Computers and Chemical Engineering 66 (2014) 276–289

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Asynchronous optimisation with the use of a cascade search algorithm

F. Cecelja[a,*], A. Kokossis[b], D. Du[a], S. Yang[a]

[a] Centre for Process & Information Systems Engineering, University of Surrey, Guildford GU2 7XH, United Kingdom
[b] School of Chemical Engineering, National Technical University of Athens, Athens GR-15780, Greece

ARTICLE INFO

Article history:
Received 3 October 2013
Accepted 14 February 2014
Available online 25 February 2014

Keywords:
Markov processes
Asynchronous optimisation
Parallel and distributed computing

ABSTRACT

This paper introduces the development of an asynchronous approach coupled with a cascade optimisation algorithm. The approach incorporates concepts of asynchronous Markov processes and introduces a search process that is benefiting from distributed computing infrastructures. The algorithm uses concepts of partitions and pools to store intermediate solutions and corresponding objectives. Population inflections are performed periodically to ensure that Markov processes, still independent and asynchronous, make arbitrary use of intermediate solutions. Tested against complex optimisation problems and in comparison with commonly used Tabu Search, the asynchronous cascade algorithm demonstrates a significant potential in distributed operations with favourable comparisons drawn against synchronous and quasi-asynchronous versions of conventional algorithms.

1. Introduction

Along with deterministic optimisation, stochastic search based optimisation often referred to as stochastic optimisation (Fouskasis & Draper, 2002), has found many applications in solving complex engineering problems because it overcomes problems associated with nonlinearities and discrete variables (Cavin, Fisher, Glover, & Hungerbuhler, 2004; Jayaraman, Kulkarni, Karale, & Shelokar, 2000). It is based on statistically random probabilistic driven search which guarantees full exploration of search space and exclusion of local optima (Kokossis, Linke, & Yang, 2011). The advantages and disadvantages of stochastic optimisation have been reviewed by numerous authors (Fouskasis & Draper, 2002; Gosavi, 2003; Jayaraman et al., 2000). Yet, in order to ensure convergence and to address search inefficiencies associates with randomness, various strategies have been developed to get use of past history of solutions and to find and favourise the most promising search direction. These have been formalised in a number of optimisation algorithms: perhaps the best known are Simulated Annealing (SA) (Kirkpatrick, Gellat, & Vecchi, 1983), Tabu Search (TS) (Glover, 1989), Genetic Algorithm (GA) (Goldberg, Deb, & Clark, 1992) and Ant Colony Optimisation (ACO) (Dorigo, DiCaro, & Gambardella, 1999). Still, stochastic optimisation generally suffers from slow convergence.

Many contributions were made to overcome slow convergence of stochastic optimisation, which include (i) introducing the level of intelligence in the form of exploiting knowledge about the application (Labrador-Darder, Cecelja, Kokossis, & Linke, 2009; Nandi et al., 2004) or capturing and managing knowledge from the past history of solutions to guide the search (Kokossis, Cecelja, Labrador-Darder, & Linke, 2008), (ii) parallelising or distributing the search process (Cantu-Paz & Goldberg, 2000; Leite & Topping, 1999; Talbi, Hafidi, & Geib, 1998), and (iii) improving existing or developing new algorithms to better accommodate parallel processing by enabling full asynchronous search and knowledge acquisition (Du, Cecelja, & Kokossis, 2011; Kokossis et al., 2008, 2011; Labrador-Darder et al., 2009).

Parallelising the search efforts is perhaps the most commonly reported approach of improving convergence of stochastic algorithms. It exploits advances in computing infrastructure where parallel computing resources are readily available in the form of grid networks or cloud computing across the Internet. Parallel versions of ACO (Reimann, Doerner, & Hartl, 2004; Yang & Zhuang, 2010), GA (Cantu-Paz & Goldberg, 2000; Lim, Ong, Jin, Sendhoff, & Lee, 2007), TS (Borfeldt, Gehring, & Mack, 2003; Cordeau & Maischberger, 2012; Crainic, Toulouse, & Gendreau, 1996; Talbi et al., 1998) and SA (Leite & Topping, 1999; Peierls & Deng, 1998) have been reported. The most common approach is sequential and/or synchronous mode where multiple threads of algorithms are executed sequentially or in parallel exploring different regions with varying degree of collaboration between them (Fouskasis & Draper, 2002; Peierls & Deng, 1998). In order to minimise redundant search, that is to minimise repeated or overlapped moves,

* Corresponding author. Tel.: +44 01483 686 585; fax: +44 01483 812 556.
E-mail addresses: f.cecelja@surrey.ac.uk (F. Cecelja), akokossis@mail.ntua.gr (A. Kokossis).

http://dx.doi.org/10.1016/j.compchemeng.2014.02.009
0098-1354/

Notation

$A_{n_p,t}$	domain of $D^h_{n_p,t}$
$D^h_{n_p,t}$	cascade of n_p pools at time instance t
f	objective function of an optimisation problem
$f^{\max}_{j,t}$	maximum f in $P_{j,t}$ at time instance t
$f^{\min}_{j,t}$	minimum f in $P_{j,t}$ at time instance t
F	superset of numerical functions including the objective function f
$F^{\max}_t$	maximum objective value of the population available at time instance t
$F^{\min}_t$	minimum objective value of the population available at time instance t
g	a numerical function in F, normally refers to inequality constraints of the optimisation problem
g^{L,n_M}_t	Markov process at time instance t
G	set of disjoint partitions in S
G_1	the highest partition associated with the temperature T_1
$G_{j,t}$	partition j at time instance t
G_{n_p}	the lowest partition associated with temperature T_{n_p}
h	a numerical function in F, normally refers to equality constraints of the optimisation problem
$h_{i,j}$	numerical function over $G_{i,j}$
h^T_j	thread, a software programme to be executed independently
h^{sm}	measure of object (C, P_j)
i	index of points (solutions) in partition (pool) j
j	index of partitions (pools) from top to bottom, $j = 1, 2, \ldots, n_p$
L	length of Markov process
M_j	number of available points in $G_{j,t}$
n_I	number of consecutive iterations
n_M	size of Markov space (number of Markov processes running in parallel)
n_T	number of threads to be executed independently
n_{ST}, n_Q	purpose defined integer numbers
P_1	the highest pool associated with the lowest quality solutions
$P_{j,t}$	pool j at time instance t
P_{n_p}	the lowest pool associated with the highest quality solutions
$Q_{j,t}$	population of partition (pool) $G_{j,t}$ $(P_{j,t})$ at time instance t
Q^{n_M}	Markov space of n_M Markov processes
R	random number distributed uniformly in $[0,1] \in \Re$
S	feasible region of an optimisation problem
$s_{i,j,t}$	point i in partition j at time instance t
T_1	temperature associated with the highest pool P_1
T_j	parameter associated with pool P_j called temperature
T_{n_p}	temperature associated with the lowest pool P_{n_p}
W^{n_T}	space of n_T threads
θ	real number defining the cooling schedule that is distribution of temperatures between T_1 and T_{n_p}
ε	a small number in $[0,1] \in \Re$

synchronisation introduced after a full set of parallel threads have been completed, at which point solution analysis takes place and new search starts afresh. In consequence, the speed of execution is dictated by the slowest thread executed on the slowest computer. The computational overhead associated with communication, computer management activities and idling of faster threads becomes dominant and benefits from parallel execution limited.

The development of a fully distributed stochastic optimisation algorithms capable of parallel processing without the need for synchronisation still remains a challenge. Cascading algorithm (CA) has been proposed as an approach to break down optimisation threads and Markov chain and hence inherently allowing for parallel execution (Kokossis et al., 2011). This algorithm stores intermediate solutions into pools (and partitions). Similarly to SA, the number of pools is equivalent to temperatures. Pools are ordered according to the quality of solutions (objective values) forming a cascade with the properties, the norm of population and deviation, which can be controlled. The CA search operates iteratively by choosing randomly a candidate point which is accepted/rejected by the probability controlled by the temperature. The cascade requirements reflect the quality of solutions which are verified upon arrival of new points, the process known as inflection. This property of CA enables independent generation of solutions and serves as the base for full asynchronous operation.

Building upon the CA approach, this paper proposes a distributed asynchronous operation of CA which is achieved by implementing asynchronous Markov processes to independently generate new points, and dynamic inflection to minimise redundant moves. More precisely, the process of dynamic inflection of population assures that new Markov processes, although executed independently, benefit from the most recent population of solutions without the need for synchronisation with other processes running in parallel. The whole process is formalised in the form of an Asynchronous Cascade Algorithm. The proposed algorithm is implemented on the Globus upperware with Grid Superscalar used to dispatch computational tasks for parallel computing. To differentiate and report benefits from distributed computing and asynchronous operations, a set of computationally expensive problems has been selected. The comparisons are drawn against optimisation algorithms which include conventional and parallel implementations of SA and TS.

The paper addresses a general formulation for the optimisation problem in the form of:

$$\begin{aligned} \min \quad & f(x,y) \\ \text{s.t.} \quad & h(x,y) = 0 \\ & g(x,y) \le 0 \\ & x \in X \equiv \Re^n, y \in Y \equiv \{0,1\}^n \end{aligned} \tag{1}$$

where (x, y) accounts for the domain and (h, g) for the problem constraints.

2. Concepts and definitions

2.1. Basic concepts of cascading

The principle of cascading allows for grouping solutions by their quality (Kokossis et al., 2011). The feasible region S of the optimisation problem (1) is

$$S = \{(x,y) | h(x,y) = 0 \wedge g(x,y) \le 0, x \in X, y \in Y\} \tag{2}$$

with the range C of f

$$C = \{f(x,y) | \forall (x,y) \in S\} \tag{3}$$

Let G be a set of disjoined partitions in S

$$G \equiv U_j G_j \subseteq S \tag{4}$$

where partition G_j is ordered set of finite elements in S

$$G_j = \{s_{1,j}, s_{2,j}, \ldots, s_{M_j,j}\}, \quad s_{i,j} \in S, \quad i = 1, 2, \ldots, M_j, \quad j = 1.2, \ldots, n_p \tag{5}$$