Analysis of Independent Roulette Selection in Parallel Ant Colony Optimization

Huw Lloyd
Informatics Research Centre,
Manchester Metropolitan University,
Chester Street,
Manchester, United Kingdom M1 5GD.
Huw.Lloyd@mmu.ac.uk

Martyn Amos
Informatics Research Centre,
Manchester Metropolitan University,
Chester Street,
Manchester, United Kingdom M1 5GD.
M.Amos@mmu.ac.uk

ABSTRACT

The increased availability of high-performance parallel architectures such as the Graphics Processing Unit (GPU) has led to significant interest in modified versions of metaheuristics that take advantage of their capabilities. Parallel Ant Colony Optimization (ACO) algorithms are now widely-used, but these often present a challenge in terms of maximizing the potential for parallelism. One common bottleneck for parallelization of ACO occurs during the *tour construction* phase, when edges are probabilistically selected. Independent Roulette (I-Roulette) is an alternative to the standard Roulette Selection method used during this phase, and this achieves significant performance improvements on the GPU. In this paper we provide the first in-depth study of *how* I-Roulette works. We establish that, even though I-Roulette works in a qualitatively different way to Roulette Wheel selection, its use in two popular ACO variants does not affect the *quality* of the solutions obtained. However, I-Roulette *significantly accelerates* convergence to a solution. Our theoretical analysis shows that I-Roulette possesses several interesting and non-obvious features, and is capable of a form of *dynamical adaptation* during the tour construction process.

ACM Reference format:
Huw Lloyd and Martyn Amos. 2017. Analysis of Independent Roulette Selection in Parallel Ant Colony Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference 2017, Berlin, Germany, July 15–19, 2017 (GECCO '17),* 8 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Ant Colony Optimization (ACO) is a meta-heuristic method for combinatorial optimization which is based on the foraging behavior of ants. The scheme was first proposed by Dorigo [8] and has subsequently appeared in several variants [10]. The algorithm is commonly applied to discrete optimization problems such as the Traveling Salesman Problem (TSP), in which the edges of a complete graph are assigned cost values, and the problem is to find the Hamiltonian circuit with minimum total cost.

Applied to the TSP, the ACO algorithm proceeds as follows: at each iteration, a number of simulated ants are placed on random vertices of the graph. Each ant then constructs a Hamiltonian circuit of the graph, selecting the next vertex from the set of unvisited vertices according to a weighted random process in which the weights are determined by heuristic values assigned to the edges. The heuristic value assigned to an edge combines the cost of the edge with the amount of pheromone deposited by ants in previous iterations. After the tour construction is complete, ants deposit pheromone on the edges visited in their tours, the amount of pheromone being inversely proportional to the cost of the tour. At each iteration, the pheromone on each edge is evaporated by a constant fraction. The iterations are repeated until some convergence criterion, or time limit, is met.

The increasing availability of high performance computing platforms such as Graphics Processing Units (GPUs) has led to growing interest in their potential as a platform for *parallel* ACO [1], [4] [11], [20]. GPUs typically offer high computational throughput (albeit with high latency) at relatively low financial cost and with low energy consumption (see, for example, [12]). However, applications require a high degree of *parallelism* in order to exploit the full performance of the hardware, and ACO is a challenging case. Both the main phases of the algorithm – tour construction and pheromone deposition – present challenges for a parallel implementation. Pheromone deposition requires concurrent access to the pheromone data, represented as a two-dimensional square ($N \times N$) matrix, with each entry representing an edge of the complete graph of order N. Although the tour construction phase can be trivially parallelized by assigning one ant to each thread, this *task-parallel* approach is not sufficiently fine-grained to take full advantage of massively parallel hardware such as GPUs.

As an alternative, Cecilia et al. [1] describe an implementation of ACO which instead uses a *data parallel* approach, and which is capable of high parallel efficiency on GPUs. A key component of this algorithm is the Independent Roulette (*I-Roulette*) method, which is used during the tour construction phase to select edges. The development of I-Roulette is motivated by the fact that the standard sequential Roulette Wheel selection method (i.e., where the chance of an edge being selected is directly proportional to its "quality") is extremely difficult to parallelize. I-Roulette is able to achieve significant performance improvements on the GPU, and has provided the foundation for recent work on parallel ACO for image processing [3, 6] and (in an adapted form) data mining [7].

Analyses of I-Roulette [3, 4] focus almost exclusively on its performance in terms of *run-time*, motivating the development of methods that are superior in terms of this metric. Although the I-Roulette method was originally intended to simply replace the

GECCO '17, Berlin, Germany

DOI: 10.475/123_4

sequential version of Roulette Wheel selection, the two methods produce very different *selection probabilities*, which may affect the performance of the algorithm in terms of solution quality and convergence speed. In this paper we address two research questions:

(1) How are the probabilities of selecting edges modified by using I-Roulette?
(2) What effect, if any, does I-Roulette have on the speed of convergence and final solution quality?

We establish that, even though I-Roulette works in a *qualitatively different* way to Roulette Wheel selection, its use in two popular ACO variants does not affect the quality of the solutions obtained, and, moreover, *significantly accelerates* the convergence to a solution. Our theoretical analysis shows that I-Roulette possesses several interesting and non-obvious features, and is capable of a form of *dynamical adaptation* during the tour construction process.

The remainder of the paper is organized as follows. In section 2 we provide a brief motivation for the current work, placing it in the context of existing studies. In section 3 we describe the ant colony algorithms used for the experiments in this paper, and the Roulette Wheel and I-Roulette selection methods. Section 4 presents an analysis of I-Roulette, in which we derive expressions for the probability of selecting an edge using I-Roulette as a function of the distribution of heuristic weights. Section 5 presents the results of experiments conducted to compare the quality of solutions obtained using the two selection methods on a range of standard problem instances using two different ACO variants (and the convergence speed of the algorithms using each method). Finally, in section 6, we summarize our findings and discuss possible directions for future research.

2 MOTIVATION AND RELATED WORK

In order to parallelize the tour construction phase, Cecilia et al. [1] introduced the *I-Roulette* (independent roulette) method. In this scheme, the heuristic weights (W_i, $i \in [1, N]$) for the N edges under consideration are independently multiplied by uniform random deviates $R_i \in [0, 1]$; the edge with the highest product $W_i R_i$ is then chosen as the next edge in the tour. The generation of the deviates and the multiplication by the weights is carried out in parallel, and the maximum is then obtained by a parallel reduction. The algorithm is used to replace the usual *Roulette Wheel Selection*, in which the probablility of selecting an edge is proportional to the edge's weight. Importantly, in the I-Roulette algorithm, that proportionality is lost.

Dawson and Stewart [4] and Dawson [3] also present a GPU implementation of ACO, and introduce *double spin roulette* as a method for selecting edges. In this work, the authors highlight the superiority of their method in terms of runtime. However, unlike I-Roulette, double spin roulette produces a probability of selecting a given edge which is proportional to its weight (as with "traditional" Roulette Wheel selection); Dawson and Stewart [4] argue that this *should* result in better quality solutions. However, the results presented in [1] show no evidence for a degradation in solution quality using I-Roulette, and, if anything, show some evidence for *improvement*. This provides the motivation for the current study, in which we conduct experiments to determine the effects of I-Roulette on the *quality of solutions* found by ant colony algorithms.

Uchida et al. [20] use four different selection algorithms. Three of these are essentially the same as Roulette Wheel selection, with different GPU implementations, while the fourth is *Stochastic Acceptance* [13], which is also equivalent in that it retains the proportionality between the edge weights and the probability of selection. Finally, Fu et al. [11] use the *'all-in roulette'* scheme in their GPU ACO implementation, which is effectively the same as I-Roulette.

The study presented in this paper has two aims: firstly, to gain an understanding of how the selection probabilities are changed by using the I-Roulette process, and, secondly, to empirically determine the effects of I-Roulette selection on ant colony optimization implementation.

3 ANT COLONY OPTIMIZATION

Since it was first described by [8], many variants of ACO have been proposed. In this study, we limit our attention to Max-Min Ant System (MMAS)[18] and Ant Colony System (ACS) [9], two of the best-performing variants.

We now expand on our earlier *informal* description of ACO algorithms for the TSP, in order to establish basic notation and terms. These algorithms proceed iteratively; each iteration comprises two stages: *tour construction* and *pheromone update*. The ant system contains m ants. At the beginning of the tour construction stage, each ant is placed randomly on one of the n vertices of the graph. At each subsequent step in the construction of a tour, ants select the next vertex to visit (and consequently the next edge to traverse) by a random process in which the probabilities of selecting edges are determined by a heuristic weight calculated from the pheromone value associated with the edge and the edge length. The probability of ant k, currently placed on vertex i, of choosing vertex j is given by

$$p_{i,j}^k = \begin{cases} \frac{[\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta}{\sum_{j \in N_i^k} [\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta} & i \in N_i^k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\eta_{i,j} = 1/d_{i,j}$ and $d_{i,j}$ is the length of the edge connecting vertices i and j. $\tau_{i,j}$ is the amount of pheromone associated with edge i, j. N_i^k is the *feasible region* for ant k on vertex i – this is simply the set of vertices not yet visited on the current tour, and is maintained in practice by using the *tabu list*, a list of the vertices already visited. The two parameters α and β are fixed at the beginning of a run, and control the relative importance of edge cost and pheromone in determining the probabilities. In the ACS algorithm, an additional parameter, $q_0 \in [0, 1]$, is introduced. In this algorithm, with probability q_0, the random selection process is replaced by 'greedy' selection – i. e. the edge with the highest weight is chosen without making a random selection.

When all ants have completed their tours, the pheromone values associated with each edge of the graph are updated. Firstly, the pheromone values are *evaporated* according to the rule

$$\tau_{i,j} \leftarrow (1 - \rho)\tau_{i,j} \forall (i, j) \in L \quad (2)$$

where $\rho \in [0, 1]$ is a parameter which controls the rate of evaporation and L is the set of edges in the complete graph. Finally, some

subset of ants deposit pheromone on all edges visited in their tours. The pheromone is updated using

$$\tau_{i,j} \leftarrow \tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,k}^{k}, \forall (i,j) \in L \tag{3}$$

where $\Delta\tau_{i,k}^{k}$ is the amount of pheromone deposited on edge (i,j) by ant k, which is given by

$$\Delta\tau_{i,k}^{k} = \begin{cases} 1/C^k & \text{if edge}(i,j) \in T^k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where T^k is the set of edges in ant k's tour, and C^k is the total cost of tour T^k, which is equal to the sum of the edge lengths, $\sum_{i,j \in T^k} d_{i,j}$. MMAS and ACS differ in how the pheromone is deposited: in MMAS, the iteration-best or best-so-far ant deposits pheronome. In ACS, *only* the best-so-far (global best) ant deposits pheromone. Finally, in the MMAS algorithm, a *clamping* procedure limits the pheromone values between some global minimum and maximum value.

The use of nearest-neighbor lists or candidate sets is an important optimization in the tour construction process [10]. When selecting the next vertex in a tour, only a fixed number of nearest neighbour vertices are considered: if all of these have already been visited (i. e. are in the *tabu* list), a random vertex is chosen. Two of the GPU implementations described in the literature include this optimization ([2], [5]).

4 ANALYSIS OF I-ROULETTE

In this section we analyze I-Roulette in terms of the effect the method has in modifying the probabilities from a given set of weighted edges during the tour construction phase, by deriving exact expressions for the probabilities of selecting edges in terms of the edge weights.

We consider the case where I-Roulette is used to select from a set of N edges with non-zero weights $W_1, W_2, \ldots, W_N$. Without loss of generality, let the weights be ordered such that $W_1 \leq W_2 \leq \ldots \leq W_N$. We first calculate the probability of selecting the highest weighted edge, N. The probability of choosing edge N using Roulette Wheel Selection is

$$P_N = \frac{W_N}{\sum_{i=1}^{N} W_i} \tag{5}$$

We seek the modified probability, which we denote P'_N, of selecting the highest weighted edge using the I-Roulette scheme. In the I-Roulette process, each of the weights $W_i, i \in [1,N]$ is multiplied by an independent uniform random deviate $R_i \in [0,1]$, and the chosen edge is

$$i_{\text{selected}} = \arg \max_{i \in [1,N]} W_i R_i \tag{6}$$

We seek the probability P'_N that $W_N R_N > W_i R_i \forall i \in [1, N-1]$. Let the cumulative probability distribution (the probability that $W_i R_i \leq x$) of $W_i R_i$ be $q_i(x)$, given by

$$q_i(x) = \begin{cases} x/W_i & x \leq W_i \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

The probability distribution function of $W_N R_N$, $p_N(x)$ is given by

$$p_N(x) = \begin{cases} 1/W_N & x \leq W_N \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The probability that $W_N R_N > W_i R_i \forall i \in [1, N-1]$ can then be written as

$$\begin{aligned} P'_N &= \int_0^{W_N} q_1(x) q_2(x) \ldots q_{N-1}(x) p_N(x) dx \\ &= \frac{1}{W_N} \int_0^{W_N} q_i(x) q_2(x) \ldots q_{N-1}(x) dx \end{aligned} \tag{9}$$

Since the W's are ordered, and $q_i(x) = 1$ for $x > W_i$, we can split the integral as follows

$$\begin{aligned} P'_N = \frac{1}{W_N} \Bigg\{ &\int_0^{W_1} q_1(x) \ldots q_{N-1}(x) dx + \\ &\int_{W_1}^{W_2} q_2(x) \ldots q_{N-1}(x) dx + \ldots + \int_{W_{N-2}}^{W_{N-1}} q_{N-1}(x) dx \\ &+ \int_{W_{N-1}}^{W_N} dx \Bigg\} \end{aligned} \tag{10}$$

Substituting for the q_i's and integrating, we find

$$\begin{aligned} P'_N = \frac{1}{W_N} \Bigg\{ &\frac{1}{N} \frac{W_1^N}{W_1 \ldots W_{N-1}} + \frac{1}{N-1} \frac{W_2^{N-1} - W_1^{N-1}}{W_2 \ldots W_{N-1}} + \\ &\ldots + \frac{1}{2} \frac{W_{N-1}^2 - W_{N-2}^2}{W_{N-1}} + W_N - W_{N-1} \Bigg\} \end{aligned} \tag{11}$$

Gathering terms in W_i and rearranging, we obtain

$$P'_N = 1 - \sum_{i=1}^{N-1} \frac{1}{(N-i)(N+1-i)} \frac{W_i^{N-i}}{\prod_{j=i+1}^{N} W_j} \tag{12}$$

For a given unmodified probability P_N, the modified probability will depend on the detailed distribution of the weights $W_1 \ldots W_{N-1}$. We now find the conditions under which P'_N takes minimum and maximum values.

For given P_N, W_N, the sum of the weights from W_1 to W_{N-1} is a constant – different values of P'_N are thus obtained by sharing out this total weight in different ways between $W_1 \ldots W_{N-1}$. Consider the case where a small amount of weight ϵ is exchanged between two adjacent weights W_k and W_{k-1}, $(k > 1)$ such that ϵ is small compared to the weights, and sufficiently small so as not to disturb the ordering of the weights. We now show that this *always* leads to an increase in the modified probability P'_N. We note that ϵ will appear in all terms of the sum in equation 12 with $i \leq k$ (these are the terms that include W_{k-1} and W_k). Let T_k be the term $i = k$, T_{k-1} be the term $i = k-1$, and S be the sum of all the terms with $i < k-1$ (if $k = 2$, $S = 0$). Setting

$$W'_k = W_k - \epsilon \tag{13}$$

and

$$W'_{k-1} = W_{k-1} + \epsilon \tag{14}$$

we can write

$$S' = S \frac{W_k W_{k-1}}{(W_k - \epsilon)(W_{k-1} + \epsilon)} \tag{15}$$

$$T'_k = T_k \left(\frac{W_k - \epsilon}{W_k}\right)^{N-k} \tag{16}$$

and

$$T'_{k-1} = T_{k-1}\left(\frac{W_k}{W_k - \epsilon}\right)\left(\frac{W_k - 1 + \epsilon}{W_{k-1}}\right)^{N+1-k}. \tag{17}$$

We now treat each of these terms in turn.

Rearranging equation 15 we obtain

$$S' = S\left(1 - \frac{\epsilon}{W_k}\right)^{-1}\left(1 + \frac{\epsilon}{W_{k-1}}\right)^{-1} \tag{18}$$

Expanding in terms of ϵ/W_k and ϵ/W_{k-1}, and retaining terms $O(\epsilon)$,

$$S' = S\left[1 + \epsilon\left(\frac{1}{W_k} - \frac{1}{W_{k-1}}\right)\right] + O(\epsilon^2). \tag{19}$$

Since $W_{k-1} \le W_k$, then $S' < S$ for small values of $\epsilon > 0$, independent of the distribution of weights within the terms of S.

Rearranging equation 16, and expanding in terms of ϵ/W_k, we obtain

$$T'_k = T_k\left(1 - \frac{\epsilon}{W_k}\right) + O(\epsilon^2). \tag{20}$$

Hence, $T'_k < T_k$ for small values of $\epsilon > 0$.

For T_{k-1}, we again expand in terms of ϵ/W_k and ϵ/W_{k-1} and retain terms $O(\epsilon)$ to obtain

$$T'_{k-1} = T_{k-1}\left[1 + \epsilon\left(\frac{N+1-k}{W_{k-1}} + \frac{1}{W_k}\right)\right] + O(\epsilon^2). \tag{21}$$

Thus, $T'_{k-1} > T_k$ for small positive values of ϵ.

In order to show that P' always increases under the transformation $W_k \leftarrow W'_k, W_{k-1} \leftarrow W'_{k-1}$, it suffices to show that $T'_k + T'_{k-1} \le T_k + T_{k-1}$, since $S' < S$, and in any case there are no terms in S for $k = 2$. This is equivalent to the condition

$$\frac{T'_{k-1} - T_{k-1}}{T_k - T'_k} \le 1 \tag{22}$$

From equations 20 and 21, and ignoring the terms $O(\epsilon^2)$, we write

$$\frac{T'_{k-1} - T_{k-1}}{T_k - T'_k} = \frac{T_{k-1}}{T_k}\left[\frac{W_k}{W_{k-1}}\frac{N+2-k}{N-k}\right] \tag{23}$$

Substituting for T_{k-1} and T_k, this takes the simple form

$$\frac{T'_{k-1} - T_{k-1}}{T_k - T'_k} = \left(\frac{W_{k-1}}{W_k}\right)^{N-k}. \tag{24}$$

Since $W_{k-1} \le W_k$ by construction, condition 22 is satisfied and P'_N increases under the transformation $W_K \leftarrow W'_k, W_{k-1} \leftarrow W'_{k-1}$, for sufficiently small values of ϵ.

We can now determine the conditions under which P'_N is a minimum and maximum. The maximum value of P'_N, for a given P_N, will occur when all the weights $W_1, W_2, \ldots W_{N-1}$ are equal to $W_N \times (1 - P_N)/(N-1)$; for any other arrangement of the weights, W_{N-1} can be reduced by exchanging ϵ with W_{N-2} leading to an increase in P'_N – hence the maximum of P_N coincides with the minimum of W_{N-1}. We apply similar arguments to find the minimum. Since we can always reduce P'_N by exchanging small amounts ϵ in the direction W_i to W_{i+1}, then the minimum value of P_N will be when W_{N-1}, W_{N-2} etc. are maximized in turn. This is achieved for W_j by setting W_j to the minimum of the remaining weight ($W_N \times (1 - \sum_{i=j+1}^{N} P_i)/P_N$) and W_{j+1}.

When $P_N = 1/M, M < N$, the minimum is constructed by setting $W_N, W_{N-1} \ldots W_{N-M}$ to the same value, and all other weights to zero. This reduces to the case where $N = M$ and the probabilities are equal, hence $P'_{N,\min} = P_N$. Between $P_N = 1/M$ and $P_N = 1/(M+1)$, there is an additional term in the series. We now use this behavior to show that $P'_{N,\min} \ge P_N$. If $1/(M+1) \le P_N \le 1/M$, M an integer less than N, then we construct $P'_{N,\min}$ as follows. For convenience, and without loss of generality, let the weights be normalized such that we can identify weights with probabilities and $W_N = P_N$. Then the weights $W_{N-M+1} \ldots W_N$ are equal to P_N, and $W_{N-M} = 1 - MP_N$. All other weights are zero. We can now use these weights with equation 12, after some manipulation, to write $P'_{N,\min}$, the minimum value of P'_N as

$$P'_{N,\min} = \frac{1}{M} - \frac{1}{M(M+1)}\left(\frac{1 - MP_N}{P_N}\right)^M \tag{25}$$

We wish to show that $P'_{N,\min} \ge P_N$ for $1/(M+1) \le P_N \le 1/M$. This will be the case if $P'_{N,\min} - P_N \ge 0$. Using equation 25, we write the condition for $P'_{N,\min} \ge P_M$ as

$$\left(\frac{1}{M} - P_N\right) - \frac{1}{M(M+1)}\left(\frac{1 - MP_N}{P_N}\right)^M \ge 0 \tag{26}$$

Writing P_N as

$$P_N = \frac{1}{M + \Delta} \tag{27}$$

with $0 \le \Delta \le 1$, condition 26 becomes

$$\frac{\Delta}{M(M+\Delta)} - \frac{\Delta^M}{M(M+1)} \ge 0 \tag{28}$$

For $0 \le \Delta \le 1$ this is always true since the second term is always less than or equal to the first term, and both terms are positive. Since this is true for any $M < N$, we conclude that $P'_{N,\min}$ is always $\ge P_N$, and hence P'_N is always $\ge P_N$.

To summarize the results on P'_N,

(1) The modified probability is given by

$$P'_N = 1 - \sum_{i=1}^{N-1} \frac{1}{(N-i)(N+1-i)} \frac{W_i^{N-i}}{\prod_{j=i+1}^{N} W_j} \tag{29}$$

(2) P'_N is a maximum for a given P_N when all the weights $W_1 \ldots W_{N-1}$ are equal.

(3) P'_N is a minimum for a given P_N when the weights W_{N-1}, $W_{N-2} \ldots$ are maximized in turn (i.e. by setting each weight W_i to the minimum of the remaining total weight and W_{i+1}).

(4) $P'_N \ge P_N$.

The modified probability of edge $N-1$, P'_{N-1} can then be found using the reduced set of edges $1 \ldots N-1$, and scaling by $1 - P'_N$. This process can then be applied recursively to obtain the complete set of modified probabilities. The general expression for P'_{N-k}, $k \in [1, N-1]$ is

$$P'_{N-k} = \left(1 - \sum_{i=N-k}^{N} P'_i\right) \times \left[1 - \sum_{i=1}^{N-k-1} \frac{1}{(N-k-i)(N+1-k-i)} \frac{W_i^{N-k-i}}{\prod_{j=i+1}^{N-k} W_j}\right] \tag{30}$$

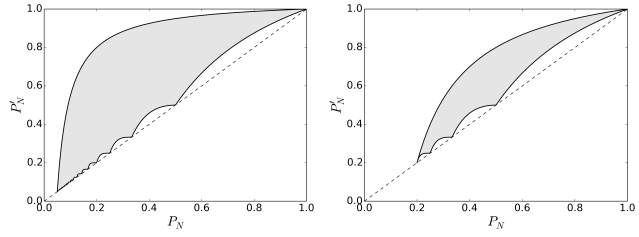


Figure 1: Plots of the minimum and maximum values of P'_N as functions of P_N for (a) $N = 20$ and (b) $N = 5$

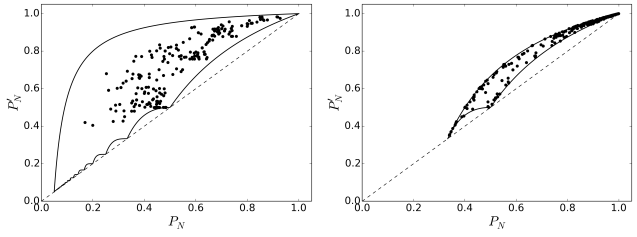


Figure 2: Values of P'_N vs. P_N extracted runs of the MMAS algorithm with I-Roulette on the test problem d198 when selecting between (a) 20 edges and (b) 3 edges. Data is extracted at the tenth iteration of the algorithm.

Figure 1 shows plots of the allowed values of P'_N (the shaded area) for $N = 3$ and $N = 20$. Note that only the region $P_N \geq 1/N$ is plotted – since W_N is the largest weight, $P_N < 1/N$ is impossible. Clearly, when N is relatively large, it is possible for P'_N to approach unity even when P_N is small. The effect of I-Roulette appears to be that when selecting from a large number of edges (as is the case early in the construction of a tour), the highest weighted edge is chosen with disproportionately high probability with respect to the weights.

It is instructive to look at real instances of I-Roulette extracted from runs of an ACO code. Figure 2 shows data extracted from runs of the MMAS algorithm on the d198 TSP test problem. The plots show probabilites derived from weights extracted from cases when ants were choosing between 20 and 3 non-zero weighted edges respectively. The maximum and minimum values of P'_N as a function of P_N, and the line $P'_N = P_N$ are shown for guidance.

For the case $N = 3$, I-Roulette selects using probabilities that do not deviate greatly from Roulette Wheel. For $N = 20$, however, we see that in many cases the algorithm *considerably amplifies* the probability of the most likely edge. These are cases where the selection method is presented with a set of edges in which one edge carries a significantly larger weight than the others. In these cases, the algorithm tends towards *greedy* selection (in which the highest weighted edge is always selected). This situation is more likely to occur at relatively early stages of tour construction , when most of the edges in the nearest-neighbor list are available.

Clearly, I-Roulette is behaving in a *qualitatively different way* to roulette selection, often amplifying the probability of an edge by a large factor in cases where there are a large number of edges to be chosen from and when one edge carries the majority of the weighting. When there are relatively few edges to choose from, as will occur in the later stages of tour construction, the behavior more closely approximates the proportional probabilities obtained from Roulette Wheel selection.

The effect of I-Roulette in amplifying certain probabilities by large factors may seem counter-intuitive, but the underlying mechanism is easily demonstrated by example. Consider the case which is schematically represented in Figure 3, in which there are 20 weights, the largest weight is 1, and the other weights are all approximately equal but < 0.8. Using Roulette Wheel selection, all the

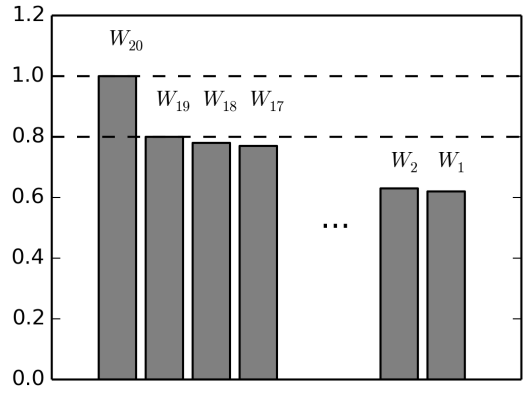


Figure 3: Schematic representation of an illustrative problem with 20 weights. See text for details.

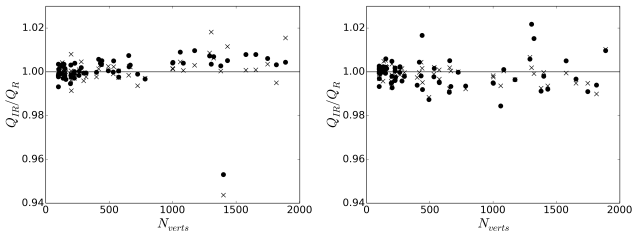
choices would carry a probability ~ 0.05, with slightly higher probability for the highest weighted choice. However, using I-Roulette, we see that there is a probability of 0.2 that W_{20} is multiplied by a random number (R_{20}) which is greater than 0.8. In this case, it is impossible for any of the other choices to 'win' the process, so the probability of selecting the highest weighted choice is at least 0.2. There will be a small additional contribution from the possibility of $W_{20}R_{20}$ winning the process when $R_{20} < 0.8$, but this will be again ~ 0.05. Thus, the probability is amplified by a factor of at least four. The I-Roulette probability is dominated in this case by the relative amount by which W_{20} is greater than its nearest rival, W_{19}. This effect is greater when N is large, since the roulette wheel probability varies as $1/N$, whereas the I-Roulette probability P'_N is dominated by the relative difference between W_N and W_{N-1}, which is independent of N.

Table 1: TSPLIB Instances used for the experimental runs.

Group A	kroA100, kroB100, kroC100, kroD100, kroE100, rd100, eil101, lin105, pr107, pr124, bier127, ch130, pr136, gr137, pr144, ch150, kroA150, kroB150, pr152, u159, rat195, d198, kroA200, kroB200, gr202, ts225, tsp225, pr226, gr229, gil262, pr264, a280, pr299, lin318, rd400, fl417, gr431, pr439, pcb442, d493
Group B	att532, ali535, u574, rat575, p654, d657, gr666, u724, rat783
Group C	dsj1000, pr1002, u1060, vm1084, pcb1173, d1291, rl1304, rl1323, nrw1379, fl1400, u1432, fl1577, d1655, vm1748, u1817, rl1889

Table 2: ACOTSP parameters for the experimental runs

m	MMAS: 50 (A) 100 (B,C), ACS: 10
Nearest-neighbors	20
α	1
β	2
ρ	MMAS: 0.2, ACS: 0.1
q_0	MMAS: 0, ACS: 0.9
Tour evaluations	50000 (A), 200000 (B), 400000 (C)



a b

Figure 4: Plots of solution quality ratio (I-Roulette/Roulette) vs. number of vertices for (a) MMAS and (b) ACS. Circles represent mean solutions, crosses best solutions.

5 EXPERIMENTAL RESULTS

In this section we describe a series of experiments conducted to investigate the effect of I-Roulette on solution quality and convergence speed with two ACO variants, using a set of standard test problems.

5.1 Experimental Setup

Runs were carried out using the standard ACOTSP code [17], with a modification to allow the roulette selection procedure to be replaced by I-Roulette. Other than this change, the code is unmodified.

5.1.1 Problem Instance Set. Problem instances were selected from the TSPLIB [16] library of TSP instances. All instances with 100–2000 vertices and edge weight types supported by ACOTSP were used in the experiments: this gives a total of 65 instances, which are listed in Table 1. For the selection of ACO parameters, these were divided into three groups: those with 100–499, 500–999 and 1000–2000 vertices respectively. For each instance, we ran 50 trials each of ACS and MMAS, both with and without I-Roulette.

5.1.2 Algorithms and Parameters. Runs were carried out using two ACO variants: Ant Colony System (ACS) and Max-Min Ant System (MMAS). These two variants are among the best performing ACO algorithms for the symmetric TSP. The parameters were chosen based on recommendations in [19], and are listed in Table 2. Note that [19] recommends a small number of ants ($m = 10$) for ACS, and, although their default setting for MMAS is $m = n$ ants (where n is the number of vertices), they obtain better results with fewer ants. Here, we use $m = 50$ for the smaller problems, and $m = 100$ in the larger problems. The runs are limited to a fixed number of tour evaluations: this parameter is set to a different value for each of the three size groups in order to ensure well-converged solutions in all cases. The tour evaluation limit is the same for both MMAS and ACS; since the ACS runs use fewer ants, they run for more iterations, but the total runtime is comparable.

5.2 Solution Quality

We define the *solution quality*, Q, as the ratio of the length of the *shortest tour* found in a run to the *known optimum* for the problem instance. The results are summarized in Figure 4, in which we plot the mean values of the solution quality obtained using I-Roulette (Q_{IR}) versus those obtained using Roulette Wheel selection (Q_R) for ACS and MMAS. Both algorithms obtain very similar solution quality, and the data is clustered closely around the line $Q_{IR} = Q_R$. With MMAS, the solution quality appears to be slightly degraded by using I-Roulette, whereas there is no noticeable effect in the ACS runs. In order to investigate any dependency on the number of vertices, we plot the ratio Q_{IR}/Q_R as a function of number of vertices in figure 5. Table 3 gives the mean and standard deviation of the ratio Q_{IR}/Q_R for each group in instances for MMAS and ACS. Values are given for ratios calculated using both the mean and best values of Q.

Figure 5 (a) shows that MMAS has marginally worse performance with I-Roulette for instances with > 1000 vertices, apart from one outlier (instance fl1400), in which I-Roulette appears to perform much better. If this point is discounted, the mean ratio Q_{IR}/Q_R for MMAS in Group C is 1.0058 ± 0.0022 for mean values, and 1.0048 ± 0.0060 for best values, which is less than a 0.5% degradation. We conclude, therefore, that I-Roulette does not lead to any significant degradation of the quality of solutions obtained using ACS and MMAS. This conclusion is supported by the data in Table 4, which shows the results of applying the Wilcoxon signed rank test to the paired values of mean and best solution quality. The null hypothesis that the median difference in quality is zero can be rejected for the mean solution quality ($\langle Q \rangle$) using MMAS, and in this case the size of the effect (from the median difference) is small (0.015).

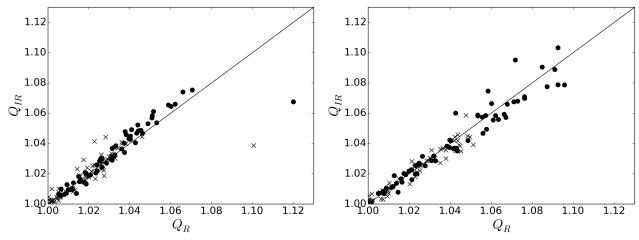


Figure 5: Plots of solution quality, IRoulette vs. Roulette for (a) MMAS and (b) ACS. Circles represent mean solutions, crosses best solutions.

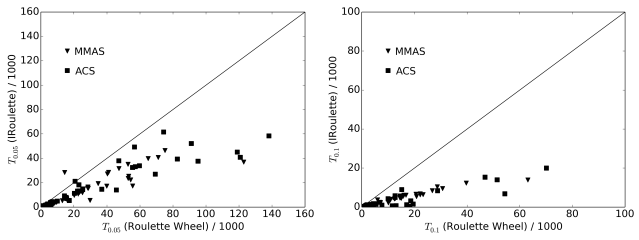


Figure 6: Scatter plots of median values for (a) $T_{0.05}$ and (b) $T_{0.1}$ for IRoulette (y-axis) vs. Roulette (x-axis).

Table 3: Mean and standard deviation of Q_{IR}/Q_R from the experimental runs.

		Mean Q		Best Q	
		$\langle Q_{IR}/Q_R \rangle$	$\sigma(Q_{IR}/Q_R)$	$\langle Q_{IR}/Q_R \rangle$	$\sigma(Q_{IR}/Q_R)$
MMAS	A	1.0001	0.0028	1.0003	0.0028
	B	1.0013	0.0033	0.9993	0.0028
	C	1.0025	0.0129	1.0010	0.0159
ACS	A	0.9996	0.0047	0.9997	0.0031
	B	0.9997	0.0043	0.9984	0.0036
	C	0.9995	0.0095	0.9980	0.0051

Table 4: Results of statistical tests. Δ is the median difference (I-Roulette minus Roulette) for the quantity x, and p is the two-sided p-value from the Wilcoxon signed rank test. Statistically significant p values are in bold face.

	x	Δ	p
MMAS	$\langle Q \rangle$	0.015	**0.004**
	Q_{best}	0.0003	0.094
	$T_{0.05}$	−3700	**0.000**
	$T_{0.1}$	−2400	**0.000**
ACS	$\langle Q \rangle$	−0.005	0.094
	Q_{best}	0.0	0.113
	$T_{0.05}$	−1800	**0.000**
	$T_{0.1}$	−480	**0.000**

5.3 Convergence Speed

We define the quantity T_f as the number of tour evaluations in a run before the solution quality is within a factor $1 + f$ of the best solution found in the run. For example, $T_{0.05}$ is the number of tours constructed in the run when the solution is within 5% of the best value found at the end of the run. For each problem instance and algorithm, we compute the median over 50 trials of $T_{0.05}$ and $T_{0.1}$. These quantities are plotted for the 65 instances in Figure 6. In all cases, bar one outlier in the MMAS runs, I-Roulette shows *considerably quicker convergence* to the region of the best solution. The values for all runs are considerably lower than the maximum number of tour evaluations for the experiments, which confirms that the solutions obtained are all well converged. As may be expected from the plots, the statistical tests (Table 4) give very strong support to the conclusion that I-Roulette accelerates the convergence to a solution.

6 CONCLUSION AND FURTHER WORK

This paper presents a detailed analysis of I-Roulette, a replacement for Roulette Wheel selection in parallel Ant Colony Optimization. The theoretical analysis shows that the probabilities are modified in a way that tends towards greedy selection in cases where there are a large number of non-zero weights, but reverts to proportional probabilities when faced with fewer choices. The algorithm will therefore tend to greedy selection early in the construction of a tour, but will become more conservative in the later stages. Our experimental results with the MMAS and ACS variants of ACO show that there is no significant effect on solution quality, and that convergence to a solution is greatly accelerated by using I-Roulette. As well as allowing efficient parallel implementations of ACO on hardware such as GPUs, I-Roulette may also confer considerable benefits, by reducing the number of trials required to reach a given quality of solution, accelerating the computation even further.

The results pose a number of questions, which we hope to address in future work. Firstly, although the behavior of I-Roulette in modifying the probabilities has been determined analytically, and the effects on the solutions obtained have been observed empirically, there is no clear mechanism which links the two: *why* the modified probabilities lead to faster convergence remains an open question. An understanding of this mechanism may lead to new variants of the ACO algorithm which use the behavior to improve performance.

Secondly, we have conducted experiments which have shown that there is, on average, little effect on solution quality and, in general, an improvement in convergence speed, but there is considerable variation among the problem instances studied here. It may be possible to predict, for a given problem instance, whether I-Roulette may be preferred over Roulette wheel selection or *vice versa*. Recent work on analyzing the performance of TSP algorithms in terms of problem instance features ([14], [15]) has determined a range of metrics which can predict the performance of some algorithms on a given TSP instance. This has enabled the development of techniques for generating instances which are 'hard' and 'easy'.

A similar analysis could be used to investigate the performance of I-Roulette in ACO.

Finally, this study used fixed values of the algorithm parameters. It is possible that a parameter tuning approach could lead to I-Roulette being even more effective in ACO, and the optimum parameters when using I-Roulette may differ from those for Roulette Wheel selection. This is an area for future study.

REFERENCES

[1] José M. Cecilia, José M. García, Andy Nisbet, Martyn Amos, and Manuel Ujaldón. 2013. Enhancing Data Parallelism for Ant Colony Optimization on GPUs. *J. Parallel Distrib. Comput.* 73, 1 (2013), 42–51.

[2] J. M. Cecilia, J. M. Garcia, M. Ujaldon, A. Nisbet, and M. Amos. 2011. Parallelization strategies for ant colony optimisation on GPUs. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*. 339–346. DOI: http://dx.doi.org/10.1109/IPDPS.2011.170

[3] Laurence Dawson. 2015. *Generic Techniques in General Purpose GPU Programming with Applications to Ant Colony and Image Processing Algorithms*. Ph.D. Dissertation. Durham University, UK.

[4] Laurence Dawson and Iain Stewart. 2013. Improving Ant Colony Optimization performance on the GPU using CUDA. In *2013 IEEE Conference on Evolutionary Computation*, Luis Gerardo de la Fraga (Ed.), Vol. 1. Cancun, Mexico, 1901–1908.

[5] Laurence Dawson and Iain A. Stewart. 2013. Candidate Set Parallelization Strategies for Ant Colony Optimization on the GPU. In *Algorithms and Architectures for Parallel Processing: 13th International Conference, ICA3PP 2013, Vietri sul Mare, Italy, December 18-20, 2013, Proceedings, Part I*, Joanna Ko lodziej, Beniamino Di Martino, Domenico Talia, and Kaiqi Xiong (Eds.). Springer International Publishing, 216–225. DOI: http://dx.doi.org/10.1007/978-3-319-03859-9_18

[6] Laurence Dawson and Iain A Stewart. 2014. Accelerating ant colony optimization-based edge detection on the GPU using CUDA. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1736–1743.

[7] Youcef Djenouri, Ahcene Bendjoudi, Malika Mehdi, Nadia Nouali-Taboudjemat, and Zineb Habbas. 2015. GPU-based bees swarm optimization for association rules mining. *The Journal of Supercomputing* 71, 4 (2015), 1318–1344.

[8] Marco Dorigo. 1992. *Optimization, Learning and Natural Algorithms*. Ph.D. Dissertation. Politecnico di Milano, Italy.

[9] M. Dorigo and L. M. Gambardella. 1997. Ant colony system: a cooperative learning approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation* 1, 1 (Apr 1997), 53–66. DOI: http://dx.doi.org/10.1109/4235.585892

[10] Marco Dorigo and Thomas Stützle. 2004. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA.

[11] Jie Fu, Lin Lei, and Guohua Zhou. 2010. A parallel Ant Colony Optimization algorithm with GPU-acceleration based on All-In-Roulette selection. In *Advanced Computational Intelligence (IWACI), 2010 Third International Workshop on*. 260–264.

[12] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Yao Zhang, and V. Volkov. 2008. Parallel Computing Experiences with CUDA. *Micro, IEEE* (2008).

[13] Adam Lipowski and Dorota Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* 391, 6 (2012), 2193 – 2196.

[14] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, Jakob Bossek, and Frank Neumann. 2013. A novel feature-based approach to characterize algorithm performance for the Traveling Salesperson Problem. *Annals of Mathematics and Artificial Intelligence* 69, 2 (2013), 151–182. DOI: http://dx.doi.org/10.1007/s10472-013-9341-2

[15] Samadhi Nallaperuma, Markus Wagner, and Frank Neumann. 2015. Analyzing the Effects of Instance Features and Algorithm Parameters for Max-Min Ant System and the Traveling Salesperson Problem. *Frontiers in Robotics and AI* 2 (2015), 18. DOI: http://dx.doi.org/10.3389/frobt.2015.00018

[16] Gerhard Reinelt. 1991. TSPLIB – A Traveling Salesman Problem Library. *ORSA Journal on Computing* 3, 4 (1991), 376–384.

[17] Thomas Stützle. 2004. ACOTSP, Version 1.03. http://www.aco-metaheuristic.org/aco-code. (2004). Accessed: 2017-01-31.

[18] T. Stutzle and H. Hoos. 1997. MAX-MIN Ant System and local search for the Traveling Salesman Problem. In *Evolutionary Computation, 1997., IEEE International Conference on*. 309–314. DOI: http://dx.doi.org/10.1109/ICEC.1997.592327

[19] Thomas Stützle, Manuel López-Ibáñez, Paola Pellegrini, Michael Maur, Marco Montes de Oca, Mauro Birattari, and Marco Dorigo. 2012. *Parameter Adaptation in Ant Colony Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 191–215. DOI: http://dx.doi.org/10.1007/978-3-642-21434-9_8

[20] A. Uchida, Y. Ito, and K. Nakano. 2012. An Efficient GPU Implementation of Ant Colony Optimization for the Traveling Salesman Problem. In *Networking and Computing (ICNC), 2012 Third International Conference on*. 94–102.