Search Bias in Constructive Metaheuristics and Implications for Ant Colony Optimisation

James Montgomery[1⋆], Marcus Randall[1], and Tim Hendtlass[2]

[1] Faculty of Information Technology, Bond University, QLD 4229, Australia
{jmontgom, mrandall}@bond.edu.au
[2] School of Information Technology, Swinburne University, VIC 3122, Australia
thendtlass@swin.edu.au

Abstract. Constructive metaheuristics explore a tree of constructive decisions, the topology of which is determined by the way solutions are represented and constructed. Some solution representations allow particular solutions to be reached on a greater number of paths in this construction tree than other solutions, which can introduce a bias to the search. A bias can also be introduced by the topology of the construction tree. This is particularly the case in problems where certain solution representations are infeasible. This paper presents an examination of the mechanisms that determine the topologies of construction trees and the implications for ant colony optimisation. The results provide insights into why certain assignment orders perform better in problems such as the quadratic and generalised assignment problems, in terms of both solution quality and avoiding infeasible solutions.

1 Introduction

An implicit assumption when using any metaheuristic is that it offers relatively unbiased access to all parts of the solution space, provided that deliberate bias towards "good" solutions is removed. That is, if search decisions are made in an undirected fashion (i.e., randomly) then each solution has approximately equal probability of being found. Of course, all common metaheuristics *are* biased towards solutions that appear promising. The neighbourhood in which constructive metaheuristics search forms a tree of constructive decisions, or *construction tree*. The nature of this construction tree can introduce a bias to the search. Constructive metaheuristics such as Ant Colony Optimisation (ACO), which use previously generated solutions to learn appropriate features to include in future solutions, operate essentially randomly during the early stages of a run. Thus, any bias that affects the undirected construction of solutions may reduce the effectiveness of the learning mechanism employed.

Research is this area is largely restricted to the work of Blum [2] (previously published in Blum and Sampels [3, 4] and Blum, Sampels and Zlochin [5]). These studies have provided an investigation of *model bias*, or the bias introduced by

⋆ Corresponding author

the interaction between a particular pheromone representation and problem constraints. The focus on how frequently particular pheromone values are updated for a given problem–pheromone combination does not fully take into account underlying biases in the constructive approach that may unfairly advantage some solutions.

This paper investigates how a chosen solution representation and construction mechanism combine to bias constructive metaheuristics. The interaction between these biases and different pheromone representations is not discussed, although these issues are addressed by Montgomery [12]. Thus, it is largely complementary to the work of Blum [2]. Section 2 considers the underlying sources of bias that act on a constructive heuristic, while Section 3 summarises the experimental work undertaken to investigate this issue. Section 4 discusses some implications of these findings for the ACO approach.

2 Constructive Metaheuristics and Bias

Constructive metaheuristics take an empty solution ($\varnothing$) and successively add *solution components* to build a complete, typically feasible, solution to the problem at hand. The nature of the solution components depends on the problem specification. For instance, in the travelling salesman problem (TSP), solution components are typically cities (see e.g., [8]), which are successively added to $\varnothing$ to produce a complete solution (i.e., a permutation of the cities). Hence, these metaheuristics explore a tree of constructive decisions, or construction tree, where the root corresponds to $\varnothing$ and leaves correspond to complete (or infeasible partial) solutions. At the heart of such metaheuristics is the constructive algorithm (hereafter denoted by $\mathcal{A}$) used to define solution components and the mapping from sequences of solution components (i.e., paths in the construction tree) to solutions. The construction tree defined by $\mathcal{A}$ is denoted by $\mathcal{T}_{\mathcal{A}}$. A distinction must be made between a sequence of solution components $\mathfrak{s}$ and the solution to which it corresponds $s = X_{\mathcal{A}}(\mathfrak{s})$, where $X_{\mathcal{A}}$ is a mapping from sequences to solutions. The set of solutions corresponding to a solution is denoted by $X_{\mathcal{A}}^{-1}(s)$.

Five common combinatorial optimisation problems are used throughout this paper to illustrate bias in constructive metaheuristics: TSP [8], subset problems such as the multiple knapsack problem (MKP) [9], quadratic assignment (QAP) [11], generalised assignment (GAP) [10], and permutation scheduling problems such as the job and open-shop scheduling problems (JSP and OSP respectively) [3]. The constructive algorithm used in each is that used most commonly in the respective ACO algorithms for these problems.

When constructive decisions are undirected two primary sources of bias may be identified: *representation* bias and *construction* bias. The term *undirected* is used to indicate that the constructive algorithm in question makes each constructive decision probabilistically using a uniform random distribution over the available choices at each step. However, it is assumed that constraints are still enforced such that at each step no options are available that would make the

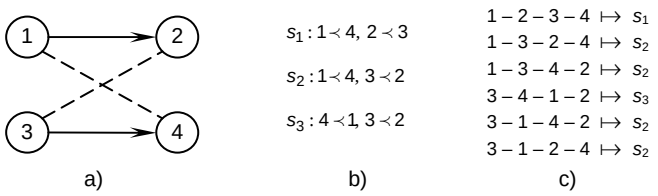

Fig. 1. Adapted from Blum and Sampels [4]. a) A small JSP instance; directed arcs indicate required order of operations within each job, dashed lines indicate operations that require the same machine. b) The three solutions to this problem described in terms of the relative order of operations that require the same machine, where $i \prec j$ indicates i is processed before j. c) The six possible solution representations an ACO algorithm may produce and the solutions to which they correspond

partial solution infeasible. All constructive algorithms discussed in this section are considered to be undirected.

The nature of the problem representation used may allow distinct solutions to be represented in multiple ways. In many problems, the number of representations per solution is not uniformly distributed. Consequently, in ACO, some solutions will be overrepresented in the representation space in which ants search. For instance, solutions to many machine scheduling problems such as the JSP and OSP are represented as permutations of the operations to be scheduled. As solutions are uniquely described in terms of the relative order of operations that require the same machine (or that are part of the same job in the OSP), some operations may be exchanged in a permutation without changing the solution represented. Consider the JSP depicted in Fig. 1. There are three distinct solutions to this problem, yet six feasible representations. Of these, four correspond to solution s_2, which accordingly appears to have a $66\frac{2}{3}\%$ probability of being discovered by an undirected search, twice that expected if each distinct solution could be found with equal probability.

Definition 1. *A constructive process $\mathcal{A}$ applied to a given combinatorial optimisation problem is said to have a* representation bias *if there exist two solutions s_1 and s_2 such that $|X_{\mathcal{A}}^{-1}(s_1)| \neq |X_{\mathcal{A}}^{-1}(s_2)|$.*

Fig. 2 depicts the possible paths an ant may take to produce feasible solutions to the JSP described in Fig. 1 when representing solutions as permutations of the operations. The probability of choosing a particular component at a given node in the tree is inversely proportional to the number of alternative components at that node. If there are no infeasible sequences defined by $\mathcal{A}$, then the degree of branching at each level in the tree will be uniform within that level. This is the case, for example, in the TSP and QAP, where all permutations of cities or facilities represent feasible solutions. In problems where some solution representations correspond to infeasible solutions, the degree of branching within each level will not be uniform, as in the JSP. Consequently, solutions found on paths with less branching are more likely to be discovered than those on paths with

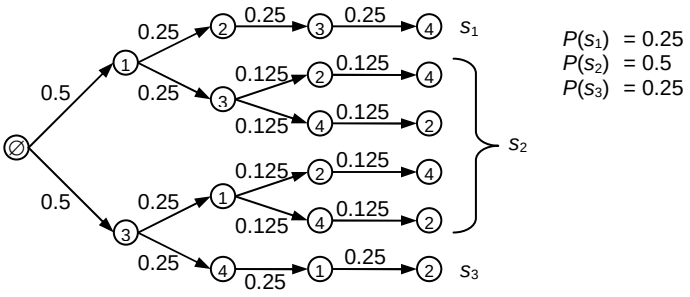

Fig. 2. Construction tree for small JSP instance (see Fig. 1 for problem description). Arcs are labelled with the probability of their being traversed given an undirected construction process. End points are labelled with the solution represented by that path. Aggregate solution probabilities appear on the right

more branching. Hence, decisions that push the solution closer to the boundaries of feasible space lead to solution representations with a higher probability of being found. This constitutes a *construction bias*.

Definition 2. *A construction tree $\mathcal{T}_\mathcal{A}$ has a* construction bias *if there exist two nodes in $\mathcal{T}_\mathcal{A}$ such that their heights are equal yet their degrees are not equal.*

The two biases interact. For instance, in the JSP depicted in Fig. 1, although solution s_2 has $66\frac{2}{3}\%$ of all solution representations, Fig. 2 shows it has only a 50% probability of being found. This is because the distribution of paths in the tree corresponding to each representation determines the actual likelihood of constructing the solutions represented.

An interesting form of construction bias exists in problems where sequences are of variable length—shorter paths typically have fewer branching points and consequently have an elevated probability of being traversed. For example, in subset problems such as the MKP, feasible solutions are of varying lengths. Analysis of the construction tree for small instances confirms that a sequence's probability is inversely proportional to its length. However, such problems also have a representation bias (assuming sequences are built from the individual items that make up the subset) where each solution of size k is represented by $k!$ sequences in the construction tree. Further analysis reveals that the representation bias in these problems will always dominate.

In highly constrained problems such as the GAP partial sequences may be constructed that cannot be completed to produce a feasible solution. In the absence of backtracking these must then be abandoned.[3] Given such infeasible partial solutions are by definition found on shorter paths than feasible solutions, they have an elevated probability of being discovered. Furthermore, the more constrained a problem, the shorter will be the paths that lead to infeasible solutions. This issue is of particular interest in assignment problems, where the

[3] Some algorithms admit infeasible solutions, often relying on an accompanying local search to transform these into feasible solutions (e.g., Lourenço and Serra [10]).

nature of the constructive algorithm used can reduce the probability of reaching infeasible solutions, an issue discussed in Section 2.2.

2.1 Assignment Problems and Assignment Order

In problems involving assignment of *items* to *groups* (i.e., facilities to locations in the QAP, jobs to agents in the GAP) there exists a choice over the order in which items are assigned. While the construction tree is defined by the constructive algorithm used to solve the TSP, MKP and JSP, in assignment problems the choice of assignment order partly determines the topology of the tree. The solutions represented remain unchanged. Solutions that share much of their respective paths in the construction tree under one assignment order may diverge much earlier under another. In effect, the assignment order determines the constructive neighbourhood in which solutions are found.

In problems with no representation or construction bias, such as the QAP, different assignment orders will not alter solutions' respective probabilities. However, they will change which solutions are neighbours (in the construction tree) and hence may produce differing results in directed algorithms such as ACO. ACO algorithms for the QAP have taken a variety of approaches to determining assignment order, including selecting items randomly [15] and predetermining an order such that facilities with high flow requirements are assigned early with the intention they are assigned relatively central locations [11].

2.2 Assignment Order and Infeasible Space

In problems which do have infeasible solution representations, different assignment orders not only redistribute solutions in the construction tree, but may also alter their respective probabilities. Decisions that take partial solutions closer to infeasibility increase the probability of the (possibly infeasible) solutions to which they correspond. Consequently, the earlier such decisions are made, the greater the increase in probability. By altering the assignment order such decisions may be moved to any level in the construction tree, thereby altering their respective solutions' probabilities. In problems where feasible solutions are not guaranteed, choosing an appropriate assignment order is thus a significant issue.

A good construction tree topology for these problems is one in which the probability of discovering a feasible solution is maximised. However, finding an assignment order that produces such a tree is non-trivial. A static order fixes the topology of the construction tree and so also fixes the probability of infeasible representations. In contrast, a dynamic random order allows a range of construction trees to be used at various times in the algorithm and will likely be superior to an arbitrary static order. More commonly, assignment orders are chosen heuristically, such as in one ACO algorithm for the GAP developed by Randall [13].

Assigning highly constrained items early will likely produce trees with shorter paths leading to infeasible solutions, which consequently have an elevated probability relative to the proportion of paths they represent. However, using such

an assignment order the total number of paths leading to infeasible solutions is reduced, as decisions that lead to infeasibility are consolidated nearer the root of the construction tree. Although a small number of infeasible paths does not guarantee a high probability of reaching a feasible solution our empirical studies show that trees with fewer infeasible paths typically have a higher probability of reaching a feasible solution than those with more infeasible paths. Indeed, a commonly used static assignment order assigns highly constrained items (e.g., jobs with high resource requirements in the GAP) early [10, 14]. Empirical testing reveals that simple heuristics for determining a static assignment order are often not the best. Accordingly, Costa and Hertz [6] consider a number of static and dynamic assignment orders for the graph colouring problem based on heuristics related to the principle of assigning more highly constrained items early.

3 Experimental Investigation

This section presents a summary of our empirical investigation of bias in five common combinatorial problems: Symmetric TSP [8], MKP [9], group-shop scheduling problem (GSP) (a generalisation of the JSP and OSP) [3], QAP [11] and GAP [10].[4]

Complete exploration of the construction trees for small instances was performed, collecting data concerning the existing distribution of costs in each instance and the expected probability of solutions given an undirected constructive algorithm (referred to as $\mathcal{A}_{undir}$).[5] Results were confirmed by comparing them against the frequency with which solutions were found by an implementation of $\mathcal{A}_{undir}$. The two analyses confirmed that the TSP and QAP have no inherent bias. In the MKP, only on the most trivial instance studied did a small solution show an elevated bias over others, further suggesting that in this problem the representation bias dominates.

The GSP and JSP show a mixture of representation and construction biases, with further analysis revealing that high probability sequences correspond to under-represented solutions and vice versa. This is an interesting property of these problems when solutions are represented as permutations of the operations, whereby those sequences with the highest probability are also those that are least able to be perturbed without changing the solution represented (see Montgomery [12] for full details).

Large problem instances were studied for all problems except the TSP using $\mathcal{A}_{undir}$.[6] On these instances, the effects of any underlying bias could not be observed, as $\mathcal{A}_{undir}$ can only take a sample from the very large space of sequences and solutions to these instances.

[4] Full results are available on request from the corresponding author.

[5] Small instances consisted of up to 14 cities in the TSP, 20 items in the MKP, 11 operations in GSP, 12 locations in the QAP, and 5 agents, 20 jobs in the GAP.

[6] Large instances consisted of up to 100 items in the MKP, 225 operations in GSP, 256 locations in the QAP, and 10 agents, 60 jobs in the GAP.

3.1 Assignment Order in the QAP and GAP

Given that the QAP has no inherent bias, changing the assignment order should have no impact on the frequency with which different solutions are found using $\mathcal{A}_{undir}$. Applying $\mathcal{A}_{undir}$ with a range of assignment orders to various QAP instances confirmed this.

In contrast, construction trees for all but the smallest contrived GAP instances contain infeasible partial solutions, the number and distribution of which is determined by the assignment order. Although the effects of representation and construction bias are not evident in large MKP and GSP instances, the effects of different assignment orders were observed in large GAP instances. Analysis was made of the construction trees for every static assignment order for a trivial instance with 3 agents and 8 jobs (adapted from a 5 agent, 15 job instance from the `gap1` problem set, available from the OR-Library [1]). Across most assignment orders, the probability of producing infeasible solutions was elevated above what would be expected given the total number of infeasible paths in the construction tree, in some cases by more than 35%. This is in line with predictions made in Section 2.2. The probability of reaching a *feasible* solution when the number of paths leading to infeasible solutions is minimal was found to be better than the probability under the worst assignment order (13% versus 3%). However, the highest probability (34%) was shown under an another assignment order in which the number of such paths was low, but not minimal. Under this assignment order, the probability of producing infeasible solutions was *below* what would be expected given their number. Furthermore, the best assignment order did not have jobs in non-increasing order of constrainedness, which had a 16% probability of producing feasible solutions.

Sampling of randomly generated static assignment orders for instances from the `gap1` (5 agents, 15 jobs) and `gap2` (5 agents, 20 jobs) problem sets [1] reveals that very few assignment orders can produce a relatively high probability of finding feasible solutions. Comparison with the construction tree produced by a static assignment order in which jobs appear in non-increasing order of constrainedness indicates that it typically produces a higher probability of finding feasible solutions than 90% of alternative static orders for `gap1` instances (although on one instance it was better than only 24% of sampled orders) and 80% of some `gap2` instances. Interestingly, several different dynamic assignment orders based on the same heuristic of assigning more highly constrained jobs early did not produce a higher probability of feasibile solutions in general.

4 Implications for ACO

The nature of the constructive process is such that solution representation and problem constraints may introduce a bias into any search process that uses it. Constructive metaheuristics such as ACO use heuristic information and accumulated pheromone information to adapt their searches towards promising solutions. Additionally, almost all current ACO algorithms use a local search

procedure to improve solutions produced by the ACO algorithm [7]. Each of these serves to counteract any underlying bias in the constructive process.

In assignment problems such as the QAP and GAP, the assignment order can alter the distribution of solutions within the construction tree. This may impact on the performance of ACO.

Infeasible space is a problem for any metaheuristic, but may be particularly so for constructive techniques as infeasible solutions individually have an elevated probability of being found. In problems involving assignment, the assignment order can alter the number and distribution of infeasible solutions in the construction tree. Heuristically determined static assignment orders can reduce the probability of reaching an infeasible solution, but may not necessarily produce the best construction tree.

References

[1] J. E. Beasley. OR-library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.*, 41:1069–1072, 1990.

[2] C. Blum. *Theoretical and practical aspects of ant colony optimization.* PhD dissertation, Université Libre de Bruxelles, 2004.

[3] C. Blum and M. Sampels. Ant colony optimization for fop shop scheduling: A case study on different pheromone representations. In *Proceedings of CEC 2002*, pages 1558–1563, 2002.

[4] C. Blum and M. Sampels. When model bias is stronger than selection pressure. In *Proceedings of PPSN-VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 893–902. Springer-Verlag, Berlin, 2002.

[5] C. Blum, M. Sampels, and M. Zlochin. On a particularity in model-based search. In *Proceedings of GECCO 2002*, pages 35–42, New York, 2002.

[6] D. Costa and A. Hertz. Ants can colour graphs. *J. Oper. Res. Soc.*, 48:295–305, 1997.

[7] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.*, 1(1):53–66, 1997.

[8] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Sys. Man Cyb. B*, 26(1):1–13, 1996.

[9] G. Leguizamón and Z. Michalewicz. A new version of ant system for subset problems. In *Proceedings of CEC 99*, pages 1459–1464, 1999.

[10] H. R. Lourenço and D. Serra. Adapative search heuristics for the generalized assignment problem. *Mathware Soft Comput.*, 9(2):209–234, 2002.

[11] V. Maniezzo and A. Colorni. The ant system applied to the quadratic assignment problem. *IEEE Trans. Knowledge Data Eng.*, 11(5):769–778, 1999.

[12] J. Montgomery. Search bias in constructive metaheuristics and implications for ant colony optimisation. Technical Report TR04-04, Faculty of Information Technology, Bond University, Australia, 2004.

[13] M. Randall. Heuristics for ant colony optimisation using the generalised assignment problem. In *Proceedings of CEC 2004*, Portland, OR, USA, 2004.

[14] C. Solnon. Ants can solve constraint satisfaction problems. *IEEE Trans. Evol. Comput.*, 6(4), 2001.

[15] É. D. Taillard and L. M. Gambardella. Adaptive memories for the quadratic assignment problem. Technical Report IDSIA-87-97, IDSIA, 1997.