utilizing the utilized optimization algorithm individually. A common type of hybridization is that the utilized optimization algorithm assembles an initial solution. Then, this solution is enhanced by a local search algorithm as failure in local optimum may take place with optimization algorithms i.e., these algorithms get the same solution again and again which means exploring the search space stops. This offers a point of view to include local search methods to escape from the local minima through expanding the neighborhood of the present solution. Consequently, improving the main ability of the host algorithm which is investigating increasingly the set of all feasible solutions (the search space) [1], [3] as depicted in Figure 1.

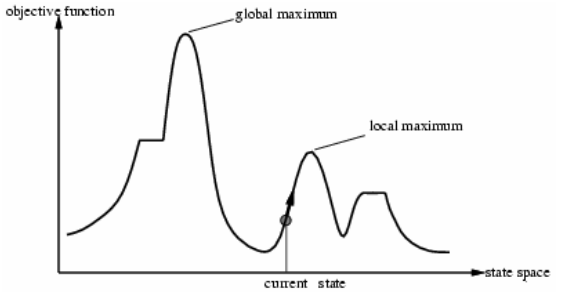

Fig. 1. Landscape of search

These algorithms start with an initial solution and try to find a superior one out of an appropriately described neighborhood of the current solution. In its simplest variant, it examines the neighborhood such that if a better solution is discovered, it will replace the current one. Such algorithms are called that as every move is performed only if the discovered solution is better than the current one. They end once they reach a local minimum. The choice of a reasonable neighborhood structure is vital for these algorithms and must be performed in a problem dependent manner [1], [7] - [9].

Local search methods have 2 types. The first one examines the neighborhood and chooses the first solution which is better than the present one. The second one completely examines the neighborhood and returns the solution of the required objective function value. Both of these end at local minima [1].

They have been employed with many optimization algorithms to keep them from getting stuck in local minima when tackling various applications. For example, it is alternatively utilized with ant colony optimization algorithms to perform incorporated actions that can't be performed by a single ant such as observing the path constructed by each agent and selects an ant or more which are then allowed to deposit more pheromone on the solution parts that they passed. Since ant colony optimization algorithms' solution construction uses a different neighborhood as opposed to local search, there is a large probability that the latter will improve a solution constructed by an ant.

However, there are circumstances where such coupling ends up recognizably useless such as very strongly constraint applications. These are the ones for which local search polynomial neighbourhoods have slight solutions or none at all and in this manner local search becomes of extremely restricted utilize and acquiring a feasible solution is exceptionally hard [4], [10]. A good example of these applications is subset applications that necessitate selecting and hence employing a local search method to the used optimization algorithm for handling them should not be recommended.

On the other hand, local search as a solving tool individually suffers from finding great beginning solutions [7]. Besides, the performance of iterative improvement techniques when applied to optimization applications is typically so inadmissible [1].

3 Parallel algorithms

There are circumstances where conventional optimization algorithms can not tackle successfully high dimensional optimization problems in reasonable time. In these situations, parallel algorithms can be utilized as they have proven their success when handling numerous optimization problems. However, tackling real-world optimization problems having complex features such as high dimensionality or being subset problems (messing many helpful features that facilitate solving them as other optimization problems) requires extra features in order for such problems to be solved within reasonable time i.e., the entire system can be having more than one algorithm to be run in parallel. This paper investigates such systems when tackling high dimensional subset problems.

The islands population-based approaches have favoured performance over the approaches since parallel searches with the exchange of information between the searches are frequently superior to independent ones. They are very sophisticated and can converge rapidly in light of the fact that the individuals' number of the sub-populations is less than the one of the whole population utilized by the traditional genetic algorithms. Additionally, every island seeks in varied areas of the entire search space which improves the exploratory feature of these techniques. Therefore, the islands approaches consolidate the speed of having numerous processors and of having parallel searches as well i.e., they are composed of parallel computational components that are executed on a parallel machine to be parallel search methods on both software and hardware levels. Islands GAs are utilized in the experiments performed in this work since they have been widely utilized for handling various real-world optimization problems [5], [11]-[14].

Generally speaking, parallel genetic algorithms have been widely applied to many fields such as numerical mathematics, graph theory, and computer science, engineering, finance and economics, etc., [15].

The thought of GAs for handling optimization applications is that they begin with a set of potential randomly created solutions. And then, the different operators are applied to this population to get a new one. To measure the quality of a solution, a fitness function is utilized. An individual having a bigger wellness is probably going to be chosen numerous times. This is in contrast to the weaker ones [16].

Starting with one population then onto the next one is accomplished through the genetic operators. The selection operator selects pairs of chromosomes that will duplicate to get newer populations. Mutation and crossover are two of the most broadly utilized operators with genetic algorithms which utilize binary representation. Mutation works on a single chromosome and alters a bit randomly. Crossover works on two parents to create 2 child offsprings [17].

Their aim for subset problems using binary representation is to get the ideal individual in which each bit represents an element out of the original subset. One or zero means the component is selected or not respectively i.e., finding the chromosome containing the most modest number of ones that achieve the best performance [16]-[17].

4 Computational experiments

Several experiments utilizing different instances of the utilized benchmark subset problem were implemented as follows.

4.1 Method

The following experiments were performed:

- Without including a local search method to the utilized algorithm, and
- With employing various local search methods to the utilized optimization algorithm. These are:
    - Nelder-Mead,
    - Conjugate-gradient, and
    - BFGS with box constraints.

The utilized algorithm accompanied with a local search algorithm follows these steps:

1. Set the needed parameters and the fitness function.
2. Randomly generate the initial population.
3. Asses each individual in the given population.
4. Create the offspring population and asses each member in it.
5. Apply the local search method on each offspring child, assess it, and supplant it if there is an improved one.
6. Execute the replacement to keep the good members in the next population.
7. This procedure stops when the stopping condition is satisfied. Otherwise, go to step 4.

The genetic algorithm that does not utilize local search methods follows these steps except the sixth one.

4.2 The benchmark utilized problem

In the experimints, we utilized the knapsack problem (KP) which is the problem of selecting a subset out of a superset of elements such that the total profit is maximized and the total weight of all the selected objects does not exceed the maximum capacity. Any element can be selected at most once and has a weight and a profit. Like other high dimensional NP optimization problems, this optimization problem can't be tackled in a linear amount of time.

Recently, it has been broadly investigated due to its enormous practical applicability in numerous domains like industry, management, and operations researches. Besides having many direct applications such as shipping organizations, numerous real-world industrial applications from different fields can be tackled as knapsack problems [3], [19]-[20].

The experiments utilized the knapsack instances appeared in Table 1.

Table 1. The details of the utilized instances,

Problem instance	No. of elements	Maximum capacity
1	100	2732
2	250	6536
3	500	13743.5
4	750	20351.5

4.3 Results

Table 2 presents the results of 2 systems with and without employing local search algorithms utilizing the knapsack instances presented in the previous Table. The results are the average of ten independent runs. The experiments were developed using R language [21]. The same setting was utilized for the utilized genetic algorithm. The binary representation was used as it is the most reasonable one for tackling subset applications.

The second, fifth, eighth, and eleventh columns of Table 2 list the average number of the selected items to be included into the knapsack. The third, sixth, ninth, and twelfth columns list the average of the total weights of the selected items in the knapsack. The fourth, seventh, tenth, and thirteenth columns show the average of the total profits of including the selected items into the knapsack. These results are gotten from employing BFGS with box constraints, conjugate-gradient, Nelder-Mead local search algorithms, and without employing any of them respectively.

We don't measure the computational time since including an additional computational component such as a local search algorithm obviously increases it.

5 Discussion

This work researches the effect of including local search on the performance of parallel optimization algorithms by implementing two sorts of investigations using several knapsack instances.

The results show that integrating different local search techniques to the used optimization algorithm can lead to better results such as the Conjugate-gradient method when handling the first instance and the Nelder-Mead method when solving the fourth instance. However, for the same instance, other local search methods can negatively affect the performance such as Nelder-Mead, and BFGS with box constraints when tackling the first three instances which emphasize that when designing a hybrid system, one should consider whether the employed local search method can improve the results or not. Subsequently, different local search techniques have to be tried. In addition to considering the general requirements for applying local search methods that are the objective function, the search space, and the neighborhood structure [1].

Moreover, one should not only consider the known local search methods but he/she can design/propose a local search method that particularly suits the given optimization problem. In addition, one can represent the given optimization problem in a new way such that a particular known local search method can be employed with the utilized optimization algorithm tackling this given optimization problem such as the work of Bondarenko [22].

6 Conclusions and future work

The aim of this paper was the investigation of the impact of adding local search to the host parallel optimization algorithm. This is achieved by 2 experiments with and without employing local search when handling high dimensional 0-1 knapsack instances.

A local search method when being integrated to an optimization algorithm when handling a subset problem may lead to getting better solution while other local search methods can worsen the performance of the host optimization algorithm besides the extra computation time.

As for the future work, much testing utilizing other subset problems should be performed. Besides, other parallel algorithms should be utilized.

References

1 C. Blum, & A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, ***35***(3), 268-308. (2003)

2 C. Solnon, & D. Bridge, An ant colony optimization meta-heuristic for subset selection problems. In: Nedjah, N., Mourelle, L.M. (eds.) Systems Engineering Using Particle Swarm Optimization, pp. 3–25. Nova Science Publishers, New York, (2006)

3 S. Fidanova, Ant colony optimization and multiple knapsack problem. In: JP. Rennard, (Ed.), Handbook of research on nature inspired computing for economics and management. pp. 498-509, Idea Group, (2007)

4 V. Maniezzo, & M. Roffilli, Very strongly constrained problems: an ant colony optimization approach. *Cybernetics and Systems: An International Journal*, ***39***(4), 395-424. (2008)

5 G. Luque, & E. Alba, Parallel genetic algorithms: Theory and real world applications (Vol. 367). Springer. (2011)

6 N. Melab, & E. G. Talbi, GPU-based island model for evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation* (pp. 1089-1096). ACM. (2010, July)

7 M. Dorigo, T. Stutzle, Ant Colony Optimization. MIT Press, Cambridge, (2004)

8 C. Blum, Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, ***2***(4), 353-373. (2005)

9 N. Abd-Alsabour, Hybrid Metaheuristics for Classification Problems, In: S. Ramakrishnan, (Ed.). Pattern Recognition - Analysis and Applications, InTech, (2016)

10 V. Maniezzo, and M. Milandri, An Ant-Based Framework for Very Strongly Constrained Problems, in M. Dorigo, et al. (Eds.): Ants 2002, LNCS, vol. **2463**, pp. 222-227, Springer, Berlin, Heidelberg (2002)

11 H. Mühlenbein, Parallel genetic algorithms in combinatorial optimization. In O. Balci, (Ed.) Computer Science and Operations Research: New Developments in Their Interfaces. pp. 441-453, Elsevier. (2014)

12 E. Cantú-Paz, A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, ***10***(2), 141-171. (1998)

13 C. C. Li, C. H. Lin, & J. C. Liu, Parallel genetic algorithms on the graphics processing units using island model and simulated annealing. *Advances in Mechanical Engineering*, **9** (7), 1687814017707413. (2017)

14 O. Roeva, S. Fidanova, & M. Paprzycki, Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on* (pp. 371-376). IEEE. (2013, September)

15 Z. Konfrst, Parallel genetic algorithms: Advances, computing trends, applications and perspectives. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International* (p. 162). IEEE. (2004, April)

16 A. P. Shukla, R. Tiwari, & R. Kala, *Real life applications of soft computing*. CRC press. (2010)

17 J. Yang, & V. Honavar, Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection* (pp. 117-136). Springer, Boston, MA. (1998)

18 KNAPSACK0-1 - Data for the 0-1 Knapsack Problems. Available at: https://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html. Last visited on 30-6-2018

19 A. Syarif, Aristoteles, A. Dwiastuti, and R. Malinda, Performance Evaluation of Various Genetic Algorithm Approaches for Knapsack Problem. *ARPN Journal of Engineering and Applied Sciences*. **11**(7), 4713-4719. (2016, April)

20 A. Güler, M. Berberler, U. Nuriyev, A New Genetic Algorithm for the 0-1 Knapsack Problem. Akademik Platform Mühendislik ve Fen Bilimleri Dergisi, **4**(3), 9-14. DOI: 10.21541/apjes.14020. (2016)

21 R: A Language and Environment for Statistical Computing [http://www.R-project.org]. R Foundation for Statistical Computing, Vienna, Austria.

22 A. Bondarenko, On Application of the Local Search and the Genetic Algorithms Techniques to Some Combinatorial Optimization Problems. *arXiv preprint arXiv:1004.5262*. (2010)

Table 2. The obtained results.

Problem instance	**Nelder-Mead**			**BFGS**			**Conjugate-gradient**			**Without local search**		
	No. of selected items	Total weights	Total profit	No. of selected items	Total weights	Total profit	No. of selected items	Total weights	Total profit	No. of selected items	Total weights	Total profit
1	61	2725.6	4052.7	61.3	2729	4069.5	61.2	2726.7	**4096.5**	61	2726.7	4058.2
2	152.3	6533.35	9512.95	152.3	6531.35	9455.55	152.05	6532.2	**9515.45**	150.2	6534	9478.25
3	296.95	13738.9	18559.3	296.3	13733.4	18545.75	293.2	13739.45	**18572.35**	294.5	13733.2	18713.3
4	424.1	20340.83	**26862.2**	430.35	20344.05	26600.8	434.15	20344.79	26783.3	424.85	20343.4	26740.85