A multi-agent system for hybrid optimization

Eric S. Fragaa\*, Veerawat Udomvorakulchaia, Lazaros G. Papageorgioua

aSargent Centre for Process Systems Engineering, University College London (UCL), Gower Street, London WC1E 7JE, United Kingdom

e.fraga@ucl.ac.uk

Abstract

Optimization problems in process engineering, including design and operation, can often pose challenges to many solvers: multi-modal, non-smooth, and discontinuous models often with large computational requirements. In such cases, the optimization problem is often treated as a black box in which only the value of the objective function is required, sometimes with some indication of the measure of the violation of the constraints. Such problems have traditionally been tackled through the use of direct search and meta-heuristic methods. The challenge, then, is to determine which of these methods or combination of methods should be considered to make most effective use of finite computational resources.

This paper presents a multi-agent system for optimization which enables a set of solvers to be applied simultaneously to an optimization problem, including different instantiations of any solver. The evaluation of the optimization problem model is controlled by a scheduler agent which facilitates cooperation and competition between optimization methods. The architecture and implementation of the agent system is described in detail, including the solver, model evaluation, and scheduler agents. A suite of direct search and meta-heuristic methods has been developed for use with this system. Case studies from process systems engineering applications are presented and the results show the potential benefits of automated cooperation between different optimization solvers and motivates the implementation of competition between solvers.

**Keywords**: multi-agent system, hybrid optimization, direct search, meta-heuristic.

* 1. Introduction

Optimization problems in process engineering, including design and operation, can often pose challenges to many solvers. In particular, these problems may be multi-modal, non-smooth, and even discontinuous. In such cases, the optimization problem is often treated as a black box in which only the value of the objective function is required, sometimes with some indication of the measure of the violation of the constraints. Such problems have traditionally been tackled through the use of direct search and meta-heuristic methods. The challenge, then, is to determine which of these methods or combination of methods may be most appropriate. Further, these methods often have a number of parameters which affect their behaviour and choosing values for these parameters may be difficult to do without significant experimentation.

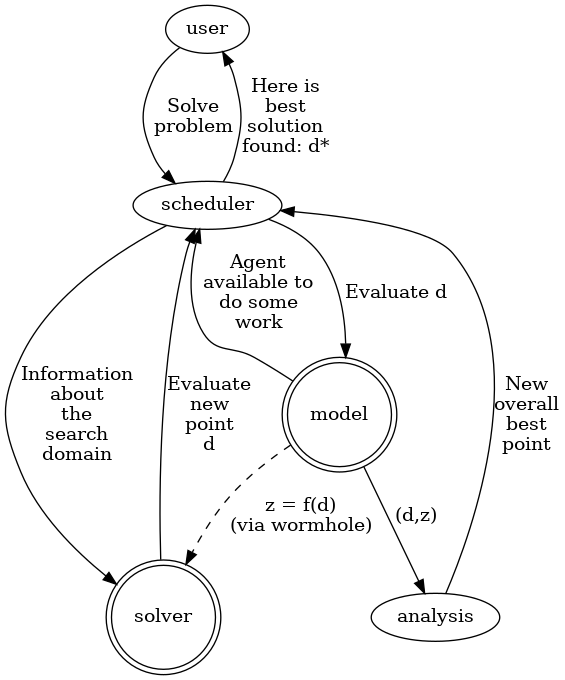
* 1. A multi-agent system

A multi-agent system is software that enables autonomous pieces of software, known as agents, to interact through the sending and receiving of messages (Nwana, 1996; Bradshaw, 1997). Multi-agent systems have been developed for engineering design problems, e.g. (Hanna and Cagan, 2008; Zhang et al., 2020) and references cited there-in. These previous implementations support cooperation by sharing information between solvers but do not provide the necessary control of model evaluation to enable an implementation for competition. The latter motivates a new multi-agent system implementation which decouples the optimization methods from the evaluation of the models. The system consists of the following agents:

* **solver**:a particular optimization method with specific parameter values which affect the method's behaviour;
* **model evaluation**:an agent which evaluates the model, consisting of the objective function and all constraints, given a point in the search domain;
* **scheduler**:an agent which accepts requests from solver agents for the evaluation of points in the search domain and allocates these requests to model evaluation agents; and,
* **analysis**: an agent which analyses all points evaluated and provides the scheduler with information about the search domain.

There will always be more than one solver agent as the aim is to explore the effect of cooperation and competition between different solvers. There may also be more than one model evaluation agent which is particularly useful to make use of modern multi-core computers for concurrent evaluation of different points.

The diagram in Figure 1 shows the multi-agent system along with the communications links between them, including both persistent links that exist throughout the execution of the agent system and ephemeral links (shown in a dashed line and labelled as a wormhole) which come into existence when required and disappear once used. The double circles around the model and solver agents indicate that there may be more than one instance of each of these agents. The user, i.e. the process systems engineer wishing to solve a particular optimization problem, is included in this diagram although obviously is not necessary a software agent.

 Figure 1: The agents in the agent based system including the scheduler, the model evaluation agents, the solver agents, and the analysis agent, showing the communications links between the agents. Solid lines are persistent communication links; the dashed line represents ephemeral links to receive the result of the evaluation of the model, f(d), at a given point, d, in the search domain. The best solution found is indicated by d\*.

At the centre of the multi-agent system is the scheduler. The main purpose of this agent is to process requests from the solvers for the evaluation of points in the search domain. When such a request is received, it is added to an evaluation queue. This queue is used to allocate the evaluation of these points when model evaluation agents become available. The availability of model evaluation agents is handled by a different queue. The current scheduling algorithm, for deciding which evaluation request to allocate to a model evaluation agent, when the latter is available, is based on the least recently used (LRU) algorithm used by operating systems to implement multi-tasking on single processor systems (Madnick and Donovan, 1974). This algorithm prioritises solvers which have not been given access to a model evaluation agent more recently than other solver agents.

The scheduler also enables sharing of information between solvers; in the first instance, it sends any new best solution found by any solver to all other solvers. It is up to the solvers individually to decide what to do with this information. The individual behaviour of the solvers and how they use this information is described in the next section.

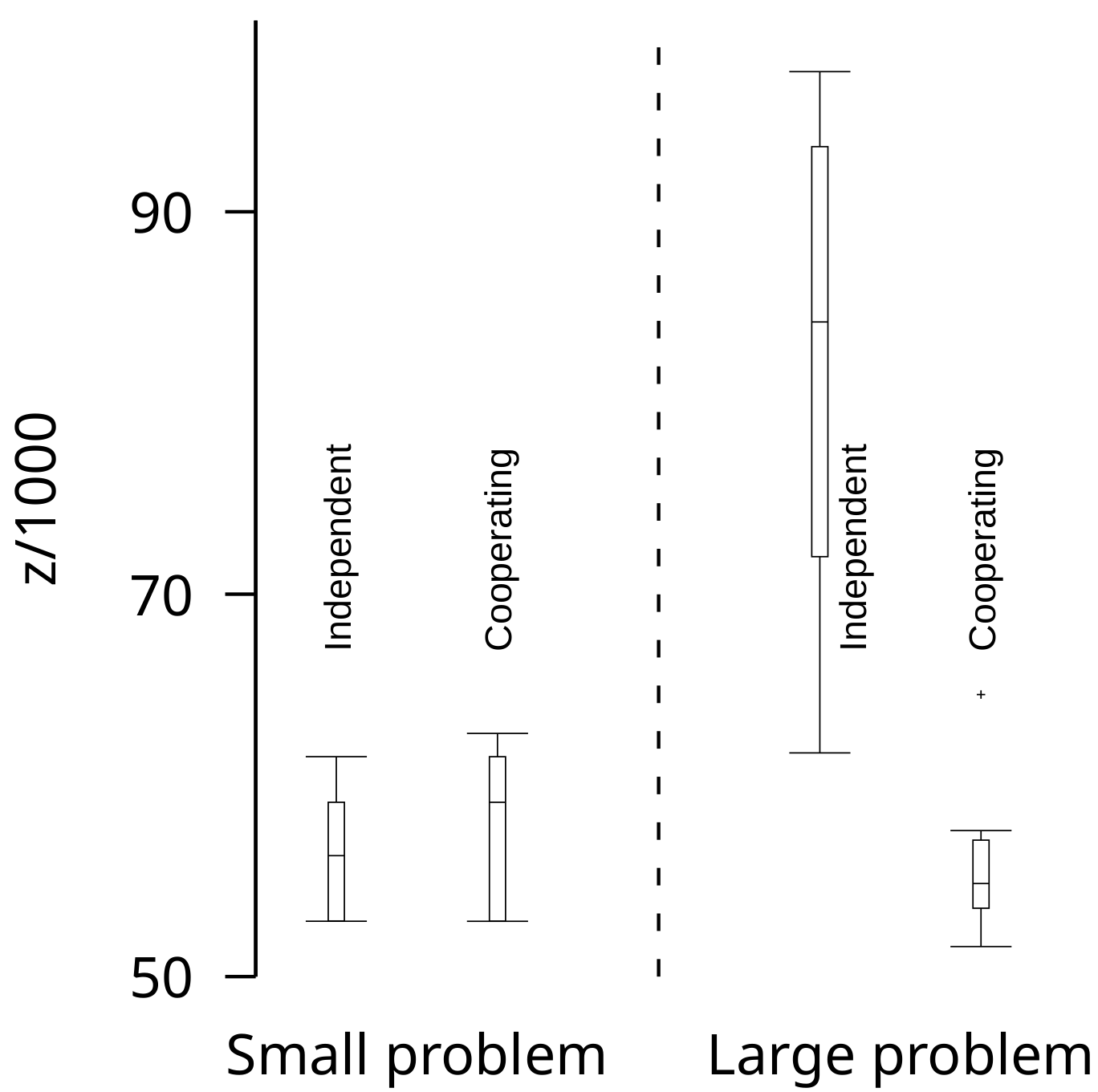
The multi-agent system has been implemented in the Julia language (Bezanson et al., 2017), a fast and modern programming language ideal for numerical analysis and computational science. Julia provides easy and transparent access to the multiple processors now typically available in modern multi-core desktop and laptop computers. Agents are easily created using the Threads.@spawn directive; communications links are instances of the Channel type. Messages between agents are encoded using our own format. Each message consists of the type of message, the agent that sent the message, and the content (which depends on the type of message). The results presented below are based on Julia 1.9.2. The resulting code is open source, portable, easy to use, and makes effective use of the computational processing capabilities available.

* 1. The solvers

Two types of solvers are considered: meta-heuristic methods inspired by nature (Fraga, 2022) and direct search methods (Kelley, 1999). A suite of methods has been implemented in the Julia language and the methods have been instrumented to enable the agent based system to share information when appropriate with each solver. The full list of solvers used in this paper follows:

* **genetic algorithm** (GA) inspired by Darwinian evolution (Holland, 1975) using crossover and mutation with fitness based selection to explore the solution space and exploit the better solutions. In what follows, the size of the population modelled is the key parameter that is explored by having multiple instances of this solver. This method incorporates new best solutions found by other solvers by simply adding the new solution to the population before the start of a generation. This is similar to the metaphor of multi-island genetic algorithms (Alba and Tomassini, 2002).
* **plant propagation algorithm** (PPA) based on the original paper (Salhi and Fraga, 2011) which is inspired by the propagation of strawberry plants using runners where the number of runners propagated by a solution is proportional to the fitness of that solution and the length of the runners inversely proportional to the fitness. This combination leads to a balance between exploitation and exploration of the solution space. The number of solutions to consider for propagation at each iteration is the key parameter explored. As with the genetic algorithm, new solutions sent by the scheduler are added to the population at the start of an iteration. In terms of nature inspiration, this could be considered as incorporating a further means of propagation used by plants: seeds carried over from other areas by animals or the wind.
* **steepest descent** is a gradient based hill-climbing method which has been implemented using finite differences for the numerical approximation of the gradient (Burden and Faires, 1989). The initial population is treated as a multi-start optimization problem where a search is started with each member of this population in turn. New solutions from the scheduler are added to the list of starting points yet to be attempted and inserted at the beginning of this list.
* **coordinate search** which searches along each dimension in turn, using a simple line search method (also used by the steepest descent method) (Kelley, 1999). The initial population and any points from the scheduler are treated as for the steepest descent method.
  1. Illustrative case study

Two case studies in the design of heat exchanger network synthesis are presented to explore the impact of the sharing of information to allow solvers to cooperate. One is a small example, with 2 hot streams and 2 cold streams, and the other is a larger problem with 5 cold and 5 hot streams based on case study 5 in (Fraga, 2009). The first is solved as an MINLP and the second as a pure NLP using a fully continuous representation of the superstructure (Fraga, 2006). Heat exchanger network design problems are combinatorial in nature and solutions often have a subset of design variables at the constraints. Previous work (Fraga, 2006) has shown that manually applying a sequence of methods can lead to better final outcomes. By using the multi-agent system, it will be interesting to see if and how the methods can cooperate to obtain good solutions.

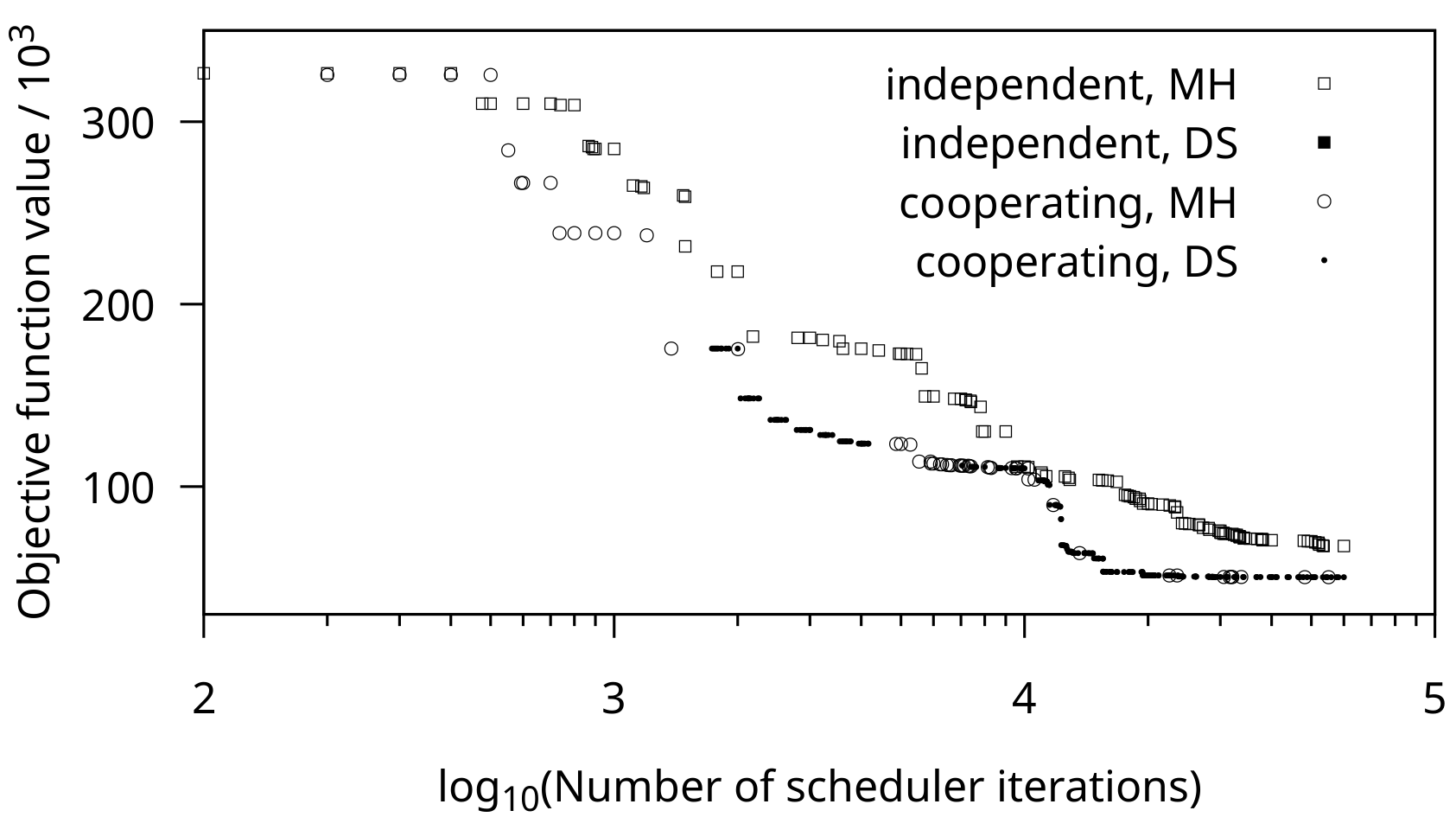
Figure 2: Box plot of the outcomes of 10 runs for both problems, comparing the outcomes for both when solvers are fully independent and when they are cooperating by sharing improved solutions when found.

The solvers considered are those described above with multiple instances of the meta-heuristic methods with different parameter settings. The base population size, np, is the number of design variables for both problems. This is adjusted for different instances of the solvers: for the GA are np (small) and 5 × np (large); for the PPA, the sizes are np/2 (small) and 2 × np (large). There is only one instance of each of the direct search methods. All methods are started with the same randomly generated population of size np with values of 10 for the smaller problem and 25 for the larger case.

The stopping criterion for the search is 60 thousand messages received by the scheduler agent, which can otherwise be thought of as 60 thousand iterations over the main loop of the scheduler. The schedule iteration loop considers not just function evaluations but also updates on the solution space by the analysis agent. The end result is that this stopping criterion equates to on the order of 5 thousand function evaluations by each of the 6 solvers.

Figure 2 presents the box plots of the variation of the objective function value for the best solution found over 10 runs, both when each solver works independently or when the solvers share any new better solution found. For the smaller problem, not sharing information actually leads to better outcomes, although only marginally. It would seem that sharing may lead to premature convergence in some cases. However, for the larger problem, sharing information leads to better solutions more quickly.

Figure 3 shows the evolution of the best solution found as a function of the number of scheduler iterations for both independent and cooperating cases. For the independent case, only the meta-heuristic methods are able to find good solutions. When the methods cooperate, both meta-heuristic and direct search methods are able to find improved solutions. It appears that when one of the meta-heuristic methods identifies a better solution, this solution is immediately improved by one of the direct search solvers. Although the methods are stochastic and repeated attempts will lead to different outcomes, the outcomes presented here are representative.

Figure 3: Evolution of best solution found, for the larger case study, as a function of the number of scheduler iterations indicating the solver responsible for the identification of the best solution to that point. The top trail of points is for when the solvers operate independently; the bottom trail is the outcome when the best solutions found are shared. MH is meta-heuristic and DS is direct search.

For the larger case study, the behaviour when solutions are shared is what we expected. The meta-heuristic methods are good at exploration but less effective at exploitation for constrained problems. The direct search methods have better exploitation properties. It should be noted, however, that the results obtained in this case study are not as good as the known optimum for this problem. This motivates the future work described below.

* 1. Conclusions

Optimization problems that arise in the process industries may often exhibit properties that preclude the use of mathematical programming, properties such as nonconvexity, nonlinearity, discontinuities, and differential equations. For those problems, a variety of meta-heuristic and direct search methods, that may treat the objective function as a black box, can be considered. However, there is a large choice of these methods, each of which may also have a large number of tunable parameters. As a result, choosing the appropriate method may be difficult a priori. Furthermore, each of these methods may have beneficial behaviour in different parts of the search space. These two factors motivate the development of a multi-agent system to enable a variety of methods, with different instances of many, to be applied to a problem simultaneously. The simultaneous application of multiple methods and instances of methods leads naturally to the idea of sharing information while the search progresses. This paper has shown that doing so leads to improved outcomes.

In the next steps, we wish to consider more complex scheduling algorithms in the scheduler agent and also explore what may happen when an element of competition is introduced where the solvers have to compete for the computational resource.

Acknowledgements. Mr. Veerawat is grateful for the support provided by the Royal Thai Government Scholarship, which covers his tuition fees and living stipends throughout the duration of his studies at University College London (UCL).

References

Alba, E.; Tomassini, M. Parallelism and Evolutionary Algorithms. IEEE transactions on evolutionary computation 2002, 6, 443–462.

Bezanson, J.; Edelman, A.; Karpinski, S.; Shah, V. B. Julia: A Fresh Approach to Numerical Computing. SIAM rev. 2017, 59, 65–98.

Bradshaw, J. M. An Introduction to Software Agents. Software agents 1997, 4, 3–46.

Burden, R. L.; Faires, J. D. Numerical Analysis; The Prindle, Weber and Schmidt series in Mathematics; Fourth.; PWS-Kent Publishing Company: Boston, 1989.

Fraga, E. S. Hybrid Methods for Optimisation. In Computer aided methods for optimal design and operations; Zilinskas, J.; Bogle, I. D. L., Eds.; World Scientific Publishing Co., 2006; pp. 1–14.

Fraga, E. S. A Rewriting Grammar for Heat Exchanger Network Structure Evolution with Stream Splitting. Engineering optimization 2009, 41, 813–831.

Fraga, E. S. Nature Inspired Methods for Optimization: A Julia Primer for Process Engineering; doi:10.5281/zenodo.7016482, 2022.

Hanna, L.; Cagan, J. Evolutionary Multi-Agent Systems: An Adaptive and Dynamic Approach to Optimization. Journal of mechanical design 2008, 131, 011010.

Holland, J. H. Adaptation in Natural and Artificial Systems; University of Michigan Press: Ann Arbor, MI, 1975.

Kelley, C. T. Iterative Methods for Optimization; SIAM, 1999.

Madnick, S. E.; Donovan, J. J. Operating Systems; McGraw-Hill Book Company, 1974.

Nwana, H. S. Software Agents: An Overview. The knowledge engineering review 1996, 11, 205–244.

Salhi, A.; Fraga, E. S. Nature-Inspired Optimisation Approaches and the New Plant Propagation Algorithm. In Proceedings of ICeMATH 2011, the international conference on numerical analysis and optimization; Yogyakarta, 2011; pp. K2:1–8.

Zhang, Z.; Han, Q.; Li, Y.; Wang, Y.; Shi, Y. An Evolutionary Multiagent Framework for Multiobjective Optimization. Mathematical problems in engineering 2020, 9147649.