Proceedings of the Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015)
Krakow, Poland

Jesus Carretero, Javier Garcia Blas
Roman Wyrzykowski, Emmanuel Jeannot.
(Editors)

September 10-11, 2015

Solution of Bi-objective Competitive Facility Location Problem Using Parallel Stochastic Search Algorithm

Algirdas Lančinskas†, Pilar Martínez Ortigosa*, Julius Žilinskas†

†Vilnius University, Lithuania
algirdas.lancinskas@mii.vu.lt
julius.zilinskas@mii.vu.lt

*University of Almeria, Spain
ortigosa@ual.es

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

I. Introduction

The Facility Location (FL) deals the optimal placement of the facilities providing goods or services in a certain geographical area with respect to maximize the utility and/or minimize an undesirable effect of the facility being located. There is a variety of FL models proposed which varies on their ingredients such as location space, attractiveness of the facilities, or customers' behavior when choosing the most attractive facility [1, 2, 4, 5]. A lot of attention is paid for the Competitive Facility Location Problems (CFLPs) in which determination of the optimal location for the new facilities involves consideration of their possible competition for the market share with the preexisting facilities.

Real-world CFLPs usually require to simultaneously consider two or several objectives when locating the new facilities; e.g. maximize the market share of the new facilities while minimizing costs for their establishment or maintenance of the facility; minimize distance between facilities and customers in accordance with requirements for the minimal distance to urban areas (actual for semi-obnoxious facilities).

Our research is focused on the Competitive Facility Location Problem for Firm Expansion (CFLP/FE) where a firm already in the market is planning to establish a set of new facilities in order to increase its market share.

II. CFLP for Firm Expansion

Consider an expanding firm F_A having n_A preexisting facilities and its competitor – the firm F_B having n_B preexisting facilities – both servicing a discrete set I of *demand points* in a certain geographical area. The firm F_A is expected to locate a set X of n_X new facilities with respect to maximize the market share of the new facilities taking into account the competition with the facilities owned by F_B. Despite the attraction of new customers from the competitor F_B, the newly established facilities can also attract customers from the facilities already owned by the expanding firm F_A thus giving rise of the *effect of cannibalism*. Therefore the firm F_A faces a bi-objective optimization problem with the following objectives: (f_1) to maximize the market share of the facilities being located and (f_2) to minimize the loss of market share of the preexisting facilities of F_A (the effect of cannibalism).

Due to conflicting objectives usually it is impossible to find a single solution which would be the best by both objectives, but rather a set of compromising (non-dominated) solutions, called Pareto set; the corresponding set of the objective functions' values is called Pareto front. Determination of the exact Pareto front usually is a hard and time consuming task. On the other hand solution of practical CFLPs usually does not require to find the exact Pareto front, but rather its approximation by a set of non-dominated solutions.

III. Parallel Multi-Objective Stochastic Search

Multi-Objective Stochastic Search (MOSS) is a random search algorithm suitable for approximation of the Pareto front of a multi-objective optimization problem. MOSS is derived from its precursor Multi-Objective Single Agent Stochastic Search (MOSASS) algorithm proposed in [3].

The algorithm begins with an initial archive $\mathbb{A}$ of solutions which are non-dominated among themselves. The new solution $\mathbf{x}'$ is generated by applying slight modifications to the solution $\mathbf{x}$ randomly sampled from $\mathbb{A}$, where the strength of the modification depends on the repetitive successful and failed iterations; see [3] for details of the generation of the new solution. If the newly generated solution $\mathbf{x}'$ is not dominated by any one in the archive $\mathbb{A}$, then $\mathbb{A}$ is updated by including $\mathbf{x}'$ and removing all solution which are dominated by $\mathbf{x}'$, and the algorithm goes to the next iteration. Otherwise, if $\mathbf{x}'$ is dominated in $\mathbb{A}$, then a symmetric (in relation with $\mathbf{x}$) solution $\mathbf{x}''$ is evaluated in the same way as $\mathbf{x}'$. If the archive is updated either by $\mathbf{x}'$ or $\mathbf{x}''$, then iteration is assumed to be successful; otherwise, the iteration is assumed to be failed.

The main computational effort of the algorithm usually is devoted to the evaluation of objective function values. The evaluations of the objective values of different solutions can be considered as independent tasks thus giving availability to distribute the computational work among different processors. In such a distribution of tasks the information about all non-dominated solutions found so far (the archive $\mathbb{A}$) as well as values of other parameters of the algorithm must be accessed by all processors. Moreover if one of the processors is updating a parameter or the archive, access to it is blocked for any other processor in order to keep memory and data consistency.

Two parallel algorithms ParMOSS/OMP and ParMOSS/MPI suitable for shared- and distributed-memory parallel computing systems, respectively, has been developed under considerations above.

The ParMOSS/OMP algorithm begins with the 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 a single processor – the master. Further each of the slaves randomly selects an individuals from the archive $\mathbb{A}$ and evaluates its objective values as well as the dominance relation in $\mathbb{A}$ (as it is described above). If any of processors is accessing the archive or any other parameter of the algorithm, the access to that parameter or the archive is blocked for all other processors.

In distributed-memory computing systems information about solutions in $\mathbb{A}$ and values of algorithm parameters must be transfered by passing messages using Message Passing Interface (MPI). In order to guarantee consistent communication between processors, one of them is devoted for the management of the communication and overall process of the algorithm. Thus the parallel version of MOSS algorithm ParMOSS/MPI for distributed-memory parallel computing systems is developed following the master-slave strategy.

The master processor selects a random solution $\mathbf{x}_i$ from $\mathbb{A}$, generates a pair of new solutions $(\mathbf{x}'_i, \mathbf{x}''_i)$ (as it is described above), and sends it to the i-th processor (the slave) with the request to evaluate the first solution $\mathbf{x}'_i$. Here i varies from 1 to the number processors p thus ensuring that the pair will be generated for each processor. After all slaves are equipped by a pair of solutions, the master proceeds to the main loop and waits for the response from any of the slaves with an evaluated solution. Although all slaves are requested to evaluate $\mathbf{x}'_i$, some of them can also be requested to evaluate $\mathbf{x}''_i$ in the later stage of the algorithm. In general the master processor proceeds depending on whether evaluation of $\mathbf{x}'_i$ or $\mathbf{x}''_i$ is received and the fitness of the received solution with respect to solutions in the archive $\mathbb{A}$.

IV. Numerical Experiments

The developed parallel algorithms ParMOSS/OMP and ParMOSS/MPI have been experimentally investigated by solving different instances of CFLP/FE: 5000, 1000, 500, and 100 demand points for ParMOSS/OMP; 5000 and 1000 demand points – for ParMOSS/MPI. The Pareto front of a single instance has been approximated by 25000 function evaluations. The average duration of a single approximation by sequential MOSS was around 728 seconds using 5000 de-

mand points, around 145 seconds – using 1000 demand points, around 73 – using 500 demand points, and around 14 seconds – using 100 demand points.

The obtained results showed that the shared-memory algorithm ParMOSS/OMP has almost linear speed-up on up to 16 shared memory processors for all instances of the problem: 5000, 1000, 500, and 100 demand points; further reduction of the number of demand points is not reasonable in practical CFLPs.

Similar experiment has been performed for the distributed-memory algorithm ParMOSS/MPI. The Pareto front of CFLP/FE with 1000 demand points has been approximated using 2, 4, 8, and 16 processors. Results of the experimental investigation showed that speed-up of ParMOSS/MPI increases linearly with the increment of the number of processors. The speed-up of ParMOSS/MPI is lower than speed-up of ParMOSS/OMP exactly by one independent on the number of processors due to an idle time of the master processor which has no computational work. These results show that the shared-memory algorithm has notable advantage against the distributed-memory one. On the other hand the shared-memory computing systems have hardware limitations in the sense of number of shared-memory processors, whereas the distributed-memory algorithm can be executed on a significantly larger number of processors.

The performance of ParMOSS/MPI has been also investigated using from 32 to 192 processors. Results of the investigation showed that the approximation of the Pareto front of the problem with 5000 demand points has been performed with almost linear speed-up of the algorithm – the speed-up on 192 processors was around 186. The approximation of the Pareto front of the problem with 1000 demand points has been performed with notably lower speed-up, comparing with previous instance – the speed-up on 192 processors is around 155 which corresponds to 80% of effectiveness of the processors. On the other hand the speed-up of the algorithm on 96 processors is around 89 which corresponds to 93% of effectiveness of the processors when performing the computations; further increment of the number of processors is not reasonable for approximation of the Pareto front of a real-world CFLP as the approximation on 128 processors has been performed within 2 seconds.

Acknowledgment

The work has been partially supported by EU under the COST Action IC1305 “Network for Sustainable Ultrascale Computing (NESUS)”.

References

[1] R. Z. Farahani, S. Rezapour, T. Drezner, and S. Fallah. Competitive supply chain network design: An overview of classifications, models, solution techniques and applications. *Omega*, 45(0):92–118, 2014.

[2] T.L. Friesz, T. Miller, and R.L. Tobin. Competitive networks facility location models: a survey. *Papers in Regional Science*, 65:47–57, 1998.

[3] A. Lančinskas, P. M. Ortigosa, and J. Žilinskas. Multi-objective single agent stochastic search in non-dominated sorting genetic algorithm. *Nonlinear Analysis: Modelling and Control*, 18(3):293–313, 2013.

[4] Frank P. Static competitive facility location: An overview of optimisation approaches. *European Journal of Operational Research*, 129(3):461–470, 2001.

[5] C. S. ReVelle, H.A. Eiselt, and M .S. Daskin. A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, 184(3):817–848, 2008.