18

# A Particle Swarm Optimisation Approach to Graph Permutations

Omar Ilaya and Cees Bil
*RMIT University*
*Australia*

## 1. Introduction

In many real-world applications, the arrangement, ordering, and selection of a discrete set of objects from a finite set, is used to satisfy a desired objective. The problem of finding optimal configurations from a discrete set of objects is known as the *combinatorial optimisation problem*. Examples of combinatorial optimisation problems in real-world scenarios include network design for optimal performance, fleet management, transportation and logistics, production-planning, inventory, airline-crew scheduling, and facility location.
While many of these combinatorial optimisation problems can be solved in polynomial time, a majority belong to the class of NP -hard (Aardal et al., 1997). To deal with these hard combinatorial optimisation problems, approximation and heuristic algorithms have been employed as a compromise between solution quality and computational time (Festa and Resende, 2008). This makes heuristic algorithms well-suited for applications where computational resources are limited. These include dynamic ad-hoc networks, decentralised multi-agent systems, and multi-vehicle formations. The success of these heuristic algorithms depends on the computational complexity of the algorithm and their ability to converge to the optimal solution (Festa and Resende, 2008). In most cases, the solutions obtained by these heuristic algorithms are not guaranteed optimal.
A recently developed class of heuristic algorithms, known as the *meta-heuristic algorithms*, have demonstrated promising results in the field of combinatorial optimisation. Meta-heuristic algorithms represent the class of all-purpose search techniques that can be applied to a variety of optimisation problems including combinatorial optimisation. The class of meta-heuristic algorithms include (but not restricted to) simulated annealing (SA), tabu search, evolutionary algorithms (EA) (including genetic algorithms), ant colony optimisation (ACO) (Aguilar, 2001), bacterial foraging (Passino, 2002), scatter search, and iterated local search.
Recently, a new family of computationally efficient meta-heuristic algorithms better posed at handling non-linear constraints and non-convex solution spaces have been developed. From this family of meta-heuristic algorithms, is particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995). Like other biologically inspired meta-heuristic algorithms, PSO is an adaptive search technique that is based on the social foraging of insects and animals. In PSO, a population of candidate solutions are modelled as a swarm of particles. At each iteration, the particles update their position (and solution) by moving stochastically

towards regions previously visited by the individual particle and the collective swarm. The simplicity, robustness, and adaptability of PSO, has found application in a wide-range of optimisation problems over continuous search spaces. While PSO has proven to be successful on a variety of continuous functions, limited success has been demonstrated to adapt PSO to more complex richer spaces such as combinatorial optimisation.
In this chapter, the concepts of the standard PSO model are extended to the discrete combinatorial space and a new PSO is developed to solve the combinatorial optimisation problem. The chapter is organised as follows: In Section 2, a brief review of related works to solving the combinatorial optimisation space using meta-heuristics is presented. In Section 3, the standard PSO model is introduced. The nature of the combinatorial optimisation problem is then presented in Section 4 before the concepts of the standard PSO model are adapted to the combinatorial space in Section 5. Section 6 analyses the stability and performance of the newly developed algorithm. The performance of the newly developed algorithm is then compared to the performance of a traditional genetic algorithm in Section 7 before Section 8 concludes with final remarks.

## 2. Related Works

In recent years, variants of traditional PSO have been used to solve discrete and combinatorial optimisation problems. A binary PSO was first developed in (Kennedy and Eberhart, 1997) to solve discrete optimisation problems. In the binary PSO, each particle encoded a binary string in the solution space. A particle moved according to a probability distribution function determined using the Hamming distance between two points in the binary space. The early concepts introduced by the binary PSO appeared in later PSO algorithms for combinatorial optimisation such as in (Shi et al., 2006); (Tasgetiren et al., 2004); (Liu et al., 2007b); (Pang et al., 2004); (Martínez García and Moreno Pérez, 2008); (Song et al., 2008); and (Wang et al., 2003). Tasgetiren et al. (Tasgetiren et al., 2004) introduced the smallest position value rule (SPV) to enable the continuous PSO algorithm to be applied the class of sequencing and combinatorial problems. In SPV, each particle assigns a position value in continuous space to each dimension in the discrete space. At each iteration, the position value is updated according to the traditional velocity update equation and the sequence of objects is re-sorted according to the values assigned to the continuous space. The method proposed by (Tasgetiren et al., 2004) is similar to the random keys in GA (Bean, 1994). Following a similar method to (Kennedy and Eberhart, 1997), Wang et al. (Wang et al., 2003) introduced the concept of a swap operator to exchange dimensions in the particle position. In (Wang et al., 2003), each particle encoded a permutation of objects and a transition from one position to the next was achieved by exchanging elements in the permutation. To account for both the personal best positions and global best positions, Wang et al. extended the concept of swap operator to swap sequence. The swap sequence was used to move a particle from one position to the next by successively applying a sequence of swap operators. Using this approach, the notion of velocity on the combinatorial space was defined; and the Hamming distance was used to exclusively determine the motion of a particle. Premature convergence was addressed by randomly applying the swap operator to the particle. Similar approaches to Wang et al. include (Shi et al., 2006); (Martínez García and Moreno Pérez, 2008); and (Bonyadi et al., 2007), where a swap sequence was also constructed through the concatenation of successive swap operators. The ordering of these swap operators influences the position of the particle at the end of each iteration. In (Wang et al., 2003); (Shi et al., 2006); (Martínez García and Moreno Pérez, 2008); and (Bonyadi et al., 2007), the swap sequence is constructed by first applying the swap operators that move the particle to it's personal best, followed by the swap operators that move the particle to it's global best. For sufficiently small perturbations, the particles will tend towards the global best position of the swarm and stimulate the loss of solution diversity. This invariably leads to the rapid convergence of the algorithm and poor solution quality. For large complex optimisation problems, the PSO must compromise the local and global search strategies effectively to find high-quality (if not optimal) solutions rapidly. In addition, the PSO framework must be sufficiently robust to adapt to a wide variety of discrete and combinatorial optimisation problems. In this chapter, a generalised combinatorial optimisation framework is introduced that builds on the works of (Wang et al., 2003); (Shi et al., 2006); (Tasgetiren et al., 2004); and (Kennedy and Eberhart, 1997) to develop a new combinatorial optimisation PSO. In the following section, a brief introduction into the traditional PSO is presented before the main results of this chapter are developed.

## 3. The Standard Particle Swarm Optimisation Model

Let $\mathcal{P}$ denote a $D$-dimensional problem, and $f(x): X \to \mathbb{R}$ an objective function for the problem that maps $X$ to the set of real numbers. Without loss of generality, consider the following optimisation problem $x^* \in X \Leftrightarrow x^* = \arg\min_{x\in X} f(x) \quad \forall x \in X$. In traditional PSO, a solution $i$ is represented by a particle in a swarm $P$ moving through $D$-dimensional space with position vector $x_k^i = (x_k^i(1),\ldots,x_k^i(d),\ldots,x_k^i(D))$ for any time $k$. At each iteration, the particles adjust their velocity $v_k^i$ along each dimension according to the previous best position of the $i$-th particle $p_k^i$ and the best position of the collective swarm $p_k^g$ (see Fig. 1). The position $x_k^i$ for the $i$-th particle is updated according to the following velocity function:

$$v_{k+1}^i = w \cdot v_k^i + c_1 \cdot r_1 \cdot (p_k^i - x_k^i) + c_2 \cdot r_2 \cdot (p_k^g - x_k^i) \tag{1a}$$

$$x_{k+1}^i = x_k^i + v_k^i \tag{1b}$$

where $r_1, r_2 \in [0,1]$ are random variables affecting the search direction, $c_1, c_2 \in \mathbb{R}$ are configuration parameters weighting the relative confidence in the personal best solutions and the global best solutions respectively, and $w$ is an inertia term influencing the momentum along a given search direction. Algorithm 1 summarises the iterative nature of the PSO algorithm.

The terms $c_1$ and $c_2$ are the main configuration parameters of the PSO that directly influence the convergence of the algorithm. For large values of $c_1$, exploration of particles is bounded to local regions of the best previously found solutions $p_k^i$. This maintains population diversity and is favourable when the problem is characterised by non-linear and non-convex solution spaces. In contrast, large $c_2$ values will encourage particles to explore regions closer to the global best solution $p_k^g$ at each iteration. Generally, this search strategy will converge faster and is practical for convex solution spaces with unique optima. Adjusting the inertia term $w$ affects the relative weighting of the local and global searches. A large $w$ encourages the particles to explore a larger region of the solution space at each iteration and maximise global search ability, whilst a smaller $w$ will restrict the particles to local search at each iteration (Shi and Eberhart, 1998b).

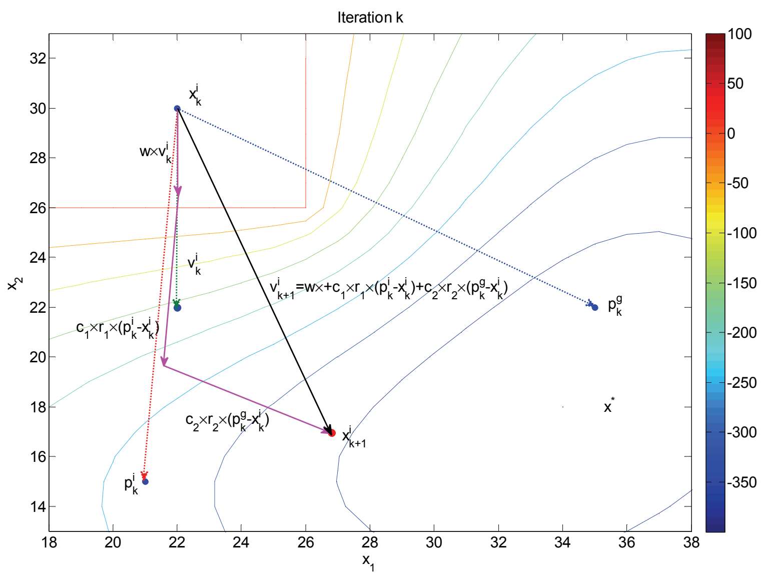

Iteration k
$x_k^i$
$w\times v_k^i$
$v_k^i$
$v_{k+1}^i=w\times+c_1\times r_1\times(p_k^i-x_k^i)+c_2\times r_2\times(p_k^g-x_k^i)$
$p_k^g$
$c_1\times r_1\times(p_k^i-x_k^i)$
$c_2\times r_2\times(p_k^g-x_k^i)$
$x_{k+1}^i$
$x^*$
$p_k^i$
$x_1$
$x_2$

Figure 1. Particle position and velocity on a two-dimensional vector space

```
0:  for all particle $i$ do
1:      initialise position $x_k^i$ randomly in the search space
2:  end for
3:  while termination criteria not satisfied do
4:      for all particle $i$ do
5:          set personal best $p_k^i$ as the best position found by the particle so far
6:          set global best $p_k^g$ as the best position found by the swarm so far
7:      end for
8:      for all particle $i$ do
9:          update velocity according to
            $v_{k+1}^i = w\cdot v_k^i + c_1\cdot r_1\cdot(p_k^i - x_k^i) + c_2\cdot r_2\cdot(p_k^g - x_k^i)$
10:         update position according to
            $x_{k+1}^i = x_k^i + v_k^i$
11:     end for
12: end while
```

Algorithm 1. Traditional PSO

## 4. Problem Description and Model Construction

The combinatorial optimisation problem for PSO is now discussed. Let $X=\{x^1,x^2,\ldots,x^i,\ldots\}$ denote the finite set of solutions to the combinatorial optimisation problem with objective function $f:X\to\mathbb{R}$. Assume the objective of the combinatorial optimisation problem is to find $x^*\in X$, such that $x^*\in X \Leftrightarrow x^* = \arg\min_{x\in X} f(x) \quad \forall x\in X$. Consider the case where a solution $x^i \in X$ to the combinatorial optimisation problem is given by the linear ordering of elements in the set $[n] = \{1,2,\ldots,n\}$, such that $\forall x^i \in X$, $x^i = (x^i(1), x^i(2),\ldots,x^i(n)) \in \{1,2,\ldots,n\}$. Then $|X| = n!$. Each integer value in the list encodes the relative ordering of a set of objects and is referred to as a *permutation* of objects (Bóna, 2004). These include cities in a tour, nodes in a network, jobs in a schedule, or vehicles in a formation. For convenience, a permutation is represented using *two-line form*. Let $g:[d] \to [n]$ be a bijection on the ordered list. If $[n]$ describes the list of numbers $[n] = \{1,\ldots,n\}$, then $[d] = \{1,\ldots,n\}$ and $g$ is also a permutation of the set $[n]$ (Bóna, 2004).

**Example 1.**

As an example, consider the following permutation $\{3,4,1,5,2\}$. The function $g:[5] \to [5]$ defined by $g(1) = 3$, $g(2) = 4$, $g(3) = 1$, $g(4) = 5$, and $g(5) = 2$ is also permutation of $[5]$ (Bóna, 2004). In two-line form, the set $[5]$ can be written as:

$$g = \begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{matrix}$$

where it is implied that $g$ maps 1 to 3, 2 to 4, 3 to 1, 4 to 5, and 5 to 2.

## 5. Fitness Landscape

In order to adapt PSO to the combinatorial space, it is convenient to define a metric space characteristic of the combinatorial optimisation problem. Let $\mathcal{N}: X \to 2^X$ denote a syntactic neighbourhood function that attaches to each solution $x^i \in X$ the neighbouring set of solutions $x^j \in \mathcal{N}_i(x^i) \subseteq X$ that can be reached by applying a unitary syntactic operation moving $x^i \mapsto x^j$ (Moraglio and Poli, 2004). Denote this unitary syntactic operator by $\varphi$ and assume that the operation is reversible, i.e. $x^j \in \mathcal{N}_i(x^i) \Leftrightarrow x^i \in \mathcal{N}_j(x^j)$. Such a neighbourhood can be associated to an undirected *neighbourhood graph* $G = (V,E)$, where $V$ is the set of vertices representing the solutions $x^i \in X$, and $E$ the set of edges representing the transformation paths for permutations. By definition, the combinatorial space endowed with a neighbourhood structure $\mathcal{N}_i(x^i)$ and induced by a distance function $h_{ij}(x^i,x^j)$ is a *metric space*. Formally, the definition of a metric or distance function is any real valued function $h_{ij}(x^i,x^j)$ that conforms to the axioms of identity, symmetry, and triangular inequality, i.e.:

1. $h_{ij}(x^i,x^j) \ge 0$ and $h_{ij}(x^i,x^i) = 0$ (identity);
2. $h_{ij}(x^i,x^j) = h_{ij}(x^j,x^i)$ (symmetric);
3. $h_{ij}(x^l,x^j) \le h_{li}(x^l,x^i) + h_{ij}(x^i,x^j)$ (triangle inequality);
4. if $i \ne j$, then $h_{ij}(x^i,x^j) > 0$.

A neighbourhood structure $\mathcal{N}_i(x^i)$ induced by a distance function $h_{ij}(x^i,x^j)$ can then be formally expressed as:

$$\mathcal{N}_i(x^i) = \{x^j \mid x^j \in X, h_{ij}(x^i,x^j) \le s\} \tag{2}$$

where $s \in \mathbb{R}$. On a combinatorial space with syntactic operator $\varphi$, any configuration $x^i$ can be transformed into any other $x^j$ by applying the operator $\varphi$ a finite number of times ($1 < s \leq n$) (Misevicius et al., 2004). In such a case, the distance metric $h_{ij}(x^i, x^j)$ is given by the Hamming distance:

$$h_{ij}(x^i, x^j) = \sum_{l=1}^{n} \text{sgn}\left|x^i(d) - x^j(d)\right|$$

and $s$ represents the minimum number of exchanges to transform $x^i$ into $x^j$. Other distance metrics can be similarly defined (see (Ronald, 1997); (Ronald, 1998); and (Moraglio and Poli, 2004) references therein for a comprehensive treatment on distance metrics defined on the combinatorial space).

For generality, only the *deviation distance metric* (Ronald, 1998) will be considered hereafter. While other distance metrics can be defined for discrete and combinatorial spaces, the decision to use the deviation distance metric is trivial with respect to algorithmic design. Other problem-specific metrics can be substituted into the developed algorithm with little influence on the procedural implementations of the algorithm.

The deviation distance metric provides a measure of the relative distance of neighbouring elements between two permutations $x^i$ and $x^j$. In problems where the adjacency of two elements influences the cost of the objective function $f(x)$, such as in TSP and flow-shop scheduling, the deviation distance function provides an appropriate choice of metric for the problem space (Ronald, 1998). Formally, the positional perturbation $\Delta_a$ of one element value $x^i(d_1)$ to its matching value in $x^j(d_2)$, such that $x^i(d_1) = x^j(d_2) = a$, $a \in [n]$, is given by the following:

$$\Delta_a = \left|d_1 - d_2\right| \tag{3}$$

For convenience, $\Delta_a$ is normalised $\overline{\Delta}_a \in [0,1]$:

$$\overline{\Delta}_a = \frac{\Delta_a}{n-1} \tag{4}$$

The deviation distance $h_{ij}(x^i, x^j)$ is then defined as the sum of the $\overline{\Delta}_a$ values:

$$h_{ij}(x^i, x^j) = \sum_{a}^{n} \overline{\Delta}_a \tag{5}$$

From Eq. (5) a large position deviation induces a greater distance in the metric space. The notion of position deviation is now used to construct the combinatorial optimisation PSO.

## 6. Proposed Algorithm

In Section 4.1, the concept of a syntactic operator $\varphi$ was discussed as a method of transforming one configuration $x^i$ to another $x^j \in \mathcal{N}_i(x^i)$. In the following section, the parallel between a syntactic operator $\varphi$ and the motion of a particle $i$ in the combinatorial space is described. Let $x^i \in X$ encode a permutation of $[d]$ objects in $D$-dimensional space. The position $x^i \in X$ of a particle $i$ in the $D$-dimensional space corresponds to a permutation of $[d]$ objects. Define $\varphi$ by a two-way perturbation (transformation) operator $\varphi := SO(d_1, d_2)$ as the *swap operator* that exchanges elements $d_1$ and $d_2$ in solution $x^i$, such that $X \to X$, $d_1, d_2 \in \{1,2,\dots D\}$, $d_1 \neq d_2$. Applying the swap operator to the permutation $x^i$, the following solution is derived:

$$x^i_{k+1} = x^i_k \oplus SO(d_1, d_2) \tag{6}$$

where $x^i(d_1) = x^j(d_2) = a$, and $x^j_k, x^{i+1}_k, x^j \in \mathcal{N}_i(x^i_k)$, and the notation $\oplus$ is used to denote $x^i_{k+1}$ is obtained from $x^i_k$ by applying the perturbation $SO(d_1, d_2)$. In the combinatorial optimisation PSO, $x^i_k$ and $x^j_k \in \mathcal{N}_i(x^i_k) \subseteq X$, $x^i_k \neq x^j_k$ encode two permutations in the combinatorial optimisation problem and represents positions in the combinatorial search space. Applying the notions of swap operator to PSO, the swap operator $SO(d_1, d_2)$ for a particle $i$ can be interpreted as a motion of the particle $x^i_k$ to a position $x^j_k$ displaced from $x^i_k$ by the deviation distance $h_{ij}(x^i_k, x^j_k)$. Consider the case when $x^j_k \notin \mathcal{N}_i(x^i_k)$. Then, the following transition $x^i_k \mapsto x^j_k$ is not possible by Eq. (6) alone. Define the following *swap sequence* (Knuth, 1998):

$$SS = \{SO_1, SO_2, \dots, SO_n\} \tag{7}$$

where $SS$ is the concatenation of swap operators and the order of the swap operators $SO_i$, $i = 1, \dots, n$ is influential to the final position $x^i_{k+1}$. The minimum number of swap operators required to move $x^i_k \mapsto x^j_k$ is given by the Hamming distance and is referred to as the *basic swap sequence* (Knuth, 1998).

Suppose particle $i$ moves according to $x^i_k \mapsto p^i_k$. The basic swap sequence transforming $x^i_k$ to $p^i_k$ can be determined by moving along each dimension of the initial position $x^i_k$ and applying the Partially Mapped Crossover function (PMX) (Goldberg and Lingle, 1985) to each dimension along $x^i_k$. The PMX function maps each dimension in the current position $x^i_k$ to the corresponding dimension in $p^i_k$ (see Fig. 2). A swap operator is invoked if the object in the $d_1$-th dimension of the $p^i_k$ solution and the $x^i_k$ are inconsistent. The $d_1$-th element in $x^i_k$ is then swapped with the $d_2$-th element in $x^i_k$ such that $x^i_k(d_2) = p^i_k(d_1)$. Algorithm 2 summarises the basic swap operator used to move $x^i_k \mapsto p^i_k$

```
1: while $d(x^i, x^j) \neq 0$
2:     if $x^i_k(d_1) \neq x^j_k(d_1)$ then
3:         find $d_2$ such that $x^i_k(d_2) = x^j_k(d_1) = a_1$, and $d_1, d_2 \in \{1, \dots, D\}$
4:             set $SO_j(d_1, d_2)$ and store as $j$-th entry in $SS$
5:     else, end if
4: end while
```

Algorithm 2. Basic Swap Operator

Note, applying the algorithm from left-right gives $d_2 > d_1$, $d_1, d_2 \in \{1,2,\ldots,D\}$.

**Example 2.**

Consider the following two solutions $x^i = \binom{1\ 2\ 3\ 4\ 5}{1\ 2\ 3\ 4\ 5}$ and $x^j = \binom{1\ 2\ 3\ 4\ 5}{2\ 3\ 1\ 5\ 4}$ represented in two-line form. Applying Algorithm 2 from left to right, the first swap operator is invoked if $x^i(1) \neq x^j(1)$. Since $x^i(1) = 1$ and $x^j(1) = 2$, the following mapping is observed between object $1 \rightarrow 2$. The first swap operator is then given by the exchange of elements 1 and 2 in $x^i$, $SO_1(1,2)$. Following $SO_1(1,2)$, particle $i$ is now at position $x' = \binom{1\ 2\ 3\ 4\ 5}{2\ 1\ 3\ 4\ 5}$. Comparing $x'$ to $x^j$, the following mapping $1 \leftrightarrow 3$ is now observed between object $x'(2)$ and $x^j(2)$. The next mapping is then given by $SO_2(2,3)$ taking $x'$ to $x'' = \binom{1\ 2\ 3\ 4\ 5}{2\ 3\ 1\ 4\ 5}$. Repeating this procedure, the swap sequence $SS$ that takes $x^i$ to $x^j$ is then given by $SS = \{SO_1(1,2), SO_2(2,3), SO_3(4,5)\}$ such that $x^j = x^i \oplus SS$.

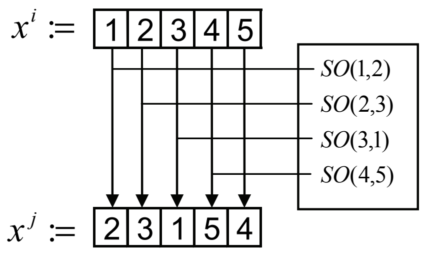


Figure 2. Partially-mapped crossover (PMX)

In traditional PSO, the motion of a particle is influenced by the personal best position $p_k^i$ and global best of the swarm $p_k^g$. In the combinatorial optimisation PSO, each position encodes a permutation to the combinatorial optimisation problem. If the personal best and global best positions are not coincident, i.e. $p_k^i \neq p_k^g$, then the swap sequences $SS_1$ and $SS_2$ that moves the $i$-th particle along the transformations $x_k^i \mapsto p_k^i$ and $x_k^i \mapsto p_k^g$ respectively, are not equivalent, i.e. $SS_1 \neq SS_2$. Application of $SS_1$ or $SS_2$ will yield $x_{k+1}^i = p_k^i$ or $x_{k+1}^i = p_k^g$ and will cause the particles to converge towards the personal best solution, or the global best solution respectively. This leads to rapid convergence and sub-optimal solution quality. The local search induced by the exclusive application of $SS_1$, and the global search induced by the exclusive application of $SS_2$ is now combined to develop a velocity update function with similar characteristics to the original PSO algorithm.

In the traditional PSO algorithm, the velocity of a particle is composed of three parts; the *momentum term*, i.e. $v \cdot w$, the *cognitive velocity* $c_1 \cdot r_1 \cdot (p_k^i - x_k^i)$, and the *social velocity* $c_2 \cdot r_2 \cdot (p_k^g - x_k^i)$. Using the notions of *momentum*, *cognitive velocity*, and *social velocity*, the following decoupled velocity update for a particle in the combinatorial space with deviation distance metric $\Delta_a$ is defined:

$$v_{k+1}^{l,i} = w \cdot v_k^{l,i} + c_1 \cdot (\Delta'_a(x_k^i, p_k^i)) \tag{8a}$$

$$v_{k+1}^{g,i} = w \cdot v_k^{g,i} + c_2 \cdot (\Delta'_a(x_k^i, p_k^g)) \tag{8b}$$

where $w$, $c_1$, and $c_2$ have the same meanings as the original PSO algorithm. For convenience, denote Eq. (8a) as the *local velocity* and Eq. (8b) as the *global velocity*. Equation (8a) and (8b) preserve the same tuning parameters as the original PSO without the random variables $r_1, r_2 \in [0,1]$. The decision to omit the random variables is trivial, but will become apparent in the proceeding section.

Recall, the position of each particle $x_k^i$, $\forall i \in P$ is a vector in the $D$-dimensional combinatorial space $x_k^i \in X$ and moves along the dimensions of the $D$-dimensional hypercube by exchanging elements via the swap operator $SO(d_1, d_2)$. The velocity of each particle $v_k^i$, $\forall i \in P$ is a vector in the $D$-dimensional *continuous* space $v_k^i \in \mathbb{R}^D$ and describes the local gradient of the fitness landscape using the deviation distance metric. Using the velocity $v_k^i \in \mathbb{R}^D$, a probability mapping is described that invokes the swap operator and preserves the contributions of both the local velocity and global velocity. Let $\Pr(x^i(d) \mid p^i(d))$ and $\Pr(x^i(d) \mid p^g(d))$ denote the sampling probability of the $i$-th particle for dimension $d$ in the particle when the individual best is $p^i(d)$ and global best is $p^g(d)$ respectively. Then, the probability that $x^i(d)$ moves to $p^i(d)$ and $p^g(d)$ is given by the following statements:

$$\Pr(x_k^i(d) \mid p_k^i(d)) := v_k^{l,i} \tag{9a}$$

$$\Pr(x_k^i(d) \mid p_k^g(d)) := v_k^{g,i} \tag{9b}$$

Since $p^i(d)$ and $p^g(d)$ is a mapping for $x^i(d) \mapsto p^i(d)$ and $x^i(d) \mapsto p^g(d)$ respectively, the probability that the swap operator $SO_j(d_1, d_2)$ is invoked by moving $x^i(d) \mapsto p^i(d)$ or $x^i(d) \mapsto p^g(d)$ using Algorithm 2 is defined using the local and global velocities respectively:

$$\Pr(SO(d_1, d_2)) := v_k^{l,i} \tag{10a}$$

$$\Pr(SO(d_1, d_2)) := v_k^{g,i} \tag{10b}$$

where $x^i(d_2) = p^i(d_1)$ or $x^i(d_2) = p^g(d_1)$ for $x^i(d) \mapsto p^i(d)$ and $x^i(d) \mapsto p^g(d)$ respectively. Following Eq. (10a) and Eq. (10b), the velocity $v_k^i(d)$ describes the probability that an element in $x_k^i(d)$ will swap with the corresponding element in $x_k^j(d)$ and invoke Algorithm 2, then the velocity on each dimension $d \in D$ must be bounded over the interval $v_k^i(d) \in [0,1]$. The velocities described in Eq. (10a) and Eq. (10b) are normalised according to:

$$v_{k+1}^{l,i} = \frac{v_{k+1}^{l,i}}{\arg\max\{v_{k+1}^{l,i}, v_{k+1}^{g,i}\}} \tag{11a}$$

$$v_{k+1}^{g,i} = \frac{v_{k+1}^{g,i}}{\arg\max\{v_{k+1}^{l,i}, v_{k+1}^{g,i}\}} \tag{11b}$$

Normalising the velocities with respect to both the personal best and global best velocity profiles is used to prioritise the order of swap operations and preserve the probability map. Once an element $x_k^i(d_1)$ has been swapped with the corresponding element $x_k^i(d_2)$ in $p_k^i(d_1)$, the associated velocity $v_k^i(d_2)$ at element $x_k^i(d_2)$ is set to zero if $x_k^i(d_2) = p_k^i(d_2)$ to prevent cyclic behaviour.

Using the definition of the sample probability in Eq. (10a) and Eq. (10b) for the personal best and global best respectively, the swap sequence induced by the combinatorial optimisation PSO can now be described. From Eq. (8a) and Eq. (8b), large deviation distances incur a large velocity. This observation is complimentary to the original concepts of the traditional PSO algorithm. Following Eq. (10a) and Eq. (10b) a large velocity will induce a greater probability that a swap operation is invoked with either the personal best or global best. Using this concept, a swap sequence can be defined using the relative probabilities of the personal best and global best velocity profiles. Consider the case when $v_k^{l,i}(d) > v_k^{g,i}(d)$. Then, the probability of exchanging $x_k^i(d) \mapsto p_k^i(d)$ is greater than the probability of exchanging $x_k^i(d) \mapsto p_k^g(d)$. In the swap sequence, the larger of the two probabilities will receive a higher priority in the swap sequence and take precedence over the lower probability swap operations. At a given iteration, particle $i$ will move according to the following swap sequence:

$$x_{k+1}^i = x_k^i \oplus SS \tag{12}$$

where $SS = (SO_1(x_k^i(d), p_k^i(d)), SO_2(x_k^i(d), p_k^g(d)))$ if $v_k^{l,i} > v_k^{g,i}$. Algorithm 3 describes the implementation of the swap sequence $SS$.

```
0:  for all $d \in D$ do
1:      if $v_k^{l,i}(d) > v_k^{g,i}(d)$ do
2:          invoke swap operator $SO_j(d_1, d_2)$ for $x_k^i \mapsto p_k^i$ using Algorithm 2
3:              if $x_k^i(d_2) = p_k^i(d_2)$ do
4:                  set $v_k^{l,i}(d_2) = 0$
5:              else, end if
6:          goto 8
7:      otherwise if $v_k^{g,i}(d) > v_k^{l,i}(d)$ do
8:          invoke swap operator $SO_j(d_1, d_2)$ for $x_k^i \mapsto p_k^g$ using Algorithm 2
9:              if $x_k^i(d_2) = p_k^g(d_2)$ do
10:                 set $v_k^g(d_2) = 0$
11:             else, end if
12:         goto 2
13:     end if
14: end for
```

Algorithm 3. Swap Sequence

Following the definition of the basic swap sequence and swap sequence in Algorithm 2 and Algorithm 3 respectively, the proposed combinatorial PSO algorithm can now be defined. Algorithm 4 describes the procedural implementation of the swap sequence within the context of the traditional PSO algorithm.

## 7. Algorithmic Analysis

The behaviour of each particle in the swarm can be viewed as a traditional line-search procedure dependent on a stochastic step size and a stochastic search direction. Both the stochastic step size and search direction depend on the selection of social and cognitive parameters. In addition, the stochastic search direction is driven by the best design space locations found by each particle and by the swarm as a whole. Unlike traditional line-search procedures however, PSO uses information from neighbouring particles to influence the search direction at each iteration. This exchange of information plays an important role in the stability and performance of the swarm. In the following section, the spectral properties of algebraic graph theory are used to show that for a fully interconnected swarm, the particles will reach a consensus on the equilibrium. The analysis begins by considering the original PSO algorithm with velocity and position update given by Eq. (1a) and Eq. (1b).

```
0:  for all particle $i$ do
1:      initialise position $x_k^i$ randomly in the search space
2:  end for
3:  while termination criteria not satisfied do
4:      for all particle $i$ do
5:          set personal best $p_k^i$ as the best position found by the particle so far
6:          set global best $p_k^g$ as the best position found by the swarm so far
7:      end for
8:      for all particle $i$ do
9:          update local velocity according to
            $v_{k+1}^{l,i} = w \cdot v_k^{l,i} + c_1 \cdot (\Delta'_v(x_k^i, p_k^i))$
10:         update global velocity according to
            $v_{k+1}^{g,i} = w \cdot v_k^{g,i} + c_2 \cdot (\Delta'_v(x_k^i, p_k^g))$
11:         normalise local velocity according to
            $v_{k+1}^{l,i} = v_k^{l,i} / \arg\max\{v_k^{l,i}, v_k^{g,i}\}$
12:         normalise global velocity according to
            $v_{k+1}^{g,i} = v_k^{g,i} / \arg\max\{v_k^{l,i}, v_k^{g,i}\}$
13:         update position according to
            $x_{k+1}^i = x_k^i \oplus SS$
            where $SS$ is determined from Algorithm 3
14:     end for
15: end while
```

**Algorithm 4. Combinatorial Optimisation PSO**

Without loss of generality, consider the following objective function for the combinatorial optimisation problem:

$$x^* = \arg\min_{x\in X} f(x) \ \ \forall x \in X$$

Then, the personal best $p_k^i$ is the current best solution of the $i$-th particle found so far; i.e. $p_k^i = \arg\min_\tau x_\tau^i$, $\forall \tau \in (0,k]$; and the global best $p_k^g$ is the current best solution of the global swarm found so far; i.e. $p_k^g = \arg\min_\tau x_\tau^i$, $\forall \tau \in (0,k]$, $\forall i \in N$. The swarm of particles is said to have reached an equilibria if and only if all the particles have reached a consensus on the value of $p_k^g$, i.e., $p_k^l = p_k^g = p^e$. For asymptotic convergence, all the particles in the swarm must globally asymptotically reach a consensus on the global best solution, such that $x^e = \lim_{k\to+\infty} x_k^i$, and $x_k^{e,i} = x_k^{e,j} = \min(X)$, $\forall i, j \in X$, $i \neq j$. For convenience, Eq. (1a) and Eq. (1b) are combined into compact matrix form:

$$\begin{bmatrix} x_{k+1}^i \\ v_{k+1}^i \end{bmatrix} = \begin{bmatrix} -(c_1r_1 + c_2r_2) & w \\ -(c_1r_1 + c_2r_2) & w \end{bmatrix} \cdot \begin{bmatrix} x_k^i \\ v_k^i \end{bmatrix} + \begin{bmatrix} c_1r_1 & c_2r_2 \\ c_1r_1 & c_2r_2 \end{bmatrix} \cdot \begin{bmatrix} p_k^i \\ p_k^g \end{bmatrix} \tag{13}$$

which can be considered as a discrete-dynamic system representation of the original PSO algorithm.

## 7.1 Equilibrium of the PSO

Before the main analysis results are presented, a brief introduction into algebraic graph modelling of swarms is presented. The information flow in the swarm of particles can be represented using an interconnected graph $G = (V, E)$, where $V$ is the enumerated set of particles $x_k^i \in V$, $i \in \{1,\ldots,N\}$ in the swarm, and $E \subseteq V \times V$ is the set of edge relations between neighbouring particles. The *order* $|V|$ and *size* $|E|$ of the graph $G$ physically represents the number of particles in the swarm and the number of edge connections. For a fully connected swarm, each particle communicates with every other particle in the population, and the graph is said to be complete. This is the case of the original PSO algorithm. The connectivity of a graph is described by the square matrix $A$, with size $|V|$, and elements $a_{ij}$ describing the connectivity of adjacent vertices $x^i$ and $x^j$, such that:

$$a_{ij} = \begin{cases} 1, & \text{if } \left(x^i, x^j\right) \in E \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

The matrix $A$ uniquely defines the connectivity of the graph $G$ and is referred to as the *adjacency matrix*. Associated with the adjacency matrix $A$ is the *graph Laplacian* $L$, and its *Laplacian potential* $\Psi_G$:

$$L = \Lambda^{-1}(\Lambda - A) \tag{15}$$

$$\Psi_G = \frac{1}{2} x^T L x \tag{16}$$

where $\Lambda$ is the square matrix containing the out-degree of each vertex along the diagonal, and $x$ is the concatenation of particles in the swarm. A well-known property of the Laplacian potential is that it is positive semi-definite and satisfies the following sum-of-squares property (Godsil and Royle, 2001):

$$x^T L x = \sum_{i,j \in E} A_{ij} \left( x^j - x^i \right)^2, \quad x \in \mathbb{R}^n \tag{17}$$

Using Eq. (17), the objective is to show that the personal best positions of each particle reaches a consensus (by way of equilibria) coincident to the global best of the swarm, i.e. $p_k^i = p_k^g$, $\forall \in P$. Eq. (16) becomes:

$$p^T L p = \sum_{i,j \in E} A_{ij} \left( p_k^j - p_k^i \right)^2, \quad p_k^i \in \mathbb{R}^n \tag{18}$$

where $p$ is the concatenated states of the personal best of each of the particles in the swarm. The closed-loop dynamics of the global best position evolve according to the following continuous-time dynamic equation:

$$\dot{p} = -Lp = -\nabla \Psi_G \tag{19}$$

The equilibrium points of Eq. (19) correspond to stationary points of $\Psi_G$ and the region outside of these points, the potential is strictly decreasing (Moreau, 2004); i.e., if $x^e$ is an equilibrium of Eq. (18), then $Lx^e = 0$. From Eq. (16):

$$\Psi_G(p^e) = \frac{1}{2} (p^e)^T L p^e = 0 \tag{20}$$

Following the connectivity of $G$, $p_i^e = p_j^e = c, \forall i, j \in N$, i.e. $p^e = (c, \ldots, c)^T, c \in X$. Since the Laplacian potential equals zero at equilibrium, then $p^g = \min(p)$ is an invariant quantity, Given the invariance property of $\min(p)$, then $\min(p^e) = \min(p(0))$, and $\min(p^e) = c$. This implies $p_k^{e,i} = \min(p(0))$, $\forall i \in P$ (Olfati-Saber and Murray, 2003). This leads to the following observations for the particle dynamics in Eq. (1a) and Eq. (1b) that are consistent with the works of (Clerc and Kennedy, 2002); (Trelea, 2003); and (Kadirkamanathan et al., 2006):

1. The system dynamics are stochastic and order two;
2. The system does not have an equilibrium point if $p_k^g \neq p_k^l$;
3. If $p_k^l = p_k^g = p^e$ is time invariant, there is a unique equilibrium at $v^e = 0$, $x^e = p^e$.

An equilibrium point thus exists only for the best particle whose local best solution is the same as the global best solution (Kadirkamanathan et al., 2006).
Consider the case for a given particle $i$ when the external input is constant (as is the case when no personal or global better positions are found). From Eq. (15) the eigenvalues of $-L$

are negative in the complex plane. Then, for particle $i$, the position asymptotically converges to the point $x^e$ in the eigenspace associated to the global minimum found by the swarm of particles (Olfati-Saber and Murray, 2003). Such a position $x^e$ is not necessarily a local or global minimiser of the combinatorial optimisation problem. Instead, it will improve towards the optimum $x^*$ if a better individual or global position is found. Discovery of better individual or global positions can be improved by increasing the population diversity of the swarm through the introduction of chaos or turbulence (Kennedy and Eberhart, 1995). In the following section, a non-stationary Markov chain is constructed to integrate the discrete syntactic swap operators introduced in Section 5 to the continuous time-dynamics of the traditional PSO

### 7.2 Non-Stationary Markov Model of combinatorial PSO

Markov chains are important in the theoretical analysis of evolutionary algorithms operating on discrete search spaces (Poli et al., 2007) and have been used to model the probabilistic convergence of population-based meta-heuristic algorithms (see (Rudolph, 1996); (Cao and Wu, 1997); (Poli and Langdon, 2007); and (Greenwood and Zhu, 2001) for examples of their implementation). While traditional PSO has operated on a continuous search space, the combinatorial PSO operates on a discrete combinatorial space. This makes Markov chains a suitable method of modelling and analysing the behaviour of the combinatorial PSO. The use of Markov chains on bare-bones PSO has previously been investigated by (Poli and Langdon, 2007) where the continuous search space was discretised using a hypercube sampling. In the following section, a non-stationary Markov chain is used to model the combinatorial PSO and account for the newly introduced swap operator.

Let $X$ denote the finite state space describing the set of permutation encodings with $r = |X| = n!$ possible solutions. Let $P \subset X$ be a population of solutions from $X$ with size $|P| = N$. Then a finite Markov chain $\Gamma \subseteq X$ describes a probabilistic trajectory over the finite state space $X$ (Rudolph, 1996) with $\mathbf{N} = \binom{m+n!-1}{n!-1}$ possible populations as states; i.e.:

$$X = \{S_1, S_2, \ldots, S_{\mathbf{N}}\} \tag{21}$$

The probability $q^{mn}_{k-1,k} := \Pr_{mn}(\Gamma^k = S_n | \Gamma^{k-1} = S_m)$ of transitioning from state $S_m \in X$ to $S_n \in X$, $m,n \in \mathbf{N}$ at step $k$ is called the *transition probability* from $m$ to $n$ at step $k$. The transition probability of a finite Markov chain can be gathered into a *transition matrix* $Q_k = \{q^{mn}_{k-1,k}\}$ (Rudolph, 1994), where each dimension $q^{mn}_{k-1,k} \in [0,1]$. In a *stationary Markov chain* the probabilities remain fixed, and the Markov chain is said to be *homogenous*; i.e., $Q = Q_k = \{q^{mn}_{k-1,k}\}$, $\forall k = 1,2,\ldots$, and $m,n = 1,2,\ldots,\mathbf{N}$. In the case of the combinatorial PSO, the probabilities of the swap operator are updated according to Eq. (8a) and Eq. (8b). This results in a *non-stationary Markov chain*. The transition probabilities of non-stationary Markov chains are calculated by considering how the population incidence vector $S_j$ describes the composition of the next iteration (Cao and Wu, 1997). Denote $z^{l,i}_k = \Pr(x^i_k | p^i_k) = v^{l,i}_k$ as the sampling probability when the personal best is $p^i_k$; likewise, denote $z_k^{g,i} = \Pr(x_k^i | p_k^g) = v_k^g$ as the sampling probability when the global best is $p_k^g$. The probability that a particle $i$ will move according to $x_k^i \mapsto p_k^i$ or $x_k^i \mapsto p_k^g$ $x_k^i$ is given by $z_k^i = (p_k^i \cup p_k^g)$. From Algorithm 3, the dimension for $z_k^{l,i} = \Pr(x_k^i | p_k^i)$ and $z_k^{g,i} = \Pr(x_k^i | p_k^g)$ is calculated independently using Eq. (8a) and Eq. (8b) and the probability of a particle sampling $p_k^i$ or $p_k^g$ is given by:

$$\Pr(p_k^i \cup p_k^g) = \prod_{d=1}^{D} \Pr\left(p_k^i(d) \cup p_k^g(d) \middle| p_k^i(d), p_k^g(d)\right) \tag{22a}$$

$$\Pr(p_k^i \cup p_k^g) = \prod_{d=1}^{D} \Pr(p_k^i(d)) + \Pr(p_k^g(d)) - \Pr(p_k^i(d)) \cdot \Pr(p_k^g(d)) \tag{22b}$$

Since personal bests can only change if there is a fitness improvement, only certain state transitions can occur. That is, a transition from state $S_m \mapsto S_n$ is possible only if the fitness of at least one particle in the swarm improves (Poli and Langdon, 2007). Because of the independence of the particles (over one time step), the state transition probability for the whole PSO is given by:

$$q_{k-1,k}^{mn} = \prod_i \Pr(p_k^i \cup p_k^g) \tag{23}$$

From Sec. 6.1, the local velocity $v_k^{l,i}$ and global velocity $v_k^{g,i}$ will tend to zero as $k \to +\infty$. This implies $\lim_{k\to\infty} q_{k-1,k}^{mn} = \lim_{k\to\infty} v_k^{l,i} = \lim_{k\to\infty} v_k^{g,i} = 0$, $m,n = 1,2,\ldots,\mathbf{N}$. Therefore, the swap operator preserves the convergent behaviour of traditional PSO and the combinatorial PSO converges to the equilibrium pair $(x^e, v^e)$.

## 8. Numerical Examples

### 8.1 The Travelling Salesman Problem

To test the efficiency of the proposed algorithm, the combinatorial optimisation PSO is tested on the travelling salesman problem (TSP). TSP is an invaluable test problem that belongs to the class of NP-hard combinatorial optimisation problems. The objective of TSP is to find a minimum-cost tour that visits a set of $n$ cities and returns to an initial point (Applegate et al., 2006). Mathematically, TSP is a combinatorial optimisation problem on an undirected graph $G = (V, E)$. Each city $c_i \in [n]$, $i = 1,2,\ldots,n$, is represented by a vertex $v_i \in V$ in the graph $G = (V, E)$ with cost of travel between adjacent cities given by $h_{ij} \in E$. A solution to TSP can be represented as a sequence of cities encoded by a permutation $x \in X$. Mathematically, the objective of TSP is given by the following optimisation problem:

$$x^* \in X \Leftrightarrow x^* = \arg\min_{x\in X} \sum_{d=1}^{n-1} h_{ij}(x(d), x(d+1)) + h_{ij}(x(n), x(1)) \tag{24}$$

Various problems, including path-finding, routing, and scheduling, can be modelled as a TSP. A repository of test-instances (and their solutions) is available through the TSPLIB library (Reinelt, 1991). In the following section, the combinatorial PSO is tested on several instances of the TPSLIB library. Table 1 summarises the test instances of TSPLIB used to validate and compare the combinatorial PSO.

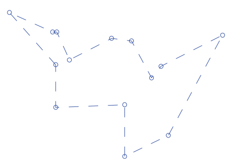
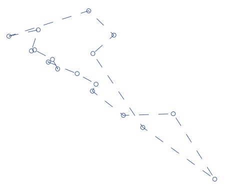
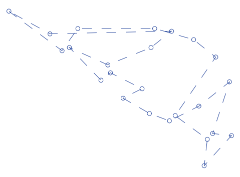
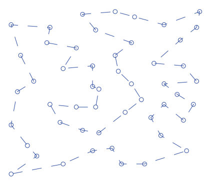
| Name | Dimension | Optimal $f(x)$ | Optimal Solution |
|---|---|---|---|
| burma | 14 | 30.8785 |  |
| gr17 | 17 | 2085 |  |
| gr24 | 24 | 1272 |  |
| eil51 | 51 | 426 |  |

Table 1. Test instances taken from TSPLIB (Reinelt, 1991) used for the validation of the combinatorial PSO

## 8.2 Optimisation Results and Discussion

In the following experiments, the combinatorial PSO is applied to each case of the TSP in Table 1. The parameters used in each experiment are selected based on the findings reported in the literature (Zhang et al., 2005); (Shi and Eberhart, 1998a); (Zheng et al., 2003); (Clerc and Kennedy, 2002); and (Eberhart and Shi, 2000). While the inertia weight, cognitive and social parameters are sensitive to the problem domain in traditional PSO, a parametric analysis of their influence on the combinatorial PSO is beyond the scope of this chapter. For illustrative purposes, the parameters given in Table 2 are considered throughout the remainder of this chapter. The influence of these parameters on the performance of the combinatorial PSO remains the subject of future research.

| Parameter | Value |
|---|---|
| $w$ | 0.8 |
| $c_1$ | 2.025 |
| $c_2$ | 2.025 |

Table 2. Combinatorial PSO parameters

To demonstrate the relative efficiency of the proposed algorithm, the performance of the combinatorial PSO is compared to a genetic algorithm. Each TSP experiment was trialled 100 times using randomly generated individuals. In both algorithms, a population of $P = 30$ was maintained for each iteration. The fitness values obtained by the combinatorial PSO and the GA over the 100 trials are presented in Table 3. Table 4 compares the success rate of the PSO and GA for each of the problems. Figure 3 compares the percentage of the solution space explored by the combinatorial PSO and the GA. This is determined as the number of unique solutions tested $x_k^i \in X$, $\forall i \in P$, $k = 1,\ldots,1000$ by the PSO and GA versus the size of the solution space $|X| = n!$.

| Problem | Optimal Solution | Minimum | | Maximum | | Average | |
|---|---|---|---|---|---|---|---|
| | | PSO | GA | PSO | GA | PSO | GA |
| burma | 30.87 | 30.87 | 30.87 | 30.87 | 34.62 | 30.87 | 31.20 |
| gr17 | 2085 | 2085 | 2085 | 2687 | 2489 | 2141.55 | 2175.02 |
| gr24 | 1272 | 1272 | 1282 | 1632 | 1810 | 1453.52 | 1488.68 |
| eil51 | 426 | 494.80 | 495.46 | 687.52 | 671.85 | 573.55 | 573.95 |

Table 3. Performance of the proposed algorithm compared to a traditional genetic algorithm for combinatorial optimisation

From Table 3, the combinatorial PSO outperformed the GA in all problem instances, except for the 51 variable eil51 problem. In this case, both the GA and combinatorial PSO failed to find the best solution over the 100 trials. Examination of Fig. 3 suggests that both the combinatorial PSO and GA were only able to search a small percentage ($<<1\%$) of the total solution space over the 1000 iterations. This suggests, that both the combinatorial PSO and GA experience a loss of solution diversity over the optimisation procedure. Figure 3 also indicates that the GA was able to cover a larger percentage of the solution space for each trial than the combinatorial PSO. This suggests that the combinatorial PSO suffers from the same rapid convergence and stagnation issues of traditional PSO. Loss of solution diversity and rapid convergence is a well-known problem in traditional PSO. In traditional PSO, the performance of the algorithm deteriorates as the number of iterations increases. Once the algorithm has slowed down (becomes stagnant), it is usually difficult to achieve a better fitness value; particularly for high-dimensionality problem spaces.

| Problem | Success Rate (%) | |
|---|---|---|
| | PSO | GA |
| burma | 100 | 92 |
| gr17 | 36 | 17 |
| gr24 | 4 | 0 |
| eil51 | 0 | 0 |

Table 4. Success rate of the combinatorial PSO and GA

Recently, several methods have been proposed to improve solution diversity and avoid stagnation in traditional PSO. These methods include the use of chaos variables (Fieldsend and Singh, 2002); (Kennedy and Eberhart, 1995); and (He et al., 2004); variable neighbourhood topologies (Kennedy, 1999); and (Liu et al., 2007a); and mutation operators (Liu et al., 2007b); and (Andrews, 2006). Many of these techniques have had varying levels of success on the traditional PSO algorithm. It is expected, that these same strategies can be adapted to the combinatorial PSO. Future work aims to investigate the potential to implement these algorithmic improvements to the combinatorial PSO and solve for larger scale combinatorial optimisation problems.

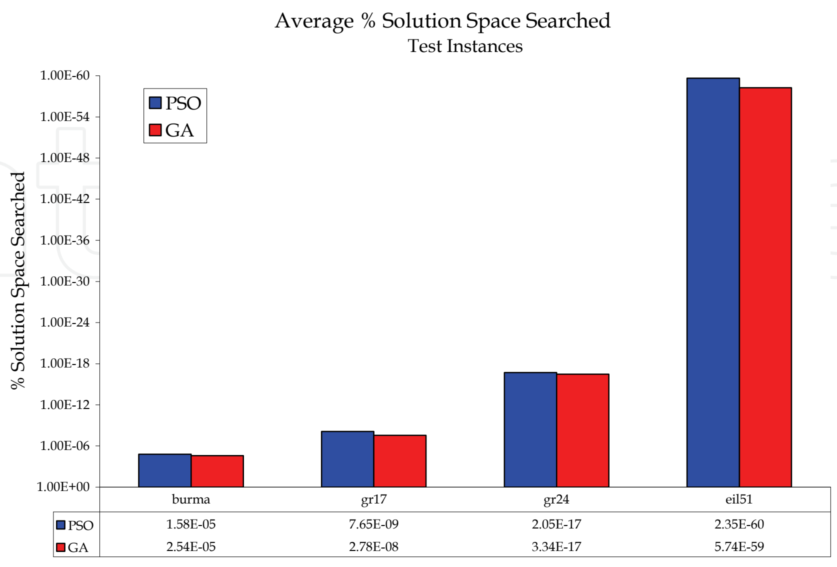


Figure 3. Comparison of the solution space searched by the combinatorial PSO and the GA

## 9. Conclusion

The PSO's simplicity, robustness, and low computational costs, makes it an ideal method for continuous optimisation problems. Previous efforts to adapt the traditional PSO algorithm to combinatorial spaces have shown varying levels of success. In this chapter, a new combinatorial optimisation PSO that builds on previous works is introduced. A distance metric was introduced to define a metric space for the combinatorial optimisation problem and a syntactic swap operator introduced. Motion was induced by associating a probability sampling function to the velocity profile of a particle on the combinatorial space and invoking the defined swap operator. The proposed algorithm was tested on several instances of TSPLIB and compared to the performance of a GA. Preliminary test results demonstrated superior performance over the GA in all test cases. For larger set sizes, the proposed algorithm failed to converge to the optimal solution. Examination of the sampled solution space suggested that the proposed algorithm suffered from the same rapid convergence and stagnation issues observed in traditional PSO. Further research is needed to clarify the effect of the various tuning parameters on the performance of the proposed algorithm, and their influence on loss of solution diversity. The generalised approach to the algorithm's development allows for the consideration of other metrics on discrete spaces, and the implementation of further algorithmic improvements. Future work aims to investigate methods to mitigate the stagnation issues of the proposed algorithm and extending the combinatorial optimisation PSO's capabilities to other discrete optimisation problems.

## 10. References

Aardal, K., Hoesel, S. v., Lenstra, J. K. and Stougie, L. (1997). *A Decade of Combinatorial Optimization*. Department of Information and Computing Sciences, Utrecht University, UU-CS-1997-12,

Aguilar, J. (2001). A General Ant Colony Model to solve Combinatorial Optimization Problems. *Revista Colombiana De Computación*, 2, 7 - 18.

Andrews, P. S. (2006). An Investigation into Mutation Operators for Particle Swarm Optimization, *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1044 - 1051, Vancouver BC, Canada, July,

Applegate, D. L., Bixby, R. E., Chvátal, V. e. and Cook, W. J. (2006). *The Traveling Salesman Problem,* Princeton University Press, New Jersey.

Bean, J. (1994). Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing,* 6, 154 - 160.

Bóna, M. (2004). *Combinatorics of Permutations,* Chapman & Hall/CRC, New York.

Bonyadi, M. R., Azghadi, S. M. R. and Hosseini, H. S. (2007). Solving Traveling Salesman Problem Using Combinational Evolutionary Algorithm, In *Artificial Intelligence and Innovations 2007: From Theory to Applications*, Vol. 247 (Eds, Boukis, C., Pnevmatikakis, L. and Polymenakos, L.) Springer, Boston, pp. 37 - 44.

Cao, Y. J. and Wu, Q. H. (1997). Convergence Analysis of Adaptive Genetic Algorithms, *Proceedings of Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 85 - 89, 2 - 4 September, Clerc, M. and Kennedy, J. (2002). The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation,* 6, 58 - 73.

Eberhart, R. C. and Shi, Y. (2000). Comparing Inertia Weights and Constriction Factors in Particle Swarn Optimization, *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 84-88, San Diego, CA,

Festa, P. and Resende, M. G. C. (2008). *Hybrid Grasp Heuristics*. AT&T Labs Research, Florham Park, July

Fieldsend, J. E. and Singh, S. (2002). A Multi-Objective Algorithm Based Upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence, *Proceedings of UK Workshop on Computational Intelligence*, pp. 37 - 44, UK,

Godsil, C. and Royle, G. (2001). *Algebraic Graph Theory,* Springer-Verlag, New York.

Goldberg, D. and Lingle, J. R. (1985). Alleles, Loci and the TSP, *Proceedings of First International Conference on Genetic Algorithms and Their Applications*, pp. 154 - 159, Hillsdale, Hew Jersey,

Greenwood, G. W. and Zhu, Q. J. (2001). Convergence in Evolutionary Programs with Self-Adaptation. *Evolutionary Computation,* 2, 147 - 157.

He, S., Wu, Q. H., Wen, J. Y., Saunders, J. R. and Paton, R. C. (2004). A Particle Swarm Optimizer with Passive Congregation. *Biosystems,* 78, 135 - 147.

Kadirkamanathan, V., Selvarajah, K. and Fleming, P. J. (2006). Stability Analysis of the Particle Dynamics in Particle Swarm Optimizer. *IEEE Transactions on Evolutionary Computation,* 10, 245 - 255.

Kennedy, J. (1999). Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance, *Proceedings of Congress on Evolutionary Computation*, pp. 1931 - 1938, Washington DC, USA,

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942-1948, 27th November - 1 December, 1995,

Kennedy, J. and Eberhart, R. C. (1997). A Discrete Binary Version of the Particle Swarm Algorithm, *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp.

Knuth, D. E. (1998). *The Art of Computer Programming,* Addison-Wesley, Reading.

Liu, H., Abraham, A. and Grosan, C. (2007a). A Novel Variable Neighborhood Particle Swarm Optimization for Multi-Objective Flexible Job-Shop Scheduling Problems, *Proceedings of 2nd International Conference on Digital Information Management*, pp. 138 - 145, October,

Liu, J., Fan, X. and Qu, Z. (2007b). An Improved Particle Swarm Optimization with Mutation Based on Similarity, *Proceedings of Third International Conference on Natural Computation*, pp. Haikou, China,

Martínez García, F. J. and Moreno Pérez, J. A. (2008). *Jumping Frogs Optimization: A New Swarm Method for Discrete Optimization*. Department of Statistics, O. R. and Computing, University of La Laguna, Tenerife, Spain, DEIOC 3/2008,

Misevicius, A., Blažauskas, T., Blonskis, J. and Smolinskas, J. (2004). An Overview of Some Heuristic Algorithms for Combinatorial Optimization Problems. *Information Technology and Control,* 30, 21 - 31.

Moraglio, A. and Poli, R. (2004). Topological Interpretation of Crossover, In *Genetic and Evolutionary Computation - GECCO 2004*, Vol. 3102/2004 Springer Berlin/Heidelberg, Berlin/Heidelberg, pp. 1377 - 1388.

Moreau, L. (2004). Stability of Continuous-Time Distributed Consensus Algorithms, *Proceedings of 43rd IEEE Conference on Decision and Control*, pp. 3998 - 4003, Atlantis, Paradise Island, Bahamas, December,

Olfati-Saber, R. and Murray, R. M. (2003). Consensus Protocols for Networks of Dynamic Agents, *Proceedings of American Control Conference*, pp. 951-956,

Pang, W., Wang, K.-p., Zhou, C.-g. and Dong, L.-j. (2004). Fuzzy Discrete Particle Swarm Optimization for Solving Traveling Salesman Problem, *Proceedings of 4th International Conference on Computer and Information Technology (CIT04), IEEE Computer Society*, pp.

Passino, K. M. (2002). Biomimicry of Bacterial Foraging for Distributed Optimization and Control. *IEEE Control Systems Magazine,* 22, 52 - 67.

Poli, R. and Langdon, W. B. (2007). Markov Chain Models of Bare-Bones Particle Swarm Optimizers, *Proceedings of 9th Annual Conference on Genetic and Evolutionary Computation*, pp. 142 - 149, London, England, 7 - 11 July,

Poli, R., Langdon, W. B., Clerc, M. and Stephens, C. R. (2007). Continuous Optimisation Theory Made Easy? Finite-Element Models of Evolutionary Strategies, Genetic Algorithms and Particles Swarm Optimizers, In *Foundations of Genetic Algorithms*, Vol. 4436/2007 Springer Berlin/Heidelberg, Berlin, pp. 165 - 193.

Reinelt, G. (1991). TSPLIB - Traveling Salesman Problem Library. *ORSA Journal on Computing,* 3, 376 - 384.

Ronald, S. (1997). Distance Functions for Order-Based Encodings, *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 49 - 54, Indianapolis, USA,

Ronald, S. (1998). More Distance Functions for Order-Based Encodings, *Proceedings of IEEE Conference on Evolutionary Computation*, pp. 558 - 563, IEEE Press

Rudolph, G. (1994). Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions on Neural Networks,* 5, 96 - 101.

Rudolph, G. (1996). Convergence of Evolutionary Algorithms in General Search Spaces, *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 50 - 54, Nagoya, Japan, 20 - 22 May,

Shi, X. H., Zhou, Y., Wang, L. M., Wang, Q. X. and Liang, Y. C. (2006). A Discrete Particle Swarm Optimization Algorithm for Travelling Salesman Problem, In *Computational Methods*(Eds, Liu, G. R., Tan, V. B. C. and Han, X.) Springer Netherlands, Netherlands, pp. 1063 - 1068.

Shi, Y. and Eberhart, R. C. (1998a). A Modified Particle Swarm Optimiser, *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. Anchorage, Alaska, May,

Shi, Y. and Eberhart, R. C. (1998b). Parameter Selection in Particle Swarm Optimization, *Proceedings of 7th International Conference on Evolutionary Programming*, pp. 591 - 600,

Song, X., Chang, C. and Cao, Y. (2008). New Particle Swarm Algorithm for Job Shop Scheduling Problems, *Proceedings of 7th World Congress on Intelligent Control and Automation*, pp. 3996 - 4001, Chongqing, China, 25 - 27 June, Tasgetiren, M. F., Sevkli, M., Liang, Y. C. and Gencyilmaz, G. (2004). Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem, In *Ant Colony, Optimization and Swarm Intelligence*, Vol. 3172/2004 Springer Berlin/Heidelberg, Berlin, pp. 382 - 389.

Trelea, I. C. (2003). The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. *Information Processing Letters,* 85, 317 - 325.

Wang, K.-p., Huang, L., Zhou, C. G. and Pang, W. (2003). Particle Swarm Optimization for Traveling Salesman Problem, *Proceedings of Second International Conference on Machine Learning and Cybernetics*, pp. 1583 - 1585, Xi'an, China, November,

Zhang, L.-p., Yu, H.-j. and Hu, S.-X. (2005). Optimal Choice of Parameters for Particle Swarm Optimization. *Journal of Zhejiang University Science,* 6, 528-534.

Zheng, Y.-L., Ma, L.-H., Zhang, L.-Y. and Qian, J.-X. (2003). On the Convergence Analysis and Parameter Selection in Particle Swarm Optimization, *Proceedings of Second International Conference on Machine Learning and Cybernetics*, pp. 1802 - 1807, Xi'an, China, November,