Metaheuristics in Nature-Inspired Algorithms

Michael A. Lones
School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh, UK
m.lones@hw.ac.uk

ABSTRACT

To many people, the terms nature-inspired algorithm and metaheuristic are interchangeable. However, this contemporary usage is not consistent with the original meaning of the term metaheuristic, which referred to something closer to a design pattern than to an algorithm. In this paper, it is argued that the loss of focus on true metaheuristics is a primary reason behind the explosion of "novel" nature-inspired algorithms and the issues this has raised. To address this, this paper attempts to explicitly identify the metaheuristics that are used in conventional optimisation algorithms, discuss whether more recent nature-inspired algorithms have delivered any fundamental new knowledge to the field of metaheuristics, and suggest some guidelines for future research in this field.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Theory

Keywords

Metaheuristics, design patterns

1. INTRODUCTION

Metaheuristic is a widely used, but under-appreciated, term. In common usage, it is often used as a synonym for a search or optimisation algorithm. However, this was not the intention of the term's originator, who used it to refer to general ideas of how to carry out search, rather than specific implementations. A true metaheuristic is like a design pattern, in that it encapsulates knowledge that may be applied to the design of a range of specific optimisation algorithms. Likewise, there is no reason why a particular algorithm can not implement multiple metaheuristics, making use of complementary ideas of how to search for optima within solution landscapes.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2609841.

Over the last decade, there has been an explosion in the development of new nature-inspired optimisation algorithms [6], a practice which has received significant criticism [5]. Even if we ignore the more superfluous of these, we are still left with a seemingly diverse group of nature-inspired algorithms in common use. Add in more traditional derivative-free optimisation algorithms [1, 3] and the range of options may seem overwhelming for a practitioner trying to solve a particular problem. It is perhaps no wonder that people from outside the field of metaheuristics sometimes use algorithms which do not seem optimal for the task at hand.

It is arguable that the main problem is not the runaway invention of new algorithms, but rather the way in which new algorithms are presented. Close inspection reveals that many of these algorithms do introduce new ideas. However, by focusing on the modeling of the domain processes by which they are inspired, their authors often fail to identify the more general metaheuristics upon which these processes are based, and as a consequence, rarely relate this knowledge back to our wider understanding of how to carry out search. Hence, we get a proliferation of new algorithms, but no proliferation of new knowledge.

The main aim of this paper is to identify metaheuristics underlying nature-inspired optimisation algorithms. This is done by identifying the implicit metaheuristics used by these algorithms and, through reference to these metaheuristics, highlighting the relationships between different optimisation algorithms. The hope is that this will contribute to our wider understanding of metaheuristics and offer some guidance to practitioners overwhelmed by the expansive literature. It is also hoped that this paper will cast some light on recent developments in nature-inspired algorithms, and offer guidelines for future developments.

The paper is organised as follows[1]: beginning with a review of local search metaheuristics, this is followed by discussion of the most commonly used nature-inspired optimisation algorithms: evolutionary algorithms, particle swarm optimisation and ant colony optimisation. After this, some recent nature-inspired algorithms are reviewed, and related to the metaheuristics we identified in previous sections. The paper concludes by offering some guidelines for those looking to develop new nature-inspired optimisation methods.

[1]Note, due to limited space, some citations are made to collections and review papers rather than primary sources.

Neighbourhood Search

Intent: Find new solutions by exploring those that are a step change—*a move*—away from the current one. A move could be anything from flipping a single bit to randomly replacing the entire solution.

Motivation: To explore variants of a known solution.

Applicability: Whenever a suitable neighbourhood can be defined.

Examples: Particular heuristics can be generated by considering different neighbourhoods, and by sampling these neighbourhoods in different ways.

Hill Climbing

Intent: Follow a sequence of local improvements in order to find a locally optimal solution. A single move is performed at each step. If this leads to a better solution, the algorithm then moves on to explore a variant of this new solution, otherwise it remains at the original point and considers a different move.

Motivation: To improve upon an existing solution.

Applicability: Whenever neighbourhood search can be performed and where solutions can be assigned an objective value.

Accepting Negative Moves

Intent: Allow moves to worse solutions.

Motivation: To prevent convergence to local optima when carrying out hill climbing.

Examples: Implementations include threshold accepting and simulated annealing [1]. The latter reduces the probability of accepting negative moves as the search process approaches the global optimum.

Multi-Start

Intent: Restart the search process in a different region once it has converged at a local optimum. After this has been repeated a number of times, the best local optimum seen is returned.

Motivation: To explore the search space more widely.

Applicability: May not be suitable for large search spaces with sparse optima.

Examples: Restart points may be random. An alternative approach, used by iterated local search [1], is to estimate the distance between neighbouring optima and apply a move of this magnitude from the current local optimum.

Adaptive Memory Programming

Intent: Use memory of past search experience to guide future search.

Motivation: The original motivation was to prevent cyclic patterns of search when attempting to escape local optima.

Applicability: Requires memory. May not be applicable in highly dynamic search landscapes, where memory of previous search may be misleading.

Examples: Adaptive memory programming is a generalisation of ideas developed in tabu search. The simplest forms of tabu search maintain a short fixed-length FIFO list, which stores recent moves or types of moves. These moves are then prohibited until they leave the list. There are many variants of this idea, including the use of long term memory, and the use of aspiration criteria to revoke tabus [1]. Many nature-inspired algorithms, including evolutionary algorithms, also carry out a form of adaptive memory programming, using a population of search points as their memory.

Figure 1: Local search metaheuristics

2. LOCAL SEARCH ALGORITHMS

Before considering nature-inspired algorithms, it is first instructive to review some of the common metaheuristics used by more traditional optimisation algorithms (see Fig. 1). Most of these are local search algorithms, which consider a single search point at a time during the search process. Fundamental to all these algorithms is the **neighbourhood search** metaheuristic. Many local search algorithms are concerned with finding trajectories that lead towards local optima. A general metaheuristic for achieving this is **hill climbing**. However, in most cases the local optimum will not be the global optimum. To address this, there are a number of metaheuristics that can be used to encourage wider exploration of the search space. Two simple, and widely used, ones are **accepting negative moves** and **multi-start**. Another popular method, and the originator of the term metaheuristic, is tabu search. This is now often referred to using the more general term **adaptive memory programming**.

3. EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EAs) [1, 4] are an abstract model of biological evolution, in which a population of candidate solutions is iteratively exposed to analogues of natural selection and genetic variation. The basic procedure is as follows: the initial population is a random sample of search points, a selection mechanism then discards search points with poor objective value, and variation operators derive new search points from those that remain. This new population then replaces the previous population, and the process of selection and variation is repeated until an optimal solution is found, or some other termination criterion is met.

Interestingly, EAs use all the metaheuristics described in the previous section. Mutation carries out a **neighbourhood search**, elitism supports **hill climbing**, stochastic selection means that the algorithm can **accept negative moves**, the use of many starting points resembles **multi-start**, and the population uses previous search points to guide future search, making it a form of **adaptive mem-**

Population-Based Search

Intent: Multiple, cooperating search processes that are typically executed in parallel.

Motivation: The sharing of information at intermediate stages of search helps guide convergence to global optima.

Applicability: Usually less efficient than local search when applied to relatively easy problems.

Examples: In an EA, sharing is implemented indirectly through the selection mechanism, which spreads information about relative fitness, and directly through the recombination operator. Whilst EAs popularised this metaheuristic, it is also also used by other optimisation algorithms—such as scatter search [1]—which are not modelled upon natural processes.

Intermediate Search

Intent: Explore the region between two or more previously visited search points, each of which is known to have a relatively high objective value.

Motivation: To explore solutions with properties derived from more than one good solution. Originally motivated by genetic crossover in biology.

Applicability: Particular implementations may not be suitable for particular search spaces.

Examples: Recombination, or crossover, is the best known example of this metaheuristic. A related approach, termed path relinking [1], has more recently been introduced to tabu search and scatter search. This is to some extent a generalisation of crossover that involves finding new (relinked) paths between known solutions and exploring the neighbourhood of this path.

Figure 2: EA metaheuristics

Directional Search

Intent: Identify productive directions within the search space, and then carry out moves accordingly.

Motivation: For PSO, animal foraging behaviours.

Applicability: PSO requires a metric space. Gradient methods require a differentiable objective function.

Examples: In PSO, productive search directions are estimated from the locations of good solutions (also resembling intermediate search). Directional search may also be guided by other sources of knowledge. For example, gradient ascent uses derivatives of the objective function to identify productive search directions. Where the objective function is not known, estimates of its derivatives can be used. An example of this is the CMA-ES algorithm [4], an evolution strategy that estimates second derivatives, using this knowledge to bias the mutation operator.

Variable Neighbourhood Search

Intent: Search different neighbourhoods around the location of a known local optimum.

Motivation: The local optimum in one neighbourhood may not be the local optimum in another.

Applicability: Requires multiple neighbourhoods.

Examples: The variable neighbourhood search algorithm [1] does this by iteratively increasing the neighbourhood size, exploring moves of increasing magnitude from the current local optimum. Although PSO takes a less structured approach, different particles approaching the current best known search point will tend to have different velocities, in effect exploring neighbourhoods of different magnitude around this point.

Figure 3: PSO metaheuristics

ory programming. However, EAs are also notable for introducing a new metaheuristic: **population-based search** (Fig. 2). The use of recombination in EAs is also distinctive, and has likewise promoted interest in a more general metaheuristic, the idea of **intermediate search** (Fig. 2).

4. PARTICLE SWARM OPTIMISATION

Particle swarm optimisation (PSO) [4] models group foraging. As for an EA, an initial population of search points is randomly sampled. Unlike an EA, each search point is explicitly associated with a search process, which also has a velocity and a memory of the best point it has seen so far. For each iteration, each search process updates its velocity so that its path veers slightly towards the best solutions seen by a subset of the search processes. It then applies this velocity to its current search point in order to derive a new search point. This method of exploring new search points continues until a termination criterion is met.

PSO has significant commonalities with EAs, including the use of **population-based search** and **intermediate search**. However, arguably the most distinctive feature of PSO is the use of **directional search** (Fig. 3). A further distinctive aspect of PSO is how it extends directional search with the notion of search velocity. In the original PSO paper, this is described as a means of promoting exploration over exploitation when carrying out directed search, primarily by allowing a particle to overshoot its target. However, it can also be seen as a combination of directional search with another metaheuristic that has recently developed some popularity, **variable neighbourhood search** (Fig. 3).

5. ANT COLONY OPTIMISATION

Ant colony optimisation (ACO) [1] is motivated by the way in which ants share their foraging experience. The idea is that search processes mark paths through the search space that lead to regions of high objective value. Marking up is done using an analogue of pheremone concentration, typically in three ways: the path that leads to the best solution is reinforced to promote followers, paths traversed by search processes are decayed to promote diversity, and all paths decay through evaporation to allow forgetting.

ACO revisits a number of the metaheuristics already discussed, notably **population-based search** and **directional search**. However, the idea of marking up the search space (often referred to as stigmergy) is distinctive. Whilst it could be described as **adaptive memory programming**, it also

Search Space Mapping
Intent: Construct a map to guide search processes that are traversing the search space.
Motivation: For ACO, ant foraging behaviour.
Applicability: Standard ACO is only applicable to discrete optimisation, though generalisations do exist.
Examples: In the case of ACO, the map takes the form of a probabilistic overlay that points towards regions of productive search. An overlay approach is also used in guided local search [1], where it modifies the underlying objective values, pointing search away from solutions with features considered undesirable. Another approach to mapping, used by branch-and-bound methods such as DIRECT [3], is to divide the search space into partitions, directing search towards partitions known to contain fit solutions.

Figure 4: ACO metaheuristic

represents a more specific metaheuristic principle concerned with **search space mapping** (Fig. 4).

6. EMERGING ALGORITHMS

EAs, PSO, ACO, and their variants, still dominate the field of nature-inspired optimisation. However, many recent nature-inspired algorithms have developed a substantial following. Some of the most popular include:

The artificial bee colony algorithm (ABC) [6] (>1000 citations) models honey bee foraging. It explores a fixed number of regions within the search space at any one time. The number of search processes in each region is determined by relative fitness. Search continues in a region whilst fitter solutions continue to be found; otherwise, a new region is randomly sampled.

Bacterial foraging optimization (BFO) [2] (>1000 citations) models *E. coli* colonies. Like PSO, it uses a population of search processes. Each performs a random walk composed of straight segments followed by random direction changes. Segment length is proportion to search gradient, so bigger steps are taken in productive directions. Objective values are overlaid by a crowding term, whose effect is to draw the random walks towards one another. A dispersal routine randomly reallocates a subset of search points.

Firefly algorithm (FA) [6] (>500 citations) is similar to PSO, and loosely motivated by the grouping behaviour of fireflies. Compared to PSO, the main differences are the lack of search velocities and the use of an inverse-square law to guide interactions between all search processes. This leads to more localised interactions and a consequent tendency to form multiple regions of search.

Cuckoo search (CS) [6] (>500 citations) uses a small population of solutions. At each step, a poor solution is replaced either randomly or by using a 'Lévy flight' applied to another, randomly selected, solution. Lévy flights are a kind of random walk with step sizes generated from a heavy-tailed probability distribution that has been shown optimal for exploring sparse foraging environments.

Each of these algorithms carries out a **population-based search**. ABC and CS make explicit use of **hill climbing** and **multi-start** strategies, and are the only algorithms not to use **intermediate search** or **directional search** metaheuristics. BFO carries out both a **variable neighbourhood search** and **search space mapping**. Random walks are used in both BFO and CS, and could be considered as a novel metaheuristic. FA, on the other hand, has little to distinguish it from PSO, with the inverse-square law having a similar effect to crowding and fitness sharing in EAs, and the use of multi-swarms in PSO. However, there is scope for recognising this general group of search balancing mechanisms as a metaheuristic, since they are often central to solving multimodal problems.

7. DISCUSSION

Arguably the main contribution of recent nature-inspired algorithms has not been to develop new metaheuristics, but rather to investigate new ways of combining metaheuristics that we already know about. However, this is typically done in an *ad hoc* fashion, with no explicit identification of the metaheuristics being used, and little understanding of why they are being combined. To address this issue, we might suggest the following considerations for the future:

Explicitly identifying metaheuristics would help spread knowledge, bring cohesion to the field, and make algorithmic descriptions more accessible. This paper is a small step in this direction, but more work is required to identify and express the metaheuristics in current use and the relationships between them.

When to combine metaheuristics is currently unclear. It is likely that different combinations are suitable for different problems. However, there is limited theoretical understanding, and experimental studies are generally done at the algorithmic level. Again, explicit identification of metaheuristics would help to develop this knowledge.

Hyper-metaheuristics could build upon the success of hyperheuristics [1], automatically evaluating different combinations of metaheuristics to produce new general-purpose algorithms, and addressing the *ad hoc* manner in which combinations are currently explored.

Design patterns, whilst distinct from metaheuristics, help to formalise implementations and mechanisms for combining metaheuristics [4](ch. 29), promoting thinking at the metaheuristic level. The documentation conventions of design patterns could also be adapted to metaheuristics, providing a method for standardisation and expressing ontologies.

8. REFERENCES

[1] M. Gendreau and J. Y. Potvin. *Handbook of Metaheuristics*. Springer, 2nd edition, 2010.

[2] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE*, 22(3):52–67, June 2002.

[3] L. Rios and N. Sahinidis. Derivative-free optimization. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[4] G. Rozenberg, T. Bäck, and J. N. Kok. *Handbook of Natural Computing*. Springer, Berlin, Heidelberg, 2012.

[5] K. Sörensen. Metaheuristics—the metaphor exposed. *Intl. Trans. in Op. Res.*, Feb. 2013. online version.

[6] X. S. Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2nd edition, 2010.