Parallel and Distributed Metaheuristics

Czesław Smutnicki(✉) and Wojciech Bożejko

Wroclaw University of Technology, Wroclaw, Poland
czeslaw.smutnicki@pwr.edu.pl

Abstract. The paper deals with problems and ultramodern approaches recommended to solve various combinatorial optimization (CO) tasks, by using different types of computing environments, including various clouds (CC). A lot of new ideas have been proposed or at least outlined. Non-standard evaluation of the goal function value is also considered.

1 Introduction

CO problems derived from practice, by reasons of characteristic features of the domain, require relatively high calculation power in order to find a solution satisfactory for the user, [11]. This power is usually required in a relatively short slot of time, to hammer out the properly good solution, used next in practice in the long application period. The amount of calculations has tendency contrary to the quality of provided results.

Researchers' and practitioners' view on optimization tasks generated by CO problems to realize an on-line decision, resource usage balancing, production and/or transport planning, scheduling, timetabling, etc. significantly has changed in recent years. Huge effort has been done by scientist in order to reinforce power of solution methods and to fulfil expectations of practitioners. Metaheuristics, perceived as universal "medicine" for basic troubles of CO algorithms, have reached the limit in very recent years and are replaced by a new ultramodern approaches being as yet in the development phase. CO problems with unimodal, convex, differentiable scalar goal functions disappeared from research labs, because a lot of satisfactory efficient methods were already designed. There are still remained very hard cases: multimodal, multi-criteria, non-differentiable, NP-hard, discrete, with huge dimensionality, with exponential increase of the number of local extremes, without a priori information about data, etc. These practical tasks, generated by computer systems and networks, industry and market, evoke serious troubles in the process of seeking global optimum. Any success in algorithms development strike practitioners fancy, so permanent research in this area are still welcome.

2 Optimization Dilemmas

Practitioners usually want to evaluate solutions from various points of views, thus using a number of different criteria. Then, we refer hereinafter to the following

R. Moreno-Díaz et al. (Eds.): EUROCAST 2015, LNCS 9520, pp. 72–79, 2015.
DOI: 10.1007/978-3-319-27340-2_10

vector optimization task: find $x^* \in \mathcal{X}$, such that

$$K(x^*) = \min_{x \in \mathcal{X}} K(x), \tag{1}$$

where

$$K(x) = [K_1(x), \ldots, K_s(x)]^T \tag{2}$$

and x, x^*, $\mathcal{X}$ and $K(x)$ are solution, optimal solution, set of feasible solutions and vector of goal function, respectively. The forms of x, $\mathcal{X}$ and $K_i(x)$, $i = 1, \ldots, s$ depend on the type of optimization task. In practical CO conditions, we assume that $\mathcal{X}$ is discrete, $K_i(x)$ are nonlinear and non-differentiable, x is a combinatorial object or composition of objects, whereas the optimization problem is proved to be strongly NP-hard.

Problems (1)–(2) covers almost all discrete optimization cases. Assuming $s = 1$ one can obtain the classical well examined but still hard single criterion optimization. For $s > 1$, the 'min' operator in (1) does not specify any particular technology of minimization over the set of vectors, therefore we still need a method of vector comparison, which implies from user preferences expressed directly or indirectly. Up to now a lot of user expectations were proposed, examined and implemented, see surveys [6,9,10] or summary in [12]. The primary goal of multi-objective optimization is to model preferences of the decision maker, expresses as the importance of each particular criteria, see also [12].

As the result of long-time research, there have been recognized main disadvantages of the solution methods, namely features commonly observed in conventional sequencing computer environments and detected also in parallel and distributed calculation environments. Among mentioned disadvantages are: (a) inability of finding any feasible solution in a reasonable time, (b) NP-completeness of checking whether $\mathcal{X}$ is empty, (c) exponentially increasing calculation cost while searching optimum solution x^* in the set $\mathcal{X}$, (d) poor approximation and/or slow convergence to the Pareto frontiers, (e) high calculation cost for management of non-dominated solutions, (f) slow convergence to optimal or suboptimal solution not too worse from $K(x^*)$, (g) premature convergence to approximate solutions of poor quality, (h) search stagnation, (i) imprecise data of the instance. Up to now, there have been identified a few reasons, discussed below, considered as responsible for appearance of these faults, [11,12]: (1) NP-hardness, (2) excessive (exponential) number of local extremes, (3) uneven distribution of local extremes, (4) deception points, (5) flat valley of extremes, (6) sequential character of calculations.

It becomes evident that the solution algorithm have to be adjusted or adapted to the type of landscape of the solution space in order to exploit fully acquired information about its structure, as well as uneven distribution of local extremes. Space landscape depends not only on the problem type, constraints on $\mathcal{X}$, form of $K(x)$, but also on the particular instance, i.e. data of the problem. Although the notion *landscape* is defined precisely, and can be perceived as "localization of solutions in the space depending on the distance among them", its refers indirectly to human's capability of interpreting 3D images in order to extract

an information supporting the search. Unfortunately, solution spaces are usually multi-dimensional, which implies that landscape features are hardly transformed into search directions, trajectory, convexity, etc. in 3D. One expects, that the detection of several recognized up to now properties of the landscape allow us to design better solution algorithms. Although several parameters characterizing search space and landscape were detected, described, examined, there is still lack of general rules how to extract the knowledge, what kind of knowledge is important and how to utilize the knowledge in the solution method, see among others [12].

There are at least a few approaches of collecting and utilizing the knowledge about the features of the solution space: (1) static, (1.1) arbitrary defined in advance, (1.2) found through a trial space sampling, (2) dynamic, (2.1) passive adaptive, (2.2) active self-learned. In (1.1), while designing the solution algorithm we set by expert some parameters in advance on the base his knowledge. Regarding (1.2), we perform three steps in order: (A) automatic analysing (e.g. by random sampling) the structure of the space; (B) calculating parameters and tuning solution algorithm; (C) searching solution with already set configuration of the algorithm. In (2.1) we collect data about solution space during the search and then use this knowledge to control searching process. Approach (2.2) keeps long-term memory to collect and extract knowledge about all problems and instances solved in the history and continuously realizes process of self-learning in the spare time of IaaS. There exist a lot of *intermediate* constructions located between these approaches. Beside the pure search over the solution space along trajectories, one can find a need of some auxiliary, accompanying calculations, such as space sampling, collecting search history, tuning, etc., fully justified and recommended for realizing in enhanced parallel and distributed calculating environments.

3 Ultramodern Approaches

The evolution of solution methods for CO problems has long, rich and gripping history, see Fig. 1. Although directions in the figure refer essentially to single-criteria case, they have an application to solution methods used in multiple-criteria problems.

The philosophy of used approaches and appreciation of their significance have changing over the span of years. The common trend observed in long-time period of time switches from universal exact methods (like ILP) to universal metaheuristics (like GA or SA), depending on the fashion. Between these extremely cases one can find numerous valuable particular solutions approaches with various solution technologies dedicated for a narrow classes of problems or even a narrow classes of instances of the same problem. The development of theory of NP-hardness allow us to set precisely the borderline between easy and hard problems. Algorithms for problems from the latter class usually employ special properties of the problem in order to improve algorithms' efficiency as much as possible. That's why a rich variety of the solution algorithms has been created

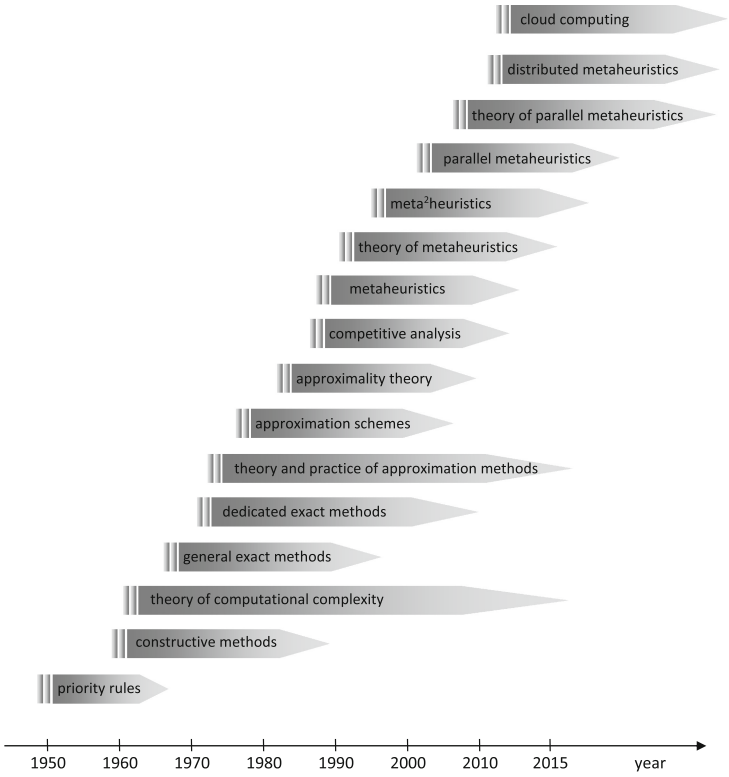

Fig. 1. Evolution of solving approaches in CO

and available now. "No free lunch" theorem clearly defines the possible areas of superiority of an algorithm over others.

Quite natural is to use model IaaS to reinforce virtual computational power by distributing calculations over virtual computing nodes distributed geographically widely. This approach requires possibility of making decomposition of CO into subproblems solved on independent nodes, however in cooperation. By using SaaS model, we have some choices: (1) homogenous engine with common interface (single method of solution) located in form of copies in calculation nodes, (2) various specialized algorithms located in different calculation nodes (own specialization) with common interface for various methods to solve the same problem.

Too high calculation cost observed for exact method forces us to use substitutes acceptable in practice. For a minimization problem *approximate algorithm* A provides solution x^A, so that

$$K(x^A) = \min_{x \in \mathcal{X}^A} K(x) \geq K(x^*) \tag{3}$$

where $\mathcal{X}^A \subset \mathcal{X}$ is the subset of solutions checked by A. The overall aim is to find x^A so that $K(x^A)$ is *close* to $K(x^*)$ by examining *the smallest as possible* $\mathcal{X}^A$. Notice, such definition is a hyper-bicriteria optimization case. The closeness to $K(x^*)$ (accuracy) can be either guaranteed a priori or evaluated a posteriori. It is clear that accuracy has opposing tendency to running time, i.e. finding better approximate solution needs longer running time (greater $\mathcal{X}^A$), and this dependence owns strongly nonlinear character. Therefore, discrete optimization manifests a variety of models and solution methods, usually dedicated for narrow classes of problems or even separate problems. Reduction of the generality of models allow us to find special features of the problem, application of which improve numerical properties of the algorithm such that running time, speed of convergence. Quite often, a strongly NP-hard problem has in the literature several various algorithms with different numerical characteristics. Knowledge about models and algorithms allow us to fit satisfactory algorithm for each newly stated problem. Bear in mind, in the considered research area the goal *is not* to formulate whatsoever model and method, but to provide *simply* model and solution method *reasonable* from the computer implementation point of view.

4 New Attitudes

Formulation (1) does not define details of calculating $K(x)$ for the given x. Because of the hardness of the most practical optimization tasks, one can expect that an approximate procedure, by selecting in $\mathcal{X}^A \subseteq \mathcal{X}$ a subset of non-dominated solutions, provides certain approximation of Pareto front. The cost of such calculations depends on the cardinality of $\mathcal{X}^A$ and the computational complexity of performing the basic step "for the given x find $K(x)$". In case of too high cost of calculations, one can either replace $K(x)$ by a cheapest its approximation $K'(x)$ or by limiting cardinality of $\mathcal{X}^A$. After an analysis we propose, by our previous paper [12], the following approaches: (1) x is deterministic, (1.2) $K(x)$ is given by an analytical formula, (1.2) $K(x)$ is given by a deterministic polynomial-time algorithm, (1.3) $K(x)$ is given by a deterministic exponential-time algorithm, (1.4) $K(x)$ is given by a deterministic algorithm provided in form of program code, (2) x is random variable, (2.1) $K(x)$ represents a statistical parameters of x or their estimators, (2.2) $K(x)$ is given by an algorithm, (2.3) $K(x)$ is a result of neural net activity, (3) x is fuzzy variable, (3.1) $K(x)$ represents certain defuzzified measure on x; (4) x is any variable, (4.1) $K(x)$ is given as the result of running simulation, (4.2) function $K(x)$ is given as the result of sensor measurement, (4.3) as previous one but in presence of noise. In any enumerated case $K(x)$ can be calculated precisely or approximately, depending on the computational complexity of required calculations, implying various requirements addressed for CC in aggregated or distributed environments.

5 Metaheuristics

New generation of metaheuristics, designed to work in parallel and distributed environments, offers powerful tool capable to overcome shortcomings observed

in traditional approximate approaches, [4,5,7]. Dynamic development of the network as well as the development of Cloud Computing (CC) technology offer new advantageous properties to build in the newly designed efficient solution methods. CC can be perceived as an infrastructure accessible transparently, remotely and virtually through the network, which eliminates the need for maintaining single-handedly expensive computing resources, [13], kept by the user. By sharing the physical resources (hardware, software, broadband, communication links, etc.) of the strong power, provided by CC operator, among large number of users, CC offers not only a reduced cost of service but also allows on rational management of computing goods in case of solving very hard problems derived from science and practice. One can say that CC offers: for scientists - a cheaper alternative to clusters, grids, and supercomputers to solve hard problems computationally expensive; for users - the powerful tool to solve quickly applicative problems for the commercial and business usage.

6 Parallel Metaheuristics

In recent years the increase of computational power of computers evolves towards parallel architectures. Since the increase of the number of processors or cores in single computer or CUDA is still too slow comparing it with the increase of the number of solutions in the space, there is no hope to vanquish barrier of NP-hardness. Even cloud computing with the use of computer clusters does not offer good alternative, chiefly because of too high calculation cost. On the other hand, computer parallelism can improve significantly metaheuristics in terms of running time and quality, [2,3]. Thus parallel metaheuristics become the most desired class of algorithms, since they link excellent quality with a short running time. Sophisticated implementations of parallel algorithms require skilful application of a few fundamental elements linked with parallel programming theory, calculation models, and practical tools, see [12].

Sequential metaheuristics can be implemented in parallel calculation environments in different manner, providing variety of algorithms with different numerical properties. Let us consider, for example, SA approach, [1]. We can adopt this method as follows: (a) single thread, conventional SA, parallel calculation of the goal function value, fine grain, conventional theory of convergence, (b) single thread, pSA, parallel moves, subset of random trial solutions selected in the neighbourhood, parallel evaluation of trial solutions, parallel theory of convergence, (c) exploration of equilibrium state at fixed temperature in parallel, (d) multiple independent threads, coarse grain, (e) multiple cooperative threads, coarse grain. Similarly, for GS we have [2,8]: (a) single thread, conventional GA, parallel calculation of the goal function value, small grain, theory of convergence, (b) single thread, parallel evaluation of population, (c) multiple independent threads, coarse grain, (d) multiple cooperative threads, (e) distributed subpopulations, migration, diffusion, island models.

7 Distributed Metaheuristics

Rapid development of computer networks offers inhomogeneous computational resources distributed geographically widespread, offering IaaS, PaaS or SaaS in different business model. Long list of solution approaches dedicated for CO problems implies existence of many alternative solution algorithms with different numerical characteristics even for the same CO case. Collection of these algorithms provide certain knowledge being the best up to now research results. In this context, net nodes can offer "highly specialized" software developed in scientific laboratories accessible in the SaaS technology. Developing of such software need less cost (since no repetitions occur) and provided a variety of solution methods.

CO problems require for solving commitment of high computing resources in the relatively short time intervals. Assuming that a market firm would like to use CO to improve the competitiveness, it solves various CO instances periodically, depending on needs. In order to obtain sufficient efficiency, firm has to buy high capacity, high availability, expensive workstation, which will be utilize in the average rather weakly. On the contrary, dispersed CC is able to provide computing resources of required power in required time interval on demand. Moreover, due to possibility of net connections, CC may locate computing tasks with green energy and low expenditures, however unreliable because of the net features.

CC offers in natural way the infrastructure to realize parallel computing (especially, parallel metaheuristics) for application of CO solution methods in the form of independent or cooperative searching threads, running on various virtual machines (VM) somewhere in the cloud, depending on current workload of real and virtual machines inside the cloud, user preferences (quality of expected service), financial expenditure versus profits from implementation of the best solution found. Notice, cooperative threads require a technology of exchanging messages between virtual machines. From this point of view CC offers IaaS by providing pure calculation power, possibly homogenous and transparent, for the run of multiple copies of searching algorithm. One hope that quality of the best solution found increases with the increased number of checked solutions. For the user it is irrelevant where individual threads run, and whether all of them hopefully have finished. Thus one can imagine an unusual scenario, where some of initiated threads simply perished somewhere in the net (their results are provided with a probability) because of communication obstacles or because of limited access to VM. In terms of parallel CO methods, the proposed approach realizes coarse grain models, skipping consciously fine grain case leading to very precise programming on a low level. These threads can be performed in different technologies enumerated at the end of the previous sections.

8 Conclusions and Comments

The given survey of optimization methodologies does not provide all details necessary to make an universal algorithm or a repository of algorithms. It rather

outlines crucial aspects important for the design and context of use of solution methods for the hard discrete optimization problems in the environment having rich variety of possible approaches. The present tendency prefer metaheuristics (sequencing as well as parallel) since they links high or good quality of generated solutions with relatively small or moderate calculation cost. Moreover they are resistant to local extremes. Real usefulness and applicability of each particular method depends on space landscape, roughness, big valley, distribution of solutions in the space and the problem balance between intensification and diversification of the search. Recent study suggests that efficient finding of Pareto from can be done by united force of a few different algorithms. If cost of calculations becomes high, for example for instances of greater size, there is recommended to consider parallel methods, possible to implement already on a PC with multicore processor or CUDA.

Acknowledgments. Paper is supported by funds of NCS, project DEC-2012/05/B/ST7/00102.

References

1. Aarts, E.H.L., van Laarhoven, P.J.M.: Simulated annealing: a pedestrian review of the theory and some applications. In: Deviijver, P.A., Kittler, J. (eds.) Pattern Recognition and Applications. Springer, Berlin (1987)
2. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. Wiley, Hoboken (2005)
3. Bozejko, W.: A New Class of Parallel Scheduling Algorithms. Oficyna Wydawnicza PWr, Wrocław, Poland (2010)
4. Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw Hill, Cambridge (1999)
5. Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Books, Bradford (2004)
6. Ghosh, A., Dehuri, S.: Evolutionary algorithms for multi-criterion optimization: a survey. Int. J. Comput. Inf. Sci. **2**(1), 38–57 (2004)
7. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Boston (1989)
9. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Struct. Multi. Optim. **26**, 369–395 (2004)
10. Nedjah, N., Coelho, L.S., de Mourelle, L.M. (eds.): Multi-objective Swarm Intelligent Systems. Studies in Computational Intelligence, vol. 261. Springer, Heidelberg (2009)
11. Smutnicki, C.: Optimization technologies for hard problems. In: Fodor, J., Klempous, R., Araujo, C.P.S. (eds.) Recent Advances in Intelligent Engineering Systems, pp. 79–104. Springer, Heidelberg (2011)
12. Smutnicki, C.: Optimization in CIS systems. In: Zamojski, W., Sugier, J. (eds.) Dependability Problems of Complex Information Systems. Advances in Intelligent Systems and Computing, vol. 307, pp. 111–128. Springer, Heidelberg (2015). doi:10.1007/978-3-319-08964-5_7
13. Vouk, M.A.: Cloud Computing - Issues, Research and Implementations. J. Comput. Inf. Technol. **16**, 235–246 (2008). doi:10.2498/cit.1001391