Parallel Shared-Memory Multi-Objective Stochastic Search for Competitive Facility Location

Algirdas Lančinskas[1], Pilar Martínez Ortigosa[2], and Julius Žilinskas[1]

[1] Institute of Mathematics and Informatics, Vilnius University,
Akademijos 4, 08663 Vilnius, Lithuania
{algirdas.lancinskas,julius.zilinskas}@mii.vu.lt
http://www.mii.lt
[2] Universidad de Almería, ceiA3
Ctra. Sacramento s/n, La Cañada de San Urbano 04120, Almería, Spain
ortigosa@ual.es
http://www.ual.es

Abstract. A stochastic search algorithm for local multi-objective optimization is developed and applied to solve a multi-objective competitive facility problem for firm expansion using shared-memory parallel computing systems. The performance of the developed algorithm is experimentally investigated by solving competitive facility location problems, using up to 16 shared-memory processing units. It is shown that the developed algorithm has advantages against its precursor in the sense of the precision of optimization and that it has almost linear speed-up on 16 shared-memory processing units, when solving competitive facility location problems of different scope reasonable for practical applications.

Keywords: Facility Location, Multi-Objective Optimization, Stochastic Search, Shared Memory Parallel Computing.

1 Introduction

The location of facilities deals with the determination of the optimal location for a facility (or a set of facilities) which is important for the firms providing services to customers in a certain geographical region. It is believed that facility location as a science has originated from Pierre de Fermat, Evangelistica Torricelli, and Battista Cavallieri as these people independently proposed the basic Euclidean spatial median problem early in the seventeenth century [2]. But formally, the Alfred Weber's book [16] is assumed to be the most important starting point in the history of location science. Nowadays there are a lot of models of facility location proposed in literature, e.g. [3,5,12,14], which differ on their properties such as location space, describing possible locations for the facilities being located, attractiveness of facilities, or behavior of customers when choosing the facility to buy a service.

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part I, LNCS 8805, pp. 71–82, 2014.

1.1 Competitive Facility Location for Firm Expansion

Important characteristic of the models of facility location is the market environment. Consider a company is planning an establishment of several facilities in a region where other companies are already providing a service. The new facilities must be located with respect to the competition with the preexisting ones for the market share. Such a kind of facility location models are known as Competitive Facility Location (CFL) models.

If the company planing establishment of new facilities is already in the market, then it is encountered with the Competitive Facility Location for Firm Expansion, where the impact of the new facilities on the preexisting facilities of the same company should be taken into account.

In this research we are interested in the bi-objective (of two objectives) CFL problem for firm expansion, which has been proposed by Fernández et al. in [4].

Consider two firms A and B, providing a service to a set I of demand points. The firm A has a set F_A of n_A facilities and the firm B has a set F_B of n_B facilities, providing service for customers in a given geographical region and competing for the market.

The firm A wants to open a set F_X of n_X facilities in order to increase the market share. On one hand the new facilities can attract new customers from the rival firm B thus increasing the total market share of the company A. On the other hand the new facilities can attract customers who are already served by own facilities from F_A, thus giving raise of the effect of cannibalism. Therefore the firm A faces a multi-objective optimization problem to find the optimal locations for the new facilities with respect to maximization of the market share captured by the new facilities and simultaneous minimization of the effect of the cannibalism.

1.2 Multi-Objective Optimization

In general a problem of facility location as well as the CFL problem for firm expansion can be formulated as a mathematical optimization problem to find a *decision vector* $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_d^*)$, describing the locations of the facilities expected to be established, so that the value of the *objective function* $f(\mathbf{x}^*)$, describing the fitness of the selected locations, would be the minimal

$$f(\mathbf{x}^*) = \min_{\mathbf{x} \in \mathbb{D}} f(\mathbf{x}), \tag{1}$$

where $\mathbb{D}$ is a set of all possible decision vectors, called *search space*. The objective function usually is based on a relevant criterion, such as maximization of the market share of the company, minimization of costs of establishment or further maintaining of the prospective facilities, minimization of costs of communication between a facility and the customers, etc.

Like most of real-world optimization problems, location of the facilities usually requires simultaneous consideration of more than one criterion. For example, maximization of the market share of the new facilities and minimization

of communication costs, minimization of communication cost while maximizing the distance from a residential area, or all of them: maximization of the market share and distance to a residential area, while minimization of communication costs. Thus a decision maker faces a *multi-objective optimization problem*.

Without reducing the generality further we will consider the CFL problem for firm expansion, described in Section 1.1. The problem consists of two objectives; one is subject to maximization (the market share of the new facilities) whereas another one is subject to minimization (the effect of cannibalism). Due to conflicting objectives, comparison of two decision vectors by the value of a single objective is meaningless as a better decision vector for one objective can be worse, or even the worst for another one. However, the fitness of two different decision vectors can be compared by the *dominance relation*. In terms of multi-objective optimization two different decision vectors $\mathbf{x}$ and $\mathbf{y}$ can be related with each other in three different ways: $\mathbf{x}$ *dominates* $\mathbf{y}$ and vice versa, as well as none of them are dominated by the other.

In general it is said that the decision vector $\mathbf{x}$ dominates the decision vector $\mathbf{y}$ if $\mathbf{x}$ is strictly better than $\mathbf{y}$ by at least one objective, but not worse by any other objective. The dominance relation is denoted by $\mathbf{x} \succ \mathbf{y}$, and the decision vector $\mathbf{x}$ is called a *dominator* of $\mathbf{y}$. If none of two decision vectors can be distinguished as a dominator of the other, it is said that these decision vectors are *indifferent* in the sense of dominance relation.

A decision vector $\mathbf{x}$ which has no dominators in the whole search space $\mathbb{D}$ is called *non-dominated*, or *Pareto-optimal*, and a set of non-dominated vectors is called the *Pareto set*. The corresponding set of the values of the objective functions is called the *Pareto front*.

Determination of the exact Pareto front of a multi-objective optimization problem is usually a hard and time consuming task, which can be even intractable within an acceptable time. On the other hand solution of practical problems usually does not require to find the exact Pareto front, but rather its approximation. Therefore multi-objective optimization methods approximating the Pareto front as well as parallel computing techniques are usually used to tackle practical problems.

1.3 Related Works

A great amount of work devoted to an approximation of the Pareto front can be found in the literature with reference to facility location. A well known class of such algorithms are Evolutionary Algorithms (EAs), which require little knowledge about the problem being solved. For example, Redondo et al. [13] proposed a general multi-objective optimization heuristic algorithm, suitable to continuous multi-objective optimization problems; Zitzler et al. [17] proposed the Strength Pareto Evolutionary Algorithm (SPEA2) and Huapu and Jifeng [6] utilized it to solve a bi-level programming model to optimize the location problem of distribution centers; Deb et al. [1] proposed the Non-dominated Sorting Genetic Algorithm (NSGA-II) and Villegas et al. [15] utilized it to solve a bi-objective

facility location problem by minimizing operational cost of Colombian Coffee supply network and maximizing the demand.

Although EAs are popular due to applicability to various practical problems, their performance can be notably improved by incorporating a local search thus deriving so called memetic algorithms. For example, Medaglia et al. [11] utilized hybrid NSGA-II and mixed-integer programming approach to solve bi-objective obnoxious facility location problem related to the hospital waste management network; Lančinskas et al. [7] proposed a Multi-objective Single Agent Stochastic Search (MOSASS) and incorporated it in the NSGA-II, thus developing a memetic multi-objective optimization algorithm called NSGA/LSP.

The NSGA-II as well as most of EAs are quite fast, however it can be time consuming if the evaluation of the objective functions are expensive in the sense of computational resources. In order to reduce the computational time, parallel variants of NSGA-II have been proposed in [8], some of which have been applied to solve the competitive facility location problems using the large scale computing system in [9]. Consequently the parallel version of memetic algorithm NSGA/LSP, suitable for hybrid distributed-shared memory architecture of parallel computing system, has been proposed and applied to solve the CFL problems in [10]. The main disadvantage of the parallel memetic algorithm appears to be a sequential local optimization of a single solution, which leads to an idle time of some processing units during the local optimization procedure.

In this paper we will focus on development and investigation of the parallel multi-objective local search algorithm, by modifying previously proposed MOSASS, as well as its application to solve CFL problems using shared-memory parallel computing system.

The reminder of the paper is organized as follows: Sections 2 and 3 consist of the descriptions of the MOSS and its parallel version proposed, respectively; the experimental investigation of the developed algorithms and discussion of the obtained results are presented in Section 4, and conclusions of the research are formulated in Section 5.

2 Multi-Objective Stochastic Search

The Multi-Objective Stochastic Search (MOSS) algorithm is based on Multi-Objective Single Agent Stochastic Search (MOSASS), previously proposed in [7]. In MOSASS a new decision vector is generated in the neighborhood of a single initial decision vector, which is updated only if its dominator is found. MOSS uses a multi-agent concept, where new decision vectors are generated in a neighborhood of a certain decision vector that has been randomly selected from the set of all non-dominated decision vectors found so far.

The MOSS algorithm begins with an initial set A of non-dominated decision vectors, which can be generated at random over the search space $\mathbb{D}$, or obtained by other multi-objective optimization algorithm, e.g. NSGA-II, SPEA2, etc. The set A can also consist of a single decision vector that can be interesting to a decision maker and which neighborhood should be locally explored.

A decision vector $\mathbf{x}$ is randomly selected from the set A to represent a reference decision vector in generation of the new decision vector $\mathbf{x}' = \mathbf{x} + \xi$, where $\xi = (\xi_1, \xi_2, \ldots, \xi_d)$ is a vector of random values. Each ξ_i is generated by

$$\xi_i = \begin{cases} \mathcal{N}(b, \sigma), & \text{if} \quad r_i \leq 1/d, \\ 0, & \text{if} \quad r_i > 1/d, \end{cases} \tag{2}$$

where $\mathcal{N}(b, \sigma)$ stands for a random number, generated following the Gaussian distribution with the bias b and the standard deviation σ, r is a random number uniformly generated over $[0, 1]$, and d is the number of variables. Such a probabilistic method for generation of a neighbor decision vector leads to the change of a single coordinate in average when generating a new decision vector; see [7] for details and advantages of the method.

If the generated decision vector is non-dominated in the set A then the set is updated by including $\mathbf{x}'$ and removing all decision vectors dominated by it:

$$A \leftarrow (A \cup \{\mathbf{x}'\}) \setminus \{\mathbf{x} \in A : \mathbf{x}' \succ \mathbf{x}\}. \tag{3}$$

The iteration is then assumed to be successful and the algorithm proceeds to the next iteration. If the new decision vector is dominated by any one from the set A, then the opposite decision vector $\mathbf{x}'' = \mathbf{x} - \xi$ is considered. If $\mathbf{x}''$ is non-dominated in the set A, then the set A is updated by including $\mathbf{x}''$ and removing all dominated decision vectors similarly to (3). The iteration is then assumed to be successful and the algorithm proceeds to the next iteration.

If the opposite decision vector $\mathbf{x}''$ is dominated in A then the iteration is assumed to be failed.

In the case of success, the number of repetitive successful iterations is increased by one, and the number of repetitive failures is reset to zero. In the case of failure, the number of repetitive failures is increased by one and the number of repetitive successes is reset to zero.

The values of b and σ are dynamically adjusted with respect to the repetitive success or failures in generation of a neighbor decision vector. For the detailed description of adjustment of the parameters' values we refer to [7].

3 Parallel Multi-Objective Stochastic Search

The MOSS algorithm can be briefly separated into three main parts: (i) the initialization of the algorithm, where loading of initial data and assignment of initial values of the parameters take place, (ii) the main loop, where iterative process of approximation of the Pareto front takes place, and (iii) the finalization, where processing and output of the obtained result take place.

The first and third parts are much less time-consuming, comparing with the second one, and their consideration as sequential parts should not make a significant impact on the performance of the parallel algorithm. The most time-consuming part of the algorithm is the second one, where the main computational effort is on the evaluation of the values of the objective functions. Assuming that

the evaluations of the fitness of different decision vectors can be considered as independent tasks, they can be assigned to different processing units. In such a distribution of tasks the consistent access to the values of all algorithm parameters and to the whole set A of the non-dominated decision vectors found so far must be guaranteed for all processing units. Moreover if one of the processing units is updating a parameter, access of any other processing unit to that parameter is blocked in order to keep memory or data consistency.

Taking these considerations into account, the parallel version of MOSS, called ParMOSS, has been developed using shared-memory parallel programming libraries. ParMOSS begins with initialization of the parameters of the algorithm as well as the data and parameters of the optimization problem to be solved. This part of the algorithm is insignificant in the sense of computational effort, and, therefore, is performed by a single processing unit – the master.

Further the computational effort is distributed among all the processing units being used for the computations. Each of the units randomly selects a decision vector $\mathbf{x}$ from the set A, generates its neighbor $\mathbf{x}'$, and evaluates its values of the objective functions. Since the values of the objectives are evaluated, the dominance relation of the generated decision vector with those in the set A is checked. During this procedure the access for updating the set A is blocked for other processing units due to consistency memory requirements.

If $\mathbf{x}'$ is non-dominated regarding the set A, then the set is updated by including $\mathbf{x}'$ and removing all dominated decision vectors (see (3) and its description). The access to the set A is also denied for other processing units till the update is complete, as well as the accesses to the bias b and the counter of the repetitive success iterations are denied while their values are being updated.

If $\mathbf{x}'$ is dominated by any of decision vectors from A, then the opposite decision vector $\mathbf{x}''$ is considered, using the same strategy and policies to access the memory, as it is when considering $\mathbf{x}'$.

Principal scheme of the ParMOSS algorithm is given in Fig. 1. The scheme is based on the standard scheme for EA, but the parallel part of the algorithm with critical operations are highlighted. The dashed lines in the scheme mark the boundaries of the region where operations are performed by all processing units (the parallel region), and the hatched figures mark critical operations, which can be performed by a single processing unit at one time; i.e. if one processing unit performs operation enabling limited access to the set, any access to the set is denied for all other processing units.

4 Computational Experiments

The developed parallel algorithm ParMOSS has been applied to solve CFL problems, described in Section 1.1, using real geographical coordinates and populations of 5000 cities and towns in Lithuania, considered as demand points.

It was assumed that the firms A and B have $n_A = n_B = 3$ preexisting facilities and that the firm A wants to extend its market share by establishing a set F_X of $n_X = 3$ facilities. The simplest model of the behavior of customers when

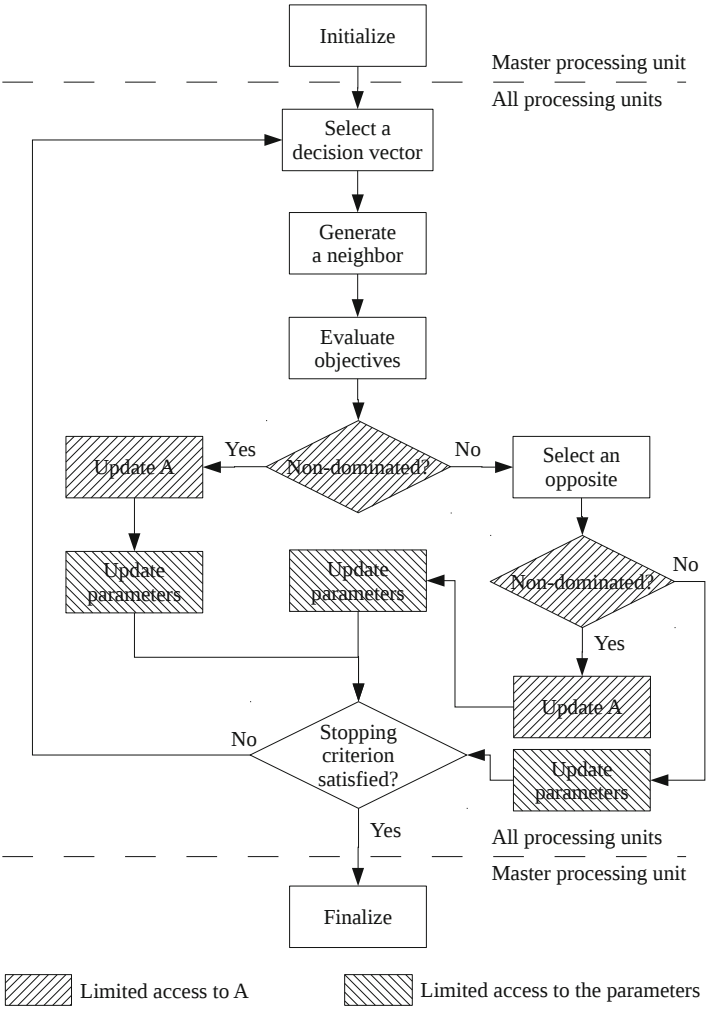

Fig. 1. Principal scheme of the ParMOSS algorithm

choosing the most attractive facility has been used, assuming that all customers from a single demand point choose the nearest facility.

Ten thousands function evaluations have been devoted for approximation of the Pareto front. Due to stochastic nature of the algorithm, each experiment has been performed 100 times, using different randomly generated initial decision vector, and the average results have been computed.

The precision of the approximation of the Pareto front has been evaluated by the Hyper-Volume (HV) metric, proposed by Zitzler and Thiele in [18]. The HV measures the area captured by the points in the obtained Pareto front approximation and the given reference point **r**. The concept of the HV metric is

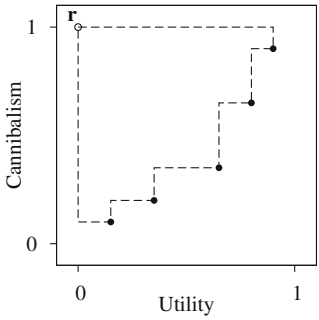

Fig. 2. Illustration of the hyper-volume metric

illustrated in Fig. 2, where the filled points stand for non-dominated decision vectors in the objectives space, hollow point – the reference point, and the dashed line marks the boundaries of the dominated area. The larger dominated area means better approximation of the Pareto front. The obtained approximation of the Pareto front has been scaled to the interval $[0, 1]^2$ with respect to the extreme values of the objectives – the maximal possible utility (the market share of the new facilities), which is equal to the total market share of both firms A and B, and the maximal possible cannibalism, which is equal to the market share of the firm A. The reference point is then chosen to be $[0, 1]$.

The performance of the parallel algorithm has been evaluated by the speed-up

$$S_p = \frac{T_0}{T_p} \tag{4}$$

of the algorithm, where T_0 stands for the time needed to solve the problem using the sequential algorithm and T_p stands for the time needed to solve the problem using the parallel algorithm on p processing units. Here $p = 1, 2, \ldots$ and T_1 might differ from T_0 as the behavior of parallel algorithm on a single processing unit might differ from the behavior of the sequential algorithm.

4.1 Impact on the Precision

Since MOSS has been derived from MOSASS by changing the strategy for selection of the decision vector from the set of non-dominated ones, the impact of the modification on the quality of the approximation must be investigated.

The quality of the approximation has been investigated by solving the CFL problem with 5000 demand points. The problems have been solved by sequential versions of MOSASS and MOSS.

The obtained results are presented in Fig. 3, where the vertical axis stands for the number of function evaluations, the horizontal one – for the average values of HV, and different curves stand for the different algorithms.

One can see from the figure, the proposed MOSS algorithm notably outperforms its precursor MOSASS independent on the number of function evaluations devoted for the approximation.

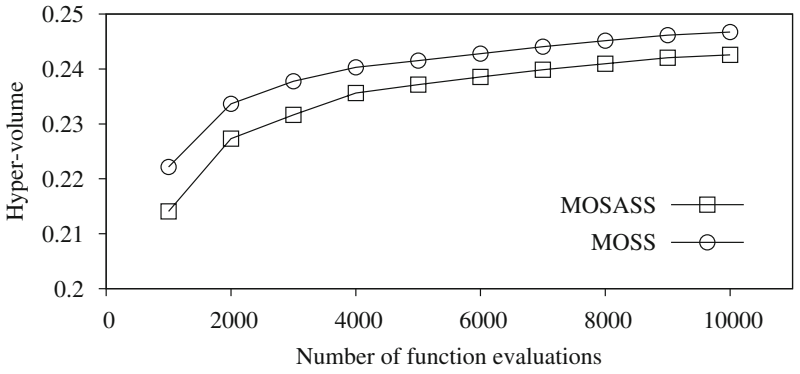

Fig. 3. Values of hyper-volume, obtained using different algorithms and different numbers of function evaluations

The parallel version of MOSS does not have exactly the same behavior as the sequential one, therefore it must be verified if the quality of the approximation has not been reduced when parallelizing.

The verification has been done by solving the CFL problem of different scope: 1000 and 5000 demand points. The problems have been solved by the parallel algorithm ParMOSS using different numbers of processing units: 1, 2, 4, 8, and 16. The obtained results have been additionally compared with the results obtained by sequential version of the algorithms.

The results are illustrated in Fig. 4, where two first couples of columns show average HV, obtained by sequential versions of MOSASS and MOSS, respectively, whereas the following ones – show average HV, obtained by ParMOSS using different numbers of processing units; different columns in a couple represent different number of demand points, and interval segment on the top of the column illustrates the confidence interval of the mean with the confidence level 0.05.

One can see from the figure that the variation of values of HV, obtained using ParMOSS on different numbers of processing units is very slight and can be considered as insignificant. This leads to the conclusion that the parallelization of MOSS does not negatively effect the precision of the approximation.

4.2 Speed-Up of the Algorithm

The efficiency of the parallel algorithm ParMOSS has been evaluated by solving the CFL problem using four different numbers of demand points: 5000, 1000, 500, and 100. The time required for computations using 5000 demand points is around 100 seconds. The problem of the smaller scope – 1000 demand points

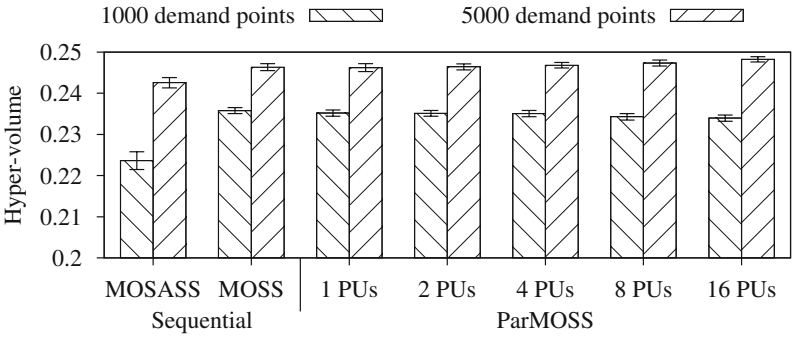

Fig. 4. Values of hyper-volume, obtained using different algorithms and different number of processing units

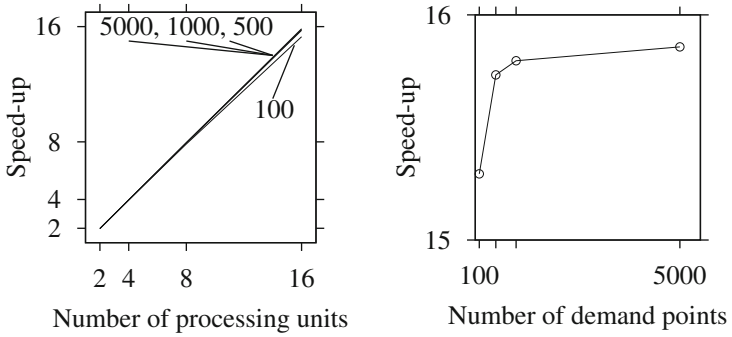

Fig. 5. The speed-up of ParMOSS, obtained using different numbers of demand points, versus the number of processing units (on the left) and the speed-up, obtained using 16 processing units, versus the number of demand points (on the right)

requires around 20 seconds, 500 demand points – 10 seconds, and 100 demand points – 2 seconds. The main effort of the computational resources (more than 99%) has been devoted to the function evaluation in all the cases, except the smallest one – 100 demand points, when function evaluations require a little bit less than 99% of all computational resources.

The time needed to solve a problem using MOSS was 0.3–0.4% larger comparing with its precursor MOSASS, as well as the ParMOSS on one processing unit requires 0.3–0.4% more time than sequential MOSS.

The speed-up of ParMOSS, obtained using 2, 4, 8, and 16 processing units is given in the left image of Fig. 5, where the horizontal axis corresponds to the number of processing units, and the vertical one – to the speed-up. One can see from the figure, the speed-up of the algorithm is almost linear for all experiments except for the case of using 16 processing units to solve the smallest problem with 100 demand points.

The dependence of the speed-up of the algorithm, obtained using 16 processing units, on the number of demand points is illustrated in the right image of Fig. 5, where the horizontal axis corresponds to the number of demand points, and the vertical axis – to the speed-up from the range from 15 to 16.

The obtained results show that ParMOSS has almost linear speed-up, since the further reduction of the demand points is not reasonable in practical CFL problems, and vice versa – further increment of the scope of the problem cannot reduce the speed-up, but rather increase, as the time required for function evaluations will be increased, thus increasing the fully parallel part of the algorithm.

5 Conclusions

The parallel algorithm for multi-objective optimization, based on stochastic search, has been developed and experimentally investigated by solving the competitive facility location problem for firm expansion using up to 16 shared-memory processing units.

The results of the investigation showed that the modifications, made to the sequential algorithm, improve the precision of the optimization, measured by the hyper-volume metrics, as well as make the algorithm more suitable for parallel computing.

The performance results of the parallel algorithm showed that the algorithm has almost linear speed-up using up to 16 processing units, solving competitive facility location problem for firm expansion of different scope, reasonable in practical applications.

Acknowledgments. This work has been supported by the project "Theoretical and engineering aspects of e-service technology development and application in high-performance computing platforms" (No. VP1-3.1-ŠMM-08-K-01-010) funded by the European Social Fund.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)
2. Drezner, Z., Klamroth, K., Schobel, A., Wesolowsky, G.O.: The Weber problem. In: Drezner, Z., Hamacher, H. (eds.) Facility Location: Applications and Theory, pp. 1–36. Springer, Berlin (2002)
3. Farahani, R.Z., Rezapour, S., Drezner, T., Fallah, S.: Competitive supply chain network design: An overview of classifications, models, solution techniques and applications. Omega 45(0), 92–118 (2014)
4. Fernández, J., Pelegrín, B., Plastria, F., Tóth, B.: Planar location and design of a new facility with inner and outer competition: An interval lexicographical-like solution procedure. Networks and Spatial Economics 7(1), 19–44 (2007)
5. Friesz, T., Miller, T., Tobin, R.: Competitive networks facility location models: a survey. Papers in Regional Science 65, 47–57 (1998)

6. Huapu, L., Jifeng, W.: Study on the location of distribution centers: A bi-level multi-objective approach. In: Logistics, pp. 3038–3043. American Society of Civil Engineers (2009)
7. Lančinskas, A., Ortigosa, P.M., Žilinskas, J.: Multi-objective single agent stochastic search in non-dominated sorting genetic algorithm. Nonlinear Analysis: Modelling and Control 18(3), 293–313 (2013)
8. Lančinskas, A., Žilinskas, J.: Approaches to parallelize pareto ranking in NSGA-II algorithm. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2011, Part II. LNCS, vol. 7204, pp. 371–380. Springer, Heidelberg (2012)
9. Lančinskas, A., Žilinskas, J.: Solution of multi-objective competitive facility location problems using parallel NSGA-II on large scale computing systems. In: Manninen, P., Öster, P. (eds.) PARA. LNCS, vol. 7782, pp. 422–433. Springer, Heidelberg (2013)
10. Lančinskas, A., Žilinskas, J.: Parallel multi-objective memetic algorithm for competitive facility location. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2013, Part II. LNCS, vol. 8385, pp. 354–363. Springer, Heidelberg (2014)
11. Medaglia, A.L., Villegas, J.G., Rodrguez-Coca, D.M.: Hybrid biobjective evolutionary algorithms for the design of a hospital waste management network. Journal of Heuristics 15(2), 153–176 (2009)
12. Plastria, F.: Static competitive facility location: An overview of optimisation approaches. European Journal of Operational Research 129(3), 461–470 (2001)
13. Redondo, J.L., Fernández, J., Álvarez, J.D., Arrondoa, A.G., Ortigosa, P.M.: Approximating the Pareto-front of continuous bi-objective problems: Application to a competitive facility location problem. In: Casillas, J., Martínez-López, F.J., Corchado, J.M. (eds.) Management of Intelligent Systems. AISC, vol. 171, pp. 207–216. Springer, Heidelberg (2012)
14. ReVelle, C., Eiselt, H., Daskin, M.: A bibliography for some fundamental problem categories in discrete location science. European Journal of Operational Research 184(3), 817–848 (2008)
15. Villegas, J., Palacios, F., Medaglia, A.: Solution methods for the bi-objective (cost-coverage) unconstrained facility location problem with an illustrative example. Annals of Operations Research 147, 109–141 (2006)
16. Weber, A.: Theory of the Location of Industries. University of Chicago Press (1929)
17. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., Tsahalis, D.T., Périaux, J., Papailiou, K.D., Fogarty, T. (eds.) Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, pp. 95–100 (2001)
18. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms – a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)