Brock University

Department of Computer Science

Ant Colony Optimization for Job Shop Scheduling Problem

M. Ventresca and B.M. Ombuki
Technical Report # CS-04-04
February 2004

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Ant Colony Optimization for Job Shop Scheduling Problem

Mario Ventresca
Dept. of Computing and Information Science
University of Guelph
Guelph, Ontario, Canada
mventres@uoguelph.ca

Beatrice M. Ombuki
Department of Computer Science
Brock University
St.Catharines, Ontario, Canada
bombuki@cosc.brocku.ca

Abstract- This paper presents an application of the Ant colony optimization metaheuristic to the job shop scheduling problem. A pheromone alteration strategy which improves the basic ant system by utilizing the behaviour of artificial ants to perform local search is introduced. Experiments using well-known benchmark JSSP problems show that this approach improves on the performance obtained by the basic ant system and is competitive with another recently proposed extension of the ant system, the MAX-MIN algorithm.

1 Introduction

This paper examines an application of the recently proposed adaptive metaheuristic Ant Colony Optimization (ACO) [1] for the job shop scheduling problem (JSSP). In the static JSSP, a finite number of jobs are to be processed by a finite number of machines. Each job consists of a predetermined sequence of task operations, each which needs to be processed without interruption for a given period of time on a given machine. Tasks of the same job cannot be processed concurrently and each job must visit each machine exactly once. A feasible schedule is an assignment of operations to time slots on a machine without violation of the job shop constraints. A makespan is defined as the maximum completion time of the jobs. The objective of the JSSP is to find a schedule that minimizes the makespan. A good schedule is one that minimizes the total amount of time machines are idle. According to complexity theory [2], the JSSP is characterized as NP-hard combinatorial optimization problem. Obtaining exact solutions for such problems is computationally intractable [2].

Inspired by the study of Argentine ants conducted by Goss et al. [3], Marco Dorigo developed the ACO which uses principles of cooperative behaviour found in real ants colonies to solve hard combinatorial optimization problems. Indirect communication and information exchange between ants is achieved by mimicking the communicative behaviour (so-called *foraging and recruiting behaviour)* of real ant colonies (see [1,4] for an an introduction and overview). In the ACO, each ant constructively builds a solution by several stepwise probablistics decisions until a solution is reached. The ACO meta-heuristic has been applied to various hard combinatorial optimization problems. For example, in the scheduling field, ACO has effectively been applied to the Flow-shop problem [5], Resource Constraint project Scheduling problem [6], the Single Machine Total Tardiness problem [7]. ACO has also been shown to solve other permutation scheduling problems such as the Travelling Salesman problem [8,9] and Vehicle routing problems [10]. However, the application of the ACO to Shop scheduling problems such as the JSSP and open shop scheduling problem has proved to be quite difficult [11] and very few papers report work on ACO implementation for the JSSP. Colorni et. al were the first to apply an ant system (AS) to the JSSP in [12]. Although the contribution of the results obtained using the ACO for the JSSP are significant to the scheduling community, "the algorithm was far from reaching state-of-the-art performance" [11]. Sjoerd van der Zwaan et. Al [13] next developed an ACO for the JSSP where he used genetic algorithm for ACO parameter tuning. More recently C. Blum et. al. have investigated the application of ACO to shop scheduling problems including the JSSP [11,14].

In this paper, we develop an ant colony optimization approach to the job shop scheduling problem that utilizes a local search technique (we call *Foot Stepping*) which aids in exploration of the solution space. Furthermore, unlike most local search techniques which are problem specific, the foot stepping based strategy is more flexible as it does not alter the ant system algorithm, but rather the information gathered by the ants to improve on the current solution.

The remainder of this paper is as follows: Section 2 gives a formal description of the job shop scheduling problem and Section 3 introduces the basic Ant System (AS) algorithm. Our extension of the AS is given in Section 4 followed by

experimental results using benchmark problems and comparison with the Max-Min technique in Section 5. Concluding remarks and future work is given in Section 6.

2 Job Shop Scheduling Problem

This paper adopts a standard model of the n-job, m-machine JSSP, denoted by n/m/G/Cmax, where the parameter G represents the technological production rules associated with the jobs (their processing order on the machines). The parameter Cmax indicates the performance measure which should be minimised (maximum time taken to complete all jobs). The following is an example of a technological matrix T (thus G=T in this case), which indicates the ordering of tasks of a job to machines.

$$T = \begin{bmatrix} M1, M2, M3 \\ M2, M3, M1 \end{bmatrix}$$

Where each row of the matrix represents a given job and the elements represent the scheduling sequence of tasks (i.e., M1 is machine 1, meaning task 1 for job 1 is processed by machine 1). The processing time of each task operation is given in the matrix represented by P as follows:

$$P = \begin{bmatrix} t_{11}, t_{12}, ..., t_{1M} \\ t_{21}, t_{22}, ..., t_{2M} \\ . \\ . \\ t_{n1}, t_{n2}, ..., t_{nM} \end{bmatrix}$$

Matrices T and P define the job shop scheduling problem. (Note that not all entities of T are shown here for simplicity)

In order to apply the ACO algorithm, the optimization problem in question must first be translated into a graph *G=(C, L)*. (Figure 2 gives a graphical representation of G for the JSSP problem given in Figure 1) *C* and *L* represent the vertices and edges, respectively, and are defined with the following characteristics and notations (adopted throughout the paper):

- A finite set of basic *components* of the problem $C=\{c_1, c_2, \ldots, c_N\}$.
- A finite set *E* of *connections* between a subset of elements of *C*, such that $|E| \leq N^2$. (N represents the number of components, i.e., N=|C|).
- For every $e_{i,j} \in E$, a *connection cost* $d_{i,j}$ representing the cost of traveling from component *i to j*, $\eta_{i,j} = 1/d_{i,j}$ and is typically termed the *heuristic distance*.
- A finite number *constraints* Ω defined over the elements of *C* and *E*.
- Given a set *S* of all possible sequences $<c_i,c_j,\ldots c_k,\ldots>$ over the elements of *C*, it is required that a set $\hat{S} \subseteq S$ exist representing the *feasible sequences* with respect to Ω.
- A *neighborhood* structure defined such that sequences s_1 and s_2 can be considered in the same neighborhood if (i) s_1 and $s_2 \in S$, (ii) s_2 can be reached from s_1 in one logical step.
- A *solution* $\Psi \in \hat{S}$.
- A *cost function* $\Phi_\Psi(E,t)$ associated with each solution Ψ, representing the total cost of the solution, where *t* represents time.

(2,10) **1**	(1,2) **2**	(3,5) **3**
(1,12) **4**	(3,6) **5**	(2,2) **6**

Figure 1 An example of a simple JSSP instance with two jobs (row 1 and 2 represent jobs 1 & 2 respectively) to be processed on three machines. The format of the data is (machine, duration), the bold numbers refer to Figure 2.

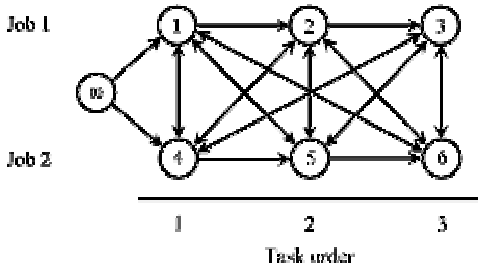

Figure 2 A graphical representation for a 2-job, 3-machine problem instance shown in Figure 1. The tasks labeled 1-3 are associated with job 1, whereas the remaindered are associated with job 2.

3 Ant System (AS)

The basic idea of the AS is to keep a population or colony of *m* artificial ants that iteratively builds a solution by continually applying a probabilistics decision policy *n* times until a solution is found. Ants that found a good solution mark their path

through the decision space by putting some amount of pheromone on the edges of the path. Ants of the next iteration are attracted to the pheromone resulting in a higher probability to follow the already traversed good paths. In addition to the pheromone values, the ants will usually be guided by some problem specific heuristic for evaluating possible decisions regarding which direction to take along the way. AS ants have a memory (tabu list) that stores visited components of their current path. Note that this is not the same tabu list as found in the tabu search strategy introduced by Glover [15].

As indicated above, in order to apply the AS algorithm, a graphical representation *G,* of the optimization problem in question must first be constructed. The meta-heuristic begins by initializing the amount of pheromone along each edge of *G* to some positive real value *c*. Each ant is then designated a starting position, which is added to its tabu list. The initial ant positions are generally randomly chosen. Upon completion of the initialization phase, each ant will independently construct a solution, employing (1) at each decision point until a complete solution has been found. After every ant's tabu list is full (i.e., It has found a valid solution), its path length will be determined by $\Phi_\Psi(E,t)$ (given in Section 2), and the best solution found would be recorded. The pheromone amount along each edge (i,j) is recomputed according to (2). Finally, all tabu lists are emptied, and if the stopping criteria have not been met, the algorithm will continue.

The decision of each ant is based not only upon the amount of pheromone $\tau_{i,j}$, present along edge (i,j), but may also consider a heuristic distance $\eta_{i,j}$ along the edge. The transition probability to branch from node *i* to node *j* for each kth ant at time *t* is defined as:

$$P^k_{i,j} = \begin{cases} \dfrac{[\tau_{i,j}(t)]^\alpha [\eta_{i,j}]^\beta}{\sum_{k \in allowed_k} [\tau_{i,k}(t)]^\alpha [\eta_{i,k}]^\beta} & \text{if } (i,j) \in tabu_k \\ 0 & \end{cases} \quad (1)$$

where $tabu_k$ is the list of traversed edges and $allowed_k = \{C\text{-}tabu_k\}$. The parameters α, β are user defined parameters that determine the degree to which the pheromone is used versus the heuristic distance in deciding where to move. Setting β = 0 will result in only the pheromone information being used whereas if α = 0, only the heuristic information will be used. In either case, the ACO algorithm would degenerate to a greedy algorithm where either only the pheromone or heuristic information is considered. Furthermore, the definition of a given problem may impose some constraints on the sequence of nodes resulting in valid solutions. In this case, given a partial solution under construction, if a node is not valid, its probability of selection would be zero. This implies that nodes that are arrived at via a high pheromone edge, and closer to the actual node in terms of heuristic distance will have a higher probability of being selected.

Ants will lay pheromone according to either a step-by-step or delayed pheromone updating technique. The conventional practice is to update the pheromone amount according to:

$$\tau_{i,j}(t+n) = \rho \cdot \tau_{i,j} + \Delta\tau_{i,j} \quad (2)$$

where ρ is a real valued coefficient such that (1- ρ) represents the *evaporation coefficient* of the pheromone along edge (i,j) between time *t* and *t*+*n*. In order to avoid an over accumulation of pheromone along an edge, it is required that $\rho \in (0,1)$.

The total amount of pheromone laid by the *m* ants, $\Delta\tau_{i,j}$ is calculated by:

$$\Delta\tau_{i,j} = \sum_{k=1}^{m} \Delta\tau^k_{i,j} \quad (3)$$

where $\Delta\tau^k_{i,j}$, (i.e., the amount of pheromone deposited by each ant) is calculated via:

$$\Delta\tau^k_{i,j} = \begin{cases} \dfrac{Q}{L_k} & \text{if the k}^{th}\text{ ant travels along edge (i,j)} \\ 0 & \end{cases} \quad (4)$$

between time *t* and *t*+*n* , where *Q* is a positive real valued constant and L_k is the length of the path of the kth ant, as calculated by $\Phi^k_\Psi(E,t)$. Hence, Q/L gives the quantity of pheromone per unit of distance.

Apart from the construction of solutions and depositing of pheromone, the AS incorporates two other methods*, pheromone evaporation* and optional *daemon activities*. Pheromone evaporation causes the amount of pheromone on each edge of *G* to decrease over time. The important property of evaporation is that it prevents premature convergence to a sub-optimal solution. In this manner, the AS has the capability of "forgetting"

bad solutions, which favors the exploration of the search space. Daemon activities can be any action such as a local search technique over the solutions of ants. Off-line pheromone updating occurs if the daemon is given the responsibility of laying pheromone on the edges of G.

4 Ant Colony Optimization for JSSP

The details of the proposed ACO algorithm for the JSSP are provided in this section. The general idea was to adapt a basic Ant System approach and investigate further exploration of the solution space. The exploration technique, termed Foot Stepping (FS) is provided here. A key feature of the FS strategy is that it does not alter the ant system algorithm, but rather alters the information gathered by the ants. That is, the ants are used as opposed to another search technique to further improve solutions (i.e., perform local search) already found by the ant colony. This means that unlike most hybrid techniques that are geared towards specific problems, the FS technique is more flexible and is not problem dependent. The perfomance of the foot stepping technique is compared with another well-known extension of the AS, based on Max-Min Ant System (MMAS) [16]. Both techniques were implemented along with a basic AS approach and applied to the JSSP.

4.1 Foot Stepping

Foot Stepping is inspired by the ability of real-life ants to modify their paths if an obstacle, say a foot, should suddenly occur in their way. In our artificial ants, if this occurs, we assume that nearly all information regarding the pheromone along a stepped-on edge will disappear. The ants are then forced to find a way around the obstacle, in the same manner as before. The goal of FS is to cause fluctuations, or footsteps, along good solutions already discovered by the AS. The FS technique is applied only after or near stagnation, and thus the colony has already discovered a good solution. A fluctuation is an alteration of the environment in which the ants exist. Specifically, it causes the amount of pheromone along certain edges of a graph G to be altered.

To achieve this *pheromone amplification* and *reduction functions* were used. These functions are embedded into the FS algorithm described below. Since ants would have already constructed a reasonable solution, any fluctuation will cause a local search around the current solution. Thus, the radius of local search from the already good solution is determined by the number of footsteps performed. Note that the global best solution found will be used to lay pheromone after each set of footsteps has occurred. In this manner, ants are forced to search for solutions to improve the global best solution.

The Foot Stepping algorithm is run as a daemon activity, and the placing of an obstacle can occur at any randomly selected component, or vertex $v \in G$. The footstep decreases the pheromone along the densest edge of v, and increases all others. The amount of increase or decrease is determined by the amplification and reduction functions (for the JSSP, the functions are given in Section 5). Two main variations of the FS activity were developed, and are henceforth referred to as Non-propagation based Foot Stepping (NPFS) and Propagation based Foot Stepping (PFS).

4.2 Non-Propagation based Foot Stepping

The NPFS algorithm begins by randomly selecting a set $V \subseteq C$, *where* $|V| = numSteps$. For each $v_i \in V$, the edge of maximum pheromone value leaving it, $v^{max}_{i,}$ is reduced according to some *pheromone reduction function* (PRF). The pheromone cannot be diminished to a value less than or equal to zero or ants will be unable to construct paths where v^{max}_{i} is a part of the solution. It should be possible to reconstruct the pre-stepped on path over time. The pheromone quantity on other edges **is** increased according to a *pheromone amplification function* (PAF). Figure 3 outlines the basic NPFS algorithm.

```
procedure NPFS
    if ( canStep() )
        mem = getPheromoneMatrix();
        for i := 1 to numSteps
            v_i = selectRandomVertex();
            max = findMaxPheromone(v_i);
            for j := 1 to v_i.numEdges()
                if ( mem[v_i][j] ≠ max )  // not v^max_i
                        mem[v_i][j] =
                PAF(mem[v_i][j]);
                else
                        mem[v_i][j] =
                PRF(mem[v_i][j]);
            end for;
        end for;
end procedure;
```

Figure 3 Non-Propagation based Foot Stepping algorithm

The canStep() method verifies that conditions are suitable to perform another footstep. This can be a lower bound on the number of iterations completed or when stagnation conditions are satisfied. The algorithm will proceed to randomly select a vertex v_i of *G*, as well as determine v^{max}_i and either increase or decrease the amount of pheromone on its outgoing edges according to the PAF or PRF, respectively. This process will continue until the requested number of footsteps have taken place. The complexity of NPFS is O(numSteps·v_i.numEdges()).

It should be noted that multiple sets of footsteps occuring over small intervals of time can cause too great of a disturbance in the information stored in the pheromone trails and thus lead to a random search. In order to avoid this the number of footsteps are restricted to a small number. Another method of avoiding too many footsteps is to step according to some small probability. Due to the positive feedback nature of the ant algorithm,it is very important to allow the colony time to process the new pheromone landscape.

4.3 Propagation based Foot Stepping

Propagation of pheromone refers to the process of depositing pheromone from the foot-stepped node(s) to all other edges occurring after it until the end of *G* has been reached. The difference between NPFS and PFS is that NPFS lacks this propagation property.

The PFS technique is very similar to NPFS except that it not only footsteps on a random number of edges, but also propagates pheromone to each consecutive edge. In this paper, the amount propagated is based on a harmonic function (a function of the form *f(x)=1/x*), although the *propagation function* may be different.

A real world analogy to PFS would be stepping on a path with enough force such that a gust of wind is created that causes nearby pheromone to blow off its trail. Therefore, not only has the stepped-on path pheromone been disrupted, but so too has the pheromone of nearby edges. This causes the ants to explore more of the surrounding area in search of a strong pheromone trail. As ants move away from the foot-stepped edge, more of the original pheromone is in place and so their degree of exploration decreases. Figure 4 outlines the general PFS algorithm.

```
procedure PFS
   if ( canStep() )
      mem = getPheromoneMatrix();
      for i := 1 to numSteps
         vi= selectRandomVertex();
         numProps  = min{N-vi, propDist}
         for k := 1 to numProps
            max  = findMaxPheromone(vi);
            subPath = findAllVerticesAfter(vi)
            for j := 1 to vi.numEdges()
               if vj ∈ subPath
                  if ( mem[vi][j] ≠ max )
                        mem[vi][j] = PF(mem[vi][j]);
                  else if ( i = 1 )
                        mem[vi][j] = PRF(mem[vi][j]);
            end for;
            vi := vi+1;
         end for;
      end for;
end procedure;
```

Figure 4 Propagation based Foot Stepping algorithm

The significant difference from NPFS in Figure 3 is the addition of the extra ***for*** *k := 1* ***to*** *numProps* loop. This structure controls the number of propagations to perform and the propagation distance (distance is the number of vertices to perform the propagation on). The number of propagation steps is limited to either the end of *G*, signaled by N-v_i, or a predefined constant *propDist*. Since the PFS algorithm is run on the global best path, a *subPath* is constructed that holds all valid vertices to propagate. In this paper, propagation is always towards the end of *G* from the footstep, although a radial function is possible that would traverse in all directions from the propagated vertex. Another difference is the use of the PF calculation instead of the usual PAF. The PFS algorithm only reduces pheromone at the original vertex via the PRF. Other propagated vertices will not receive a footstep, however, those edges with large amounts of pheromone will not receive any additional pheromone. The result is that exploration from the propagated vertex will be increased, while not reinforcing current maximum pheromone values on successive edges. This is because an increase in all pheromone values by a quantity *q* will not cause a higher exploration of edges with little pheromone. Reinforcement of the global best solution will continue as in NPFS. The complexity of PFS is O(numSteps·numProps·v_i.numEdges()).

4.4 Dynamic Foot Stepping

In the above descriptions of NPFS and PFS, the value of *numSteps* was declared as a predefined constant. A weakness with using a constant number of steps is that it is likely that not enough fluctuations occurred to adequately explore the search space. Furthermore, there is also a possibility that too many steps occur, making it difficult for the ants to discover small improvements in solution quality that may result from a subtle fluctuation. A solution to this dilemma was to dynamically decide the value of *numSteps.*

It was explained above (Sec. 4.2) that the value of *numSteps* determines a search radius around the global optimum solution. The larger the value of *numSteps*, the further the search is exploring from its center (center is the currrent best solution). An initial number of steps must be defined before the algorithm runs, however, it will possess the capability to automatically adjust the size of the search radius, and ultimately the degree of exploration/exploitation exhibited by the ants.

4.5 Max-Min Ant System

The Max-Min Ant System was developed by T. Stutzle, H. Hoos and outlined in [16]. MMAS was designed to obtain a stronger exploitation of the best solutions found when searching, while at the same time avoiding premature stagnation. It differs from the classic AS in three major ways:

- An elitism model is used that utilizes not only the global best solution, but also uses the iteration best solution.
- The concept of minimum and maximum pheromone values $[\tau_{min}, \tau_{max}]$ along an edge (i,j) is utilized to aid in avoiding premature stagnation.
- Initially all edges of G are initialized to τ_{max}, in this way the MMAS causes a higher exploration rate at the onset of the algorithm.

The most important aspect of MMAS focuses on an elitism model that uses either the global or iteration best solution for pheromone trail update. In this manner, those iteration best solutions that frequently appear as global best solutions receive extra reinforcement. The rate at which iteration best solutions are used is experimentally decided, and will have a direct effect on the extent of exploration.

In order to avoid stagnation, the amount of pheromone on each edge of G is restricted to a range $[\tau_{min}, \tau_{max}]$. It was shown that τ_{max} should be set to:

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{\Phi_{best}(E,\tau)} \tag{5}$$

where *best* refers to either the iteration or global best solution. The lower bound τ_{min} should be set to:

$$\tau_{min} = \frac{\tau_{max}(1-p_{dec})}{(avg-1)p_{dec}} \tag{6}$$

where *avg* represents the average number of possible choices an ant is presented with at each decision point. The value p_{dec} is the probability of making the correct choice at each decision point. Thus, if p_{best} represents the probability of finding the best solution then p_{dec} can be calculated by:

$$p_{dec} = \sqrt[n]{p_{best}} \tag{7}$$

where n represents the size of a path. If it occurs that $\tau_{min} > \tau_{max}$, then $\tau_{min} = \tau_{max}$, in this case only heuristic information is being used to determine choices at decision points. Due to equation (5), values for τ_{min} and τ_{max} will dynamically change as better solutions are found.

Following an iteration every edge is tested to ensure its pheromone quantity lies within $[\tau_{min}, \tau_{max}]$. Any quantities lying outside will be reset to either τ_{min} or τ_{max}, depending if the amount was too small or too large. The final characteristic of MMAS is its pheromone initialization. The amount of pheromone initially placed on each edge of G is set to τ_{max}. As a result a higher level of exploration is forced.

5 Experimental Results

The proposed ant based optimization approach was tested using well-known JSSP instances found at OR-Library [17]. The results were compared to the best-known published results (based on various metaheuristics).The performance of our approach was also compared with, the MAX-MIN algorithm. All simulations were run on an Intel Pentium 4 CPU at 1.6 GHz, 512MB RAM running Microsoft Windows 2000. The programming language used was Java 1.4.

The parameters used for the test runs are: $\alpha = 0.85, \beta = 0.15, \rho = 0.95$. The number of ants in each generation was $m = numMachines * numJobs$**.** Every test was carried with 10 runs on every instance and every run was stopped after 3000 iterations. The

initial pheromone along each edge of G was set to 10. The choices for α and β cause a high amount of exploration, whereas the relatively large ρ value allows for a quick enough convergence, while still permitting a fair amount of exploration. All experiments were utilizing the global best solution to guide the search, implemented as an elitist strategy.

In order to implement the NPFS and PFS techniques the PAF, PRF, and PF equations must be specified. Referring to the respective NPFS and PFS algorithms, they are as follows:

$$\mathrm{PAF}(x) = x + \mathrm{v}^{\max}{}_{\mathrm{i}} \tag{8}$$

$$\mathrm{PRF}(\mathrm{x}) = \frac{x}{1000} \tag{9}$$

$$\mathrm{PF}(\mathrm{x}) = \mathrm{x} + \frac{1000 * \mathrm{v}^{\max}{}_{\mathrm{i}}}{i - \mathrm{v}_{\mathrm{i}} + 1} \tag{10}$$

The NPFS algorithm was tested for values of *numSteps* = 1, 3, 5. For instances la01-la10, *numSteps* = 3 yielded better results, while *numSteps*=5 seemed to achieve higher quality solutions for larger instances. The PFS algorithm was run with 1, 2, 3 footsteps, of which *numSteps* = 2 yielded the better results over all instances.

Figure 5 illustrates a typical run of the NPFS and PFS algorithm for the Lawrence 15 problem. Figure 6 shows a comparison of constant versus dynamic Foot Stepping for the la17 problem instance. The first footstep set occurred near iteration 200, where both solutions show an almost immediate improvement. Another set of footsteps occurred near iteration 260, where the constant FS technique shows no improvement. The lack of improvement is due to a large fluctuation in pheromone values. This does not allow ants to key in on a new optimal solution. The dynamic technique, finding an improvement in the last FS session, attempts to improve the solution at closer search radius. A smaller number of fluctuations occurred and actually in all five runs, an improvement was shown. The result is that the dynamic approach achieved a solution within a full 5% accuracy relative to the best-known solution of 784.

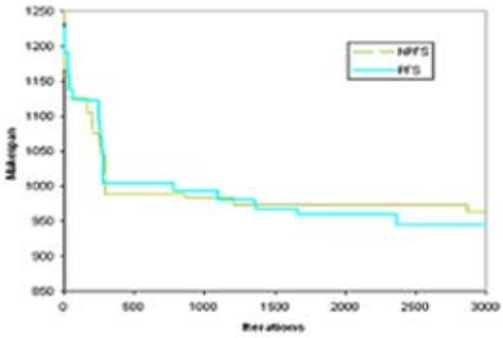

Figure 5 Typical run for NPFS and PFS

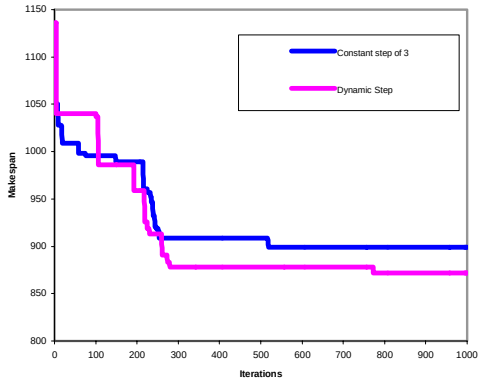

Figure 6 Constant versus dynamically decided number of footsteps for problem instance la17. The graph is of the average of the best solutions averaged over five runs**.**

Tables 1-3 depict a summary of the Lawrence problem instances up to la20. The column labeled *Best Known* in Tables 1 and 2, shows the best known (published) solution for the given problem instance. The values given for the columns labeled *static NPFS* (or *PFS*) and *dynamic NPFS* (or *PFS*) in Table 1 and 2 are the average lengths of the best schedules over 10 runs, whereas the column labeled *Best* gives the best schedule found among the 10 runs. The values in boldface font in Tables 1 and 2 indicate that the best known solution was found. In Table 1, out of 20 problem instances, 16 solutions found by the dynamic NPFS were better than or equal to the solutions by static NPFS. In Table 2, dynamic PFS had 14 of 20 instances better or equal to static PFS. It can be seen from Tables 1 and 2 that using dynamic steps in the foot stepping strategy improves on the static implementation. Table 3 gives a comparison of the best solution found for MMAS, PFS and NPFS. The shaded values indicate the best solution found in a given instance. For most of these problems, all three techniques (MMAS, PFS and NPFS) were able to determine a large number of best-known solutions. Specifically concerning the

PFS and NPFS algorithms, as the instances became larger NPFS was encountering increasing difficulty in determining good solutions. It seems that PFS may be a more powerful technique, however future analysis on these techniques must be performed. Notice that the PFS algorithm failed to find high quality solutions for the smaller la02 – la04 instances. This occurred because too many fluctuations were caused, and the ants never had the opportunity to efficiently explore the solution space. This then strongly implies that the choice of pheromone propagation function (PF described above) is the key to finding good solutions. Lastly, Table 3 shows that the proposed NPFS and PFS approaches are very competitive with results obtained via the MMAS based approach.

Table 1: Results for Static and Dynamic Non-Propagation based Foot Stepping

Instance	Best Known	Static NPFS	Dynamic NPFS	Best
la01	666	671	667	**666**
la02	655	690	674	666
la03	597	636	624	617
la04	590	639	616	611
la05	593	593	593	**593**
la06	926	926	926	**926**
la07	890	917	919	894
la08	863	870	867	**863**
la09	951	951	951	**951**
la10	958	958	958	**958**
la11	1222	1223	1225	**1222**
la12	1039	1050	1047	1040
la13	1150	1151	1151	**1150**
la14	1292	1292	1292	**1292**
la15	1207	1301	1315	1298
la16	945	1062	1044	1036
la17	784	888	861	854
la18	848	952	964	957
la19	842	977	968	959
la20	902	1026	1008	1000

Table 2: Results for Static and dynamic Propagation based Footstepping

Instance	Best Known	Static PFS	Dynamic PFS	Best
la01	666	671	670	**666**
la02	655	704	690	690
la03	597	644	644	642
la04	590	641	622	610
la05	593	593	593	**593**
la06	926	933	930	**926**
la07	890	931	942	929
la08	863	910	880	**863**
la09	951	951	951	**951**
la10	958	958	958	**958**
la11	1222	1233	1225	**1222**
la12	1039	1053	1047	**1039**
la13	1150	1153	1159	**1150**
la14	1292	1292	1292	**1292**
la15	1207	1306	1329	1286
la16	945	1054	1039	1028
la17	784	879	878	854
la18	848	946	975	941
la19	842	971	966	963
la20	902	964	1021	951

Table 3: Results for MMAS, best NPFS and best PFS

Instance	MaxMin[1]	NPFS	PFS
la01	666	666	666
la02	671	666	690
la03	624	617	642
la04	607	611	610
la05	593	593	593
la06	926	926	926
la07	895	894	929
la08	863	863	863
la09	951	951	951
la10	958	958	958
la11	1222	1222	1222
la12	1039	1040	1039
la13	1150	1150	1150
la14	1292	1292	1292
la15	1288	1298	1286
la16	1009	1036	1028
la17	878	854	854
la18	916	957	941
la19	975	959	963
la20	1001	1000	951

[1] MMAS runs in the same range of time as does NPFS and PFS

6 Conclusions

This paper presented an ant colony optimization algorithm approach to the job shop-scheduling problem. The main goal was to examine pheromone-updating techniques and their effects on solution space exploration. A rudimentary Ant System was examined along with a Max-Min Ant System approach. The latter is a well known extension of the AS, that utilizes pheromone boundaries and the transition from iteration to global best solution for pheromone updating. A new technique called Foot Stepping was proposed that improves the solution quality of the ant system. Foot Stepping dynamically alters the environment of the colony in such a way as to cause a higher degree of exploration, in actuality the FS causes the ants to perform local search. Two variations of FS were proposed, Non-Propagation and Propagation based. These differ in that the latter is actually capable of propagating pheromone through the environment. The propagation is based on a number of central propagation points and continues outward causing a decreasing amount of perturbations in the pheromone. MMAS and FS differ in another major manner; FS is run as a daemon activity. Experimental results show that our proposed approach gave competitive schedule length compared to MMAS and well-known benchmark best solutions for the problem instances tested.

Future work involves determining statistical methods for locating propagation points, as well as an investigation into possible pheromone propagation and reduction functions. By examining these key issues, the FS technique can be fully exploited. Other types of problems should be investigated using FS.

Acknowledgments

This research is supported by the Natural Science and Engineering Research Council of Canada Grant No. 249891-02.

Bibliography

[1] M.Dorigo and G.Di Caro, "The ant colony optimization meta-heuristic," in *New Ideas in Optimization*, edited by D. Corne, M.Dorigo, and F. Glover, McGraw-Hill, pp. 11-32, 1999.

[2] M.R.Garey, and D.S, "Computers and Intractability, A Guide to the Theory of NP-Completeness", W.H Freeman and Company, 1979.

[3] S. Goss, S.Aron, J. L. Deneubourg, and J. M. Pasteels.*Self-organized shortcuts in the Argentine Ant*.Naturwissenschaften, 76:579-581, 1990

[4] Dorigo M., G. Di Caro & L. M. Gambardella. "Ant Algorithms for Discrete Optimization"**,** *Artificial Life,* 5(2):137-172, 1999.

[5] T. Stutzle, "An ant approach for the flow shop problem", *In Fifth European Congress on Intelligent Techniques Soft Computing* (EUFIT'98), vol.3, Verlag Mainz, Aachen, pp. 1560-1564, 1998.

[6] D. Merkle and M. Middendorf and H. Schmeck, "Ant Colony Optimization for Resource Constrained Project Scheduling", *IEEE Transactions on Evolutionary Computation*, 6(4):333-346,2002.

[7] D. Merkle and M. Middendorf, "Ant Colony Optimization with Global Pheromone Evaluation for Scheduling a Single Machine", *Journal of Applied Intelligence*, Kluwer Academic Publishers, pp, 105-111, 2003.

[8] Dorigo M. & L.M. Gambardella , "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation,* 1(1):53-66, 1997.

[9] Bullnheimer B., R. F. Hartl & C. Strauss, **"**A New Rank Based Version of the Ant System: A Computational Study", *Central European Journal for Operations Research and Economics,* 7(1):25-38, 1999.

[10] L. M, Gambardella, E. Taillard and G. Agazzi, "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Window" In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pp.63-76, McGraw-Hill, London, 1999.

[11] C. Blum and M. Sampels, "An ant Colony Optimization Algorithm to tackle Shop Scheduling Problems", Submitted.

[12] Colorni A., M. Dorigo, V. Maniezzo and M. Trubian , " Ant system for Job-shop Scheduling**.**", *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science,* 34(1):39-53, 1994.

[13] S.van der Zwaan, C. Marques "Ant Colony Optimization for Job Shop Scheduling", In *proceedings of Third workshop on Genetic Algorithms and Artificial Life*, 1999.

[14] C. Blum and M. Sampels, ""An ant Colony Optimization Algorithm for FOP Shop Scheduling: A case study on different pheromone representations", *Proceedings of the 2002 Congress on Evolutionary Computation*, CEC'02, pp.1558-1563, 2002.

[15] Glover F., "Tabu Search - Part II", *ORSA Journal on*Computing", 2(1), Winter, 4-32, 1990.

[16] T. Stutzle, H. Hoos, "The Max-Min Ant System", *Future Generation Computer Systems,*16(8): 889-914, 2000.

[17] OR Library, URL http://mscmga.ms.ic.ac.uk