**Swarm Intelligence manuscript No.**
(will be inserted by the editor)

# Set-Based Particle Swarm Optimization applied to the Multidimensional Knapsack Problem

**Joost Langeveld** · **Andries P. Engelbrecht**

Received: date / Accepted: date

**Abstract** Particle swarm optimization algorithms have been successfully applied to discrete-valued optimization problems. However, in many cases the algorithms have been tailored specifically for the problem at hand. This paper proposes a generic set-based particle swarm optimization algorithm for use on discrete-valued optimization problems that can be formulated as set-based problems. A detailed sensitivity analysis of the parameters of the algorithm is conducted. The performance of the proposed algorithm is then compared against three other discrete particle swarm optimization algorithms from literature using the multidimensional knapsack problem, and is shown to statistically outperform the existing algorithms.

**Keywords** Discrete Optimization · Multidimensional Knapsack Problem · Particle Swarm Optimization · Set-Based Optimization Problem

## 1 Introduction

Particle Swarm Optimization (PSO) has established itself as a valuable tool in the field of continuous optimization. Proposed by Kennedy and Eberhart (1995), it was inspired by the movement of flocking birds. In order to solve discrete-valued optimization problems (DOP), a number of variations of PSO have been proposed, starting with the binary PSO algorithm by Kennedy and Eberhart (1997). Since then, a variety of different discrete PSO methods have been developed. Typical applications of discrete PSOs are problems that involve ordering (Wang et al., 2003; Clerc, 2004), scheduling (Abraham et al., 2006; Tasgetiren et al., 2004), or feature selection (Tu et al., 2008). Many such problems are combinatorial, which gives the problems additional structure. This structure has been used to develop problem specific optimization methods (Li et al., 2008).

Joost Langeveld
Department of Computer Science, University of Pretoria
E-mail: jclangev@gmail.com

Andries P. Engelbrecht
Department of Computer Science, University of Pretoria
Tel.: +27 12 420 3578
E-mail: engel@cs.up.ac.za

This paper introduces a new generic set-based PSO algorithm called Set-Based PSO (SBPSO) and compares its performance in solving discrete-valued optimization problems, specifically set-based problems, to existing PSO algorithms. The term generic means that no problem specific information is used in the algorithm other than in the objective function. This allows the algorithm to be seemlessly applied without alteration to any DOP that allows for a set-based representation of the solution. The set-based approach is chosen as an alternative to the more traditional binary string implementations of discrete PSO and the permutation implementation often used for combinatorial optimization problems. Thus a particle position is defined as a set of elements. This has the important implication that the size of the particle position can change as the algorithm executes, and also that the positions of the particles in the swarm will, in general, have different sizes.

The multidimensional knapsack problem (MKP) is chosen as the test problem because it can be formulated as a set-based optimization problem and it allows for straight-forward objective function evaluation of particles. Thus the SBPSO can be evaluated, and compared to alternative PSO algorithms, based only on the quality of the solutions determined by the PSO algorithm and not aided by domain specific operators. It is acknowledged that problem specific algorithms can yield better solutions, but the scope of this paper is to find an efficient generic set-based PSO algorithm to apply to DOPs exemplified by the MKP.

The remainder of this paper is structured as follows: first a brief overview of the continuous PSO algorithm is given. Then a review of existing discrete PSO algorithms and existing set-based PSO algorithms is provided. Section 3 describes the SBPSO algorithm. Section 4 formally defines the MKP, and existing studies that use swarm intelligence to solve the MKP are highlighted. Section 5 explains the experimental procedure conducted, and describes how the control parameters of the individual PSO algorithms were tuned. Section 6 uses the results of the parameter tuning process to conduct a sensitivity analysis of the SBPSO control parameters. Section 7 lists the results of applying the tuned PSO algorithms to the MKP test problems, followed by conclusions and an indication of future work in section 8.

## 2 Particle Swarm Optimization

This section gives a brief overview of the continuous PSO algorithm, and describes three swarm topologies used in PSO. This is followed by a review of existing discrete and set-based PSO algorithms.

### 2.1 Continuous Particle Swarm Optimization

Kennedy and Eberhart (1995) were the first to propose an optimization algorithm inspired by bird flocking behavior. The first PSO algorithm was developed to solve optimization problems with continuous-valued parameters. Each particle has a position $\mathbf{x}$ in the search space, and a velocity $\mathbf{v}$ indicating direction and step-size of change in current position. Each particle keeps track of the quality of the solution to the optimization problem it represents, the best position it has visited in the past, $\mathbf{y}$, and the best position visited in the past by a particle in its neighborhood, denoted $\hat{\mathbf{y}}$.

Let $i$ be a particle in an $n$-dimensional search space with velocity $\mathbf{v}_i = (v_i)_{j=1}^n$, position $\mathbf{x}_i = (x_i)_{j=1}^n$, personal best position $\mathbf{y}_i = (y_i)_{j=1}^n$, and neighborhood best position

$\hat{\mathbf{y}}_i = (\hat{y}_i)_{j=1}^n$. The original velocity update equation,

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)\,[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)\,[\hat{y}_{i,j}(t) - x_{i,j}(t)] \quad (1)$$

computes the magnitude of change in the particle's position in each dimension $j$, where $c_1$ is the cognitive component weight, $c_2$ is the social component weight, and $\mathbf{r}_1$ and $\mathbf{r}_2$ are $n$-dimensional random vectors with each $r_{1,j}, r_{2,j} \sim U(0,1)$ drawn independently. The position is updated by adding the updated velocity to the current position:

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (2)$$

To improve the performance of the algorithm and to better control the balance between exploration of new areas of the search space and exploitation of promising areas, various additions have been proposed. A first addition was by Eberhart et al. (1996), who proposed *velocity clamping* which restricts the velocity to a predetermined maximum in each dimension. After the velocity has been updated, but before the position update, the velocity clamping,

$$v_{i,j}(t+1) = \min\{\max\{v_{i,j}(t+1), V_{\min,j}\}, V_{\max,j}\} \quad (3)$$

is applied, where $V_{\min,j}$ and $V_{\max,j}$ with $V_{\min,j} < V_{\max,j}$ denote the minimum and maximum velocity in a single dimension $j$.

An addition proposed by Shi and Eberhart (1998) was a scalar, $\omega$, called the *inertia weight*, which determines the acceleration or deceleration in the current direction. The inertia weight scales the component indicating the particle's current velocity, $v_{i,j}(t)$, in equation (1), resulting in an alternative velocity update equation,

$$v_{i,j}(t+1) = \omega\, v_{i,j}(t) + c_1 r_{1,j}(t)\,[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)\,[\hat{y}_{i,j}(t) - x_{i,j}(t)] \quad (4)$$

Algorithm 1 describes the flow of the PSO algorithm for a maximization problem with objective function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. A similar definition is easily obtained for a minimization problem.

### 2.2 Swarm Topologies

One of the strengths of PSO is the flow of information through the swarm due to the interaction of the particles. Particles with a good objective function value attract other particles, hopefully to good areas of the search space. Particles that have found a good solution attract particles for which they are the best neighbor. If two particles $i$ and $j$ are not connected (not in each other's neighborhood), then they can not directly attract each other. If a common neighbor $k$ is attracted to a good solution $i$ and becomes a good solution itself, such that it is the best solution in the neighborhood of $j$, then $j$ can be said to be indirectly influenced by $i$. For each particle, the social structure, called the swarm topology, determines which particles it can be attracted to.

Kennedy and Eberhart (1995) proposed two possible social structures for the particle neighborhoods, and called the two resulting algorithms the global best (*gbest*) PSO and local best (*lbest*) PSO. The gbest PSO uses a *star* topology, while the lbest PSO uses a *ring* topology. The ring topology is a loosely connected topology, while the star topology is one where each particle is directly connected to all other particles in the swarm. A study of the impact of the swarm topology was done by Kennedy and Mendes (2002), considering various topologies, including random, star, Von Neumann and ring topologies. Kennedy and Mendes (2002) suggested that the *Von Neumann* topology, which has an intermediate level of connectivity, can be a good choice for a particle swarm.

**Algorithm 1:** Continuous PSO for Maximization Problems

```
Set N equal to the number of particles in the swarm;
for i = 1,...,N do
    Initialize x_i uniformly random over the search space ;
    Initialize v_i = 0 ;
    Calculate f(x_i) ;
    Initialize f(y_i) = −∞ ;
    Initialize f(ŷ_i) = −∞ ;
end
while stopping condition is false do
    for i = 1,...,N do
        // set the personal best position ;
        if f(x_i) > f(y_i) then
            y_i = x_i;
        end
        // set the neighborhood best position ;
        for all neighbors l of particle i do
            if (f(y_i) > f(ŷ_l) then
                ŷ_l = y_i;
            end
        end
    end
    for i = 1,...,N do
        Update v_i according to equation (4);
        Update x_i according to equation (2);
        Calculate solution quality f(x_i);
    end
end
```

### 2.3 Discrete Particle Swarm Optimization

This section reviews PSO algorithms developed to solve DOPs, namely the binary PSO, the modified binary PSO, the probability binary PSO, the angle modulated PSO, fuzzy and rank-based binary PSO algorithms, and PSO algorithms that redefine the meaning of particle positions, velocities, and arithmetic operators.

*Binary PSO:* Kennedy and Eberhart (1997) were the first to define a discrete version of the PSO algorithm, referred to as the binary PSO (BPSO). In this algorithm the particle positions are binary strings, while the velocities exist in continuous space. Velocities are mapped to a scalar value between 0 and 1 using a sigmoidal transformation function, $S$. This scalar value is interpreted as the probability that the corresponding part of the binary position string is bit 1. The velocity update equation of the BPSO algorithm is the same as equation (4). Using the transformation function,

$$S\big(v_{i,j}(t+1)\big) = \frac{1}{1+e^{-v_{i,j}(t+1)}} \tag{5}$$

the position update becomes

$$x_{i,j}(t+1) = \begin{cases} 1 & \text{if } r_{3,j} < S\big(v_{i,j}(t+1)\big) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $r_{3,j}$ is an independent random variable, uniformly distributed on $(0,1)$. Eberhart et al. (2001) proposed to use velocity clamping as defined in equation (3) in BPSO to prevent saturation of the sigmoid function.

Many variants of the BPSO algorithm have been proposed: Khanesar et al. (2007) defined a BPSO that has separate velocity terms depending on whether a bit in the current position vector $\mathbf{x}$ is 0 or 1, Gao et al. (2006) removed the randomness from the position update step, and Yang et al. (2004) proposed the quantum BPSO by introducing the idea of a superposition of states.

*Modified Binary PSO:* Shen et al. (2004) proposed the modified binary PSO (MBPSO) to select variables in multiple linear regression and partial least-squares modeling. The velocity update equation of MBPSO is the same as equation (4). For the position update, each bit $x_{i,j}(t)$ in the position vector $\mathbf{x}_i(t)$ is updated according to:

$$x_{i,j}(t+1) = \begin{cases} x_{i,j}(t) & \text{if } 0 \leq v_{i,j}(t+1) \leq p_{\text{stat}} \\ y_{i,j}(t) & \text{if } p_{\text{stat}} < v_{i,j}(t+1) \leq 0.5(1+p_{\text{stat}}) \\ \hat{y}_{i,j}(t) & \text{if } 0.5(1+p_{\text{stat}}) < v_{i,j}(t+1) \leq 1 \end{cases} \tag{7}$$

where $p_{\text{stat}}$ is a parameter in $(0,1)$ called the *static probability*.

Shen et al. (2004) stated that after the velocity and position updates have been applied, a fraction of particles "are forced to fly randomly not following the two best particles". This statement has been interpreted as a random re-initialization of both the velocity and the position of a percentage of the swarm at each iteration, similar to Ma et al. (2010). The fraction of particles that is re-initialized at each iteration is denoted by $p_{\text{reset}}$.

*Probability Binary PSO:* Wang et al. (2008) proposed a variant of BPSO called the probability binary PSO (PBPSO) and applied this to the MKP. The velocity update equation of PBPSO is the same as for continuous PSO given in equation (2). A continuous-valued position, $\mathbf{x}'$, is introduced, which is updated according to equation (4). A linear transformation,

$$L\left(x'_{i,j}(t+1)\right) = \frac{x'_{i,j}(t+1) - R_{\min}}{R_{\max} - R_{\min}} \tag{8}$$

is used to transform the continuous-valued position into a binary-valued position, $\mathbf{x}$, using

$$x_{i,j}(t+1) = \begin{cases} 1 & \text{if } r_{i,j} < L\left(x'_{i,j}(t+1)\right) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where each $r_{i,j}$ is an independent random variable, uniformly distributed on $(0,1)$. The parameters $R_{\min}$ and $R_{\max}$ used in the linear transformation are usually chosen such that $R_{\max} > 0$ and $R_{\min} = -R_{\max}$. (Menhas et al., 2011) extended the PBPSO algorithm to also include a mutation operator. After application of the linear transformation in equation (8), each bit was given a probability $p_{\text{mut}} \in [0,1]$ of mutating, resulting in the position update,

$$x_{i,j}(t+1) = \begin{cases} 1 - x_{i,j}(t+1) & \text{if } r_{i,j} < p_{\text{mut}} \\ x_{i,j}(t+1) & \text{otherwise} \end{cases} \tag{10}$$

where each $r_{i,j}$ is an independent random variable, uniformly distributed on $(0,1)$.

*Angle modulation:* Pampara et al. (2005) developed a different approach to converting the continuous-valued velocity of PSO to a binary string by applying the concept of *angle modulation*. Angle modulation PSO starts with a swarm of particles in a continuous four dimensional space, and uses a continuous PSO algorithm to update the particle velocities and positions. For each particle, the four position components are used as parameters for a trigonometric function, and this function is sampled $n$ times to generate an $n$-dimensional bit-string. If the function produces a positive value, then bit 1 is recorded, otherwise bit 0 is recorded.

*Fuzzy binary PSO approaches:* Fuzzy logic has also been used to construct discrete PSO algorithms. Where the particle position in binary PSO is a binary vector with a "crisp" separation of bits into 0 and 1, fuzzy binary PSO instead has a position vector with fuzzy bits. It uses a membership function $\mu$ to indicate a truth value in $[0,1]$ for the degree to which each fuzzy bit has value 1. The fuzzy PSO algorithm works in continuous space and a separate mechanism called *defuzzification* is used to convert the fuzzy particle position into a binary vector. The first published article on using a fuzzy approach to the discrete PSO is by (Shi and Eberhart, 2001). Pang et al. (2004a) and Shen et al. (2006) provided refinements to the fuzzy method and applied it to the traveling salesman problem (TSP). Du et al. (2005) applied their fuzzy PSO to the shape matching problem, while Abraham et al. (2006); Liu et al. (2010); Liu and Abraham (2007) applied fuzzy discrete PSO algorithms to job scheduling problems and to the quadratic assignment problem.

*Rank ordering approaches:* A different approach is where discrete PSO algorithms use the concept of rank ordering to transform a continuous-valued position to a discrete-valued position. Tasgetiren et al. (2004) introduced such a modification to the continuous PSO algorithm and applied it to scheduling problems, exemplified by the single machine total weighted tardiness problem. Solutions for such scheduling problems are sequences or permutations of tasks that indicate the order in which the tasks are performed. A candidate solution is represented as a sequence $S_i = [s_{i,1}, \ldots, s_{i,n}]$ of the numbers $1, \ldots, n$, where each $s_{i,k}$ is unique and denotes one of the $n$ tasks to be scheduled.

The particle velocities and positions are updated according to equations (4) and (2) respectively. Each position, $\mathbf{x}_i$, is then translated to a sequence $S_i$ using the *smallest position value* (SPV) rule. The SPV rule takes the position component, $x_{i,j}$, with the smallest value in $\mathbf{x}_i$, and sets $s_{i,1}$ equal to $j$. Then it takes the next smallest position component, $x_{i,k}$, and sets $s_{i,2} = k$. This process continues until the sequence $S_i$ has been filled.

Similar algorithms have been proposed by Pang et al. (2004b), who used the *greater value priority* to transform the continuous-valued position $\mathbf{x}_i$ to a sequence $S_i$ and applied the resulting PSO algorithm to the TSP. Liu et al. (2007) used an almost identical approach called *rank order value* and applied this method to the flow shop scheduling problem (FSSP).

*Redefined PSO operators:* Clerc (2004) formulated a discrete PSO algorithm by redefining the particles, velocities and operators used in PSO. A general mathematical specification is given as well as an implementation that is then applied to the TSP. A particle position is defined as a sequence of $N+1$ arcs between nodes, where $N$ is the number of nodes in the TSP. A velocity is defined as a list of *exchange operations* $(i, j)$, where nodes $i$ and $j$ in a position are swapped. Special operations are also defined for subtraction of two positions, the addition of two velocities, and the multiplication of a scalar and a velocity. These new operators are then used in a formulation of the velocity update equation in the discrete PSO that is very similar to equation (4) used in continuous PSO.

Wang et al. (2003), Zhang et al. (2007), and Zhong et al. (2007) proposed similar approaches to modifying the PSO operators and each applied the resulting PSO to the TSP. García et al. (2006) applied an adapted PSO algorithm to the response time variability problem, where the particle velocity is defined as an ordered list of transformations called *movements*. Benameur et al. (2009) proposed a similar discrete PSO and applied it to the frequency assignment problem. Chandrasekaran et al. (2006) applied a discrete PSO with redefined operators to the FSSP, where the velocity is a set of transpositions with ordering values. The transpositions contained in the velocity are applied to the position in the order of high to low ordering values.

## 2.4 Particle Swarm Optimization using Sets

In literature, a number of PSO algorithms that use mathematical sets already exist. It is the opinion of this paper, however, that these existing methods are not truly set-based or not always generically applicable to all set-based optimization problems.

The algorithm proposed by Correa et al. (2006) for attribute selection and the related algorithm by Bock and Hettenhausen (2012) for ontology alignment both have set-like characteristics, but both contain problem specific elements. Especially, the concept of a *personal likelihood* that requires each element in a particle position to have its own partial objective function value, prevents these algorithms from being applied to many discrete optimization problems, including the MKP.

Veenhuis (2008) proposed a generic, set-based definition of a PSO algorithm. Velocities and positions in this algorithm are both defined as sets. However, the chosen update equations lead the velocities and positions to always increase in size, an effect called *set bloating*. To counter this, a reduction operator with a relatively complex clustering mechanism was introduced. This clustering mechanism requires a function that defines the distance between any two set elements, while a general mathematical set does not support the concept of distance. Veenhuis (2008) has therefore chosen a problem specific distance function. This means that the algorithm is no longer truly generic, and in its current form is not applicable to discrete problems such as the MKP.

Neethling and Engelbrecht (2006) proposed the set-based algorithm called SetPSO and applied it to RNA structure prediction. The problem is defined as finding the correct stems (bindings of base pairs) in the RNA structure from the set of all possible stems. Particle positions are defined as sets of stems. In the position update, three probabilities help determine which elements are added and which elements are removed from the position. Although generically applicable, recent work (Langeveld and Engelbrecht, 2011) has shown that SetPSO performs less well on the MKP than other PSO methods.

Chen et al. (2010) proposed a generic set-based PSO method called S-PSO that can be used to adjust a continuous PSO algorithm to a discrete one. S-PSO was applied to the TSP and the MKP. The candidate solution represented by a particle position is called a set, but has a fixed size, where for each "dimension" of the set an element is chosen from a set of available elements. Thus the position can not be called a true set. Velocity is defined as a *set with possibilities*, which grows in size as the algorithm runs. Positions are rebuilt at each iteration using a constructive process that may include heuristic operators. Wu et al. (2010) applied a variant of S-PSO based on (continuous) constriction PSO to the problem of cloud computing workflow scheduling.

Khan and Engelbrecht (2010) proposed an algorithm called fuzzy PSO (FPSO) to optimize the topology design of distributed local area networks (DLANs). The term fuzzy in FPSO refers to the fuzzy aggregation operator, the *unified And-Or operator*, that is used to aggregate the multiple objectives in the DLAN topology design problem into a single objective function. The particle position is defined as a set of links between nodes in the network. The number of links in the position is exactly $N-1$, where $N$ is the number of nodes in the network. The particle velocity is defined as a set of *link exchange operations*, which remove a single link in the position and replace it by another. Because the size of the position is fixed, the algorithm is not generally applicable to discrete problems such as the MKP.

## 3 Set-Based Particle Swarm Optimization

This section describes in detail the SBPSO first mentioned in (Langeveld and Engelbrecht, 2011), which is revised and investigated in much more detail in this paper. SBPSO can be applied to any DOP which can be defined as set-based optimization problems. Section 3.1 defines the SBPSO set-based concepts, while sections 3.2 and 3.3 respectively redefine the arithmetic operators and PSO update equations to operate on sets.

### 3.1 Set-Based Concepts

SBPSO defines a particle's position and velocity as mathematical sets. The position is a set of elements from the universe of discourse $U$, that is, the universe of elements defined by the problem. The velocity is a set of *operation pairs* defined below. The solution that SBPSO finds for the optimization problem is thus the best position found by the swarm, represented as a set of elements from $U$.

The definitions below assume that SBPSO is applied to a maximization task, but a similar definition for a minimization task is easily derived from this. Let

- $U = \{e_n\}_{n \in N_U}$ be the universe of discourse containing all elements, $e_n$, of which there are a finite number $N_U$,
- $X_i(t)$ be the position of particle $i$ at iteration $t$, a subset of $U$,
- $V_i(t)$ be the velocity of particle $i$ at iteration $t$,
- $f$ be the objective function to be optimized,
- $Y_i(t)$ be the personal best position of particle $i$, that is, $Y_i(t) = X_i(\tau)$, $\tau \in \{1,\ldots,t\}$, such that $f\big(Y_i(t)\big) = f\big(X_i(\tau)\big) = \max\{f(X_i(s))\big|s = 1,\ldots,t\}$,
- $\widehat{Y}_i(t)$ be the neighborhood best position for particle $i$ at iteration $t$, that is, $\widehat{Y}_i(t) = Y_j(t)$ for the particle $j$ in $i$'s neighborhood that maximizes $f\big(Y_j(t)\big)$.

Figure 1(a) shows a particle position $X(t)$ as a set in the universe $U$. This universe and mathematical sets in general do not have a spatial structure, so the placements of the elements denoted with small squares is arbitrary and no elements can be said to be close to or far away from each other.

The PSO paradigm is built on the idea of movement through the search space, using the concept of velocity. For SBPSO this idea of movement needs to be defined. In continuous PSO, attraction of a particle to its personal best position partly determines the particle's velocity. In SBPSO the same attraction to the personal best applies. Figure 1(b) shows a particle position $X(t)$ and personal best position $Y(t)$. Here $X(t)$ and $Y(t)$ are shown to partially overlap, though this is not necessarily true. The *movement* of $X(t)$ *towards* $Y(t)$ in SBPSO means that the two sets are made more similar by removing elements from $X(t)$ that are not in $Y(t)$ (pictured as $\diamond$), and by adding to $X(t)$ missing elements that are in $Y(t)$ (pictured as *). Elements that are in both $X(t)$ and $Y(t)$ are not affected by this attraction, nor are elements that lie outside both $X(t)$ and $Y(t)$.

The velocity is defined as a set of *operation pairs*, where an operation pair is the addition *or* deletion of a single element. An operation pair is denoted as $(\pm, e)$, with $(+, e)$ for the addition of element $e \in U$ and $(-, e)$ for the deletion of element $e$. The velocity of particle $i$, $V_i(t)$, is then written as $\{v_{i,1},\ldots,v_{i,k}\} = \{(\pm, e_{n_{i,1}}),\ldots,(\pm, e_{n_{i,k}})\}$, where $k$ is the number of operation pairs in $V_i(t)$, and each $e_{n_{i,j}}$ is an element in $U$ identified by the index $n_{i,j}$.

As an example, consider position $X = \{a, c\}$ and velocity $V = \{(+, b), (-, c)\}$ consisting of two operation pairs. Adding velocity $V$ to position $X$ means that element $b$ is added while element $c$ is removed, resulting in a new position, $X' = \{a, b\}$.

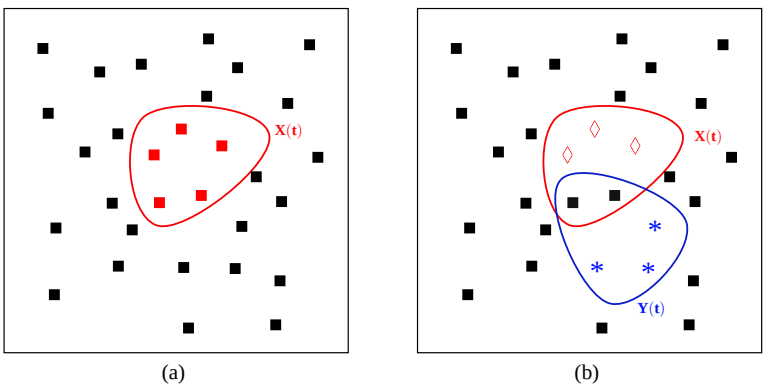


**Fig. 1** Particle positions in SBPSO: (a) shows a particle position $X(t)$ in SBPSO is a set in the universe $U$. The small squares represent elements in the universe $U$. (b) shows a particle position $X(t)$ and a particle's personal best position $Y(t)$. The open diamonds ($\diamond$) represent elements in $X(t)$ that are *not* in $Y(t)$, and the asterisks (*) represent elements in $Y(t)$ that are *not* in $X(t)$.

Attraction towards the personal best $Y(t)$ does not have to mean that the position $X(t)$ moves to the personal best position in one step such that $X(t+1) = Y(t)$. Velocity update equation (4) contains the attraction to $\mathbf{y}_i(t)$ as $c_1\mathbf{r}_1(t)\,[\mathbf{y}_i(t) - \mathbf{x}_i(t)]$, meaning that the difference between $y_{i,j}(t)$ and $x_{i,j}(t)$ is scaled by a factor $\gamma_j(t) = c_1 r_{1,j}(t)$, for all $j = 1,\ldots,n$. If $\gamma_j(t) = 1$, then $x_{i,j}(t+1) = y_{i,j}(t)$, if the other terms of equation (4) are disregarded. If $\gamma_j(t) < 1$ then $\mathbf{x}_i(t)$ is pulled only partly towards $\mathbf{y}_i(t)$ in dimension $j$, while if $\gamma_j(t) > 1$ then $\mathbf{x}_i(t)$ will overshoot $\mathbf{y}_i(t)$ in dimension $j$. In a set-based representation, this overshooting is difficult to define because there is no direction for $X(t)$ to overshoot $Y(t)$ since $U$ has no spatial structure. In contrast, for $X(t+1) = \gamma(t)[Y(t) - X(t)]$, the case $\gamma(t) < 1$ can be defined in a set-based representation, by making only some and not all of the changes required to turn set $X(t)$ into $Y(t)$. Figure 2(a) shows this in action, assuming $\gamma(t) = 0.5$. The set $X(t)$ requires six changes to "move to" $Y(t)$: the three elements indicated as $\diamond$ need to be deleted from $X(t)$, and the three elements indicated as * need to be added to $X(t)$. The scaling by a factor of 0.5 means only three of these changes, selected randomly, are made to $X(t)$. This results in the new position, $X(t+1)$.

The attraction of $X(t)$ to the particle's neighborhood best position works in a similar manner. Figure 2(b) shows positions $X(t), Y(t)$, and $\widehat{Y}(t)$ to partially overlap, with one common element indicated by a triangle ($\triangle$), although this does not necessarily happen in practice. However, should an element be present in all three sets $X(t), Y(t)$, and $\widehat{Y}(t)$, then the above described attraction to $Y(t)$ and $\widehat{Y}(t)$ can *not* lead to the removal of this element from $X(t)$. Also the attraction to $Y(t)$ and $\widehat{Y}(t)$ can *not* lead to the addition of any element to $X(t)$ that is outside of both $Y(t)$ and $\widehat{Y}(t)$. Such elements are indicated with symbol '+' in figure 2(b). For both cases a mechanism needs to be included in SBPSO to ensure that the

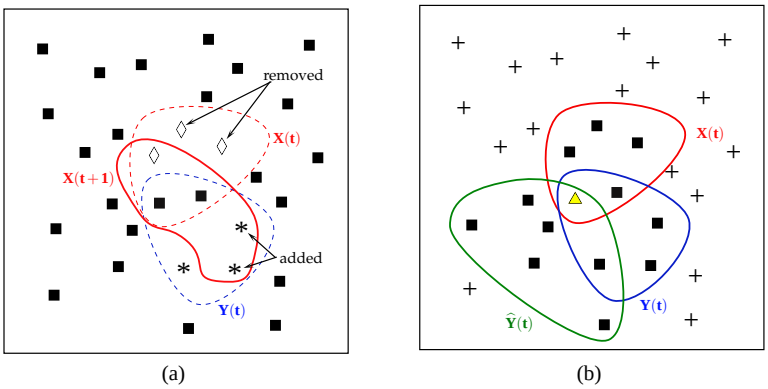


**Fig. 2** Particle attraction and movement in SBPSO: (a) shows how a particle moves from its current position $X(t)$ in direction of its personal best position $Y(t)$ to its new position $X(t+1)$, (b) shows a particle position $X(t)$, its personal best position $Y(t)$, and the neighborhood best position $\widehat{Y}(t)$.

whole universe $U$ is in theory reachable from every possible starting position[1]. These two mechanisms are defined in section 3.2.

For a strict mathematical definition of position, velocity, and objective function, denote with $\mathscr{P}(U)$ the power set (that is, the set of all subsets) of $U$. A position $X_i(t)$ is an element of $\mathscr{P}(U)$. The objective function $f$ maps a position to a quality score in $\mathbb{R}$, written as $f : \mathscr{P}(U) \to \mathbb{R}$ . The velocity $V_i(t)$ is generally defined as a function that maps a position to a new position, that is, $V_i(t) : \mathscr{P}(U) \to \mathscr{P}(U)$.

Note that the definition of velocity using operation pairs is narrower than the general mapping, $V : \mathscr{P}(U) \to \mathscr{P}(U)$. Consider for example $U = \{0,1\}$, and mapping $V$ such that

1. $V(\emptyset) = \emptyset$ ($V$ can not contain any additions),
2. $V(U) = U$ ($V$ can not contain any deletions),
3. $V(\{0\}) = \{1\}$ (requires one addition and one deletion), and
4. $V(\{1\}) = \{0\}$ (requires one addition and one deletion).

Then, $V$ is a valid mapping from $\mathscr{P}(U)$ to $\mathscr{P}(U)$ that can not be denoted as a set of additions and deletions.

## 3.2 Set-Based Operators

To describe SBPSO mathematically, new operators are defined. These operators act on velocities (sets of operation pairs) and positions (sets of elements from $U$) to replicate the PSO concept of velocity and position updates. Special operators are defined to allow (i) a particle position to add elements that are not in the personal best $Y_i(t)$ nor in the neighborhood best

[1] Consider a particle $i$ in SBPSO. Because the swarm usually consists of multiple particles, movement of particles other than $i$ can change $\widehat{Y}_i(t)$ by finding a new best candidate solution. This can then cause $\widehat{Y}_i(t)$ to contain an element $e$ that was first outside of $X_i(t), Y_i(t)$, and $\widehat{Y}_i(t)$. So strictly speaking only elements that are outside of $X_j(t)$ and $Y_j(t)$ for *all* particles $j$ in the swarm (and hence also outside $\widehat{Y}_j(t)$ for all $j$) can not be added to $X_i(t)$ by the attraction mechanism. Similarly, only an element $e$ that is contained in $X_j(t)$ and $Y_j(t)$ for *all* particles $j$ in the swarm is one that can not be removed by the attraction mechanism.

$\widehat{Y}_i(t)$, and (ii) a particle position to remove elements that are present in $X_i(t)$ as well as both $Y_i(t)$ and $\widehat{Y}_i(t)$.

*The addition of two velocities,* $V_1 \oplus V_2$, is a mapping $\oplus : \mathscr{P}(\{+,-\} \times U)^2 \to \mathscr{P}(\{+,-\} \times U)$, that takes two velocities as input and yields a new velocity. Denoted as $V_1 \oplus V_2$, the mapping is defined as the simple union of the two sets of operation pairs:

$$V_1 \oplus V_2 = V_1 \cup V_2. \tag{11}$$

*The difference between two positions,* $X_1 \ominus X_2$, is a mapping $\ominus : \mathscr{P}(U)^2 \to \mathscr{P}(\{+,-\} \times U)$, that takes two positions as input and yields a velocity. If a particle moves by the resulting velocity, the difference between the two positions $X_1$ and $X_2$ is the "distance" that is traversed in one step. This mapping is defined as a set of operation pairs that indicate the steps required to convert $X_2$ into $X_1$ using additions and removals of single elements:

$$X_1 \ominus X_2 = \big(\{+\} \times (X_1 \backslash X_2)\big) \cup \big(\{-\} \times (X_2 \backslash X_1)\big). \tag{12}$$

Therefore, $X_1 \ominus X_2$ is the union of (i) the product of $\{+\}$ and all elements in $X_1$ not in $X_2$ (all such elements are added) and (ii) the product of $\{-\}$ and all elements in $X_2$ not in $X_1$ (all such elements are removed). This operator thus yields the velocity $V$ to get from $X_2$ to $X_1$.

*The multiplication of a velocity by a scalar,* $\eta \otimes V$, is a mapping $\otimes : [0,1] \times \mathscr{P}(\{+,-\} \times U) \to \mathscr{P}(\{+,-\} \times$ that takes a scalar and a velocity and yields a velocity. The mapping is defined to mean picking a subset of $\lfloor \eta \times |V| \rfloor$ elements at random from velocity $V$ to yield a new velocity. Here $\lfloor x \rfloor$ for $x \in \mathbb{R}^+$ denotes the largest $\nu \in \mathbb{N}$ for which $x \geq \nu$. The operand $\eta$ is restricted to values in $[0,1]$ since sets can not have a negative number of elements and sets do not allow multiple instances of the same element. Note that $0 \otimes V = \emptyset$ and $1 \otimes V = V$.

*The addition of a velocity and a position,* $X \boxplus V$, is a mapping $\boxplus : \mathscr{P}(U) \times \mathscr{P}(\{+,-\} \times U) \to \mathscr{P}(U)$ that takes a position and a velocity and yields a position. Recall that a velocity is itself a function that maps a position to a new position. The operator $\boxplus$ is defined as the action of applying the velocity function $V$ to the position $X$:

$$X \boxplus V = V(X) \tag{13}$$

This is further specified as applying the full set of operation pairs $V = \{\nu_1, \ldots \nu_n\}$ to the position $X$ one-by-one and, for each operation pair, one element is added to $X$ or removed from $X$.

Section 3.1 referred to two special mechanisms to remove elements from $X(t)$ that are in $X(t) \cap Y(t) \cap \widehat{Y}(t)$ and to add elements to $X(t)$ from outside of $X(t) \cup Y(t) \cup \widehat{Y}(t)$. These mechanisms are explained below.

*The removal of elements* in $X(t) \cap Y(t) \cap \widehat{Y}(t)$ from a position $X(t)$ uses the operator $\odot^-$. Denoted $\beta \odot^- S$, where $S$ is shorthand for the set of elements $X(t) \cap Y(t) \cap \widehat{Y}(t)$, this is a mapping $\odot^- : [0, |S|] \times \mathscr{P}(U) \to \mathscr{P}(\{+,-\} \times U)$, which takes a scalar and a set of elements, and yields a velocity. The operator $\odot^-$ is implemented as *randomly selecting* a number of elements determined by $\beta$ from $S$ to remove from $X(t)$ and constructs operation pairs that are deletions:

$$\beta \odot^- S = \{-\} \times (\tfrac{N_{\beta,S}}{|S|} \otimes S) \tag{14}$$

The number of elements that are selected from $S$ is denoted by $N_{\beta,S}$, and defined as

$$N_{\beta,S} = \min\{|S|, \lfloor\beta\rfloor + \mathbb{1}_{\{r<\beta-\lfloor\beta\rfloor\}}\} \tag{15}$$

for a random number $r \sim U(0,1)$. Here $\mathbb{1}_{\{bool\}}$ is the indicator function with $\mathbb{1}_{\{bool\}} = 1$ if $bool = true$ and $\mathbb{1}_{\{bool\}} = 0$ if $bool = false$. Thus the number of elements selected is at least $\lfloor\beta\rfloor$, and the fractional remainder $\beta - \lfloor\beta\rfloor$ is the probability of the number of elements selected being one larger. The number of elements is also capped at the number of elements in $S$, which in turn means that $\beta$ is also capped at the number of elements in $S$.

The choice is made to *randomly* select elements from $S$ instead of spending more computational effort to select good candidate elements for removal from $X(t)$. Note that the aim of this operation is to allow exploration of the entire search space. It will likely lead to a worse objective function value at present, as the element removed from $X(t)$ is likely of "good quality" given that it is included in both the personal best and the neighborhood best. The assumption is that any extra effort to select a better element to remove from $X(t)$ will yield only a limited return above that from random selection.

*The addition of elements* outside of $X(t) \cup Y(t) \cup \widehat{Y}(t)$ to $X(t)$ uses the operator $\odot^+$. Denoted $\beta \odot^+ A$, where $A$ is shorthand for the set of elements $U \setminus \left(X(t) \cup Y(t) \cup \widehat{Y}(t)\right)$, this is a mapping $\odot^+ : [0,|A|] \times \mathscr{P}(U) \to \mathscr{P}(\{+,-\} \times U)$, which takes a scalar and a set of elements, and yields a velocity. The operator $\odot^+$ is implemented to use *marginal objective function* information for the position $X(t)$ to choose which elements from $A$ to add to $X(t)$, and constructs operation pairs that are additions. The marginal objective function value of element $e$ for a particle with position $X(t)$ is defined as the objective function value of a new particle with position equal to $X(t)$ *plus* $e$, that is, $X(t) \cup \{e\}$. A $k$-tournament selection algorithm incorporating this marginal objective function information is used to select elements to add to $X(t)$ and is outlined in algorithm 2. The implementation of the operator $\odot^+$ thus depends on the parameter $k$ used in the tournament selection, and is denoted as $\odot^+_k$. The operator $\odot^+_k$ thus is defined as

$$\beta \odot^+_k A = \{+\} \times k\text{-Tournament Selection}(A, N_{\beta,A}) \tag{16}$$

where $N_{\beta,A}$, the number of elements to be added to $X(t)$, is defined as in equation (15). The number of elements to be added is capped at the number of elements in $A$, which in turn means that $\beta$ is also capped at the number of elements in $A$.

**Algorithm 2:** $k$-Tournament Selection$(A, N)$

```
Set V_temp = ∅ ;
for n = 1,...,N do
    for j = 1,...,k do
        Randomly select e_j from A;
        Set score_j = f(X(t) ∪ {e_j});
    end
    Select m ∈ {1,...,k} such that score_m = max_j{score_j};
    Set V_temp = V_temp ⊕ ({+} × e_m);
end
Return V_temp;
```

In summary, $\beta \odot^+_k A$ means selecting $N_{\beta,A}$, possibly overlapping, elements $\{e_j\}_{j=1}^{N_{\beta,A}}$, where each element $e_j$ in turn is the best performing in a tournament of $k$ elements selected

randomly from $A$. The best performing element $e'$ here means maximizing the objective function value of $X_i \cup \{e'\}$. Note that a higher value of $\beta$ leads to more elements from $A$ being added to the position $X(t)$, while a higher value of $k$ means the algorithm is more greedy in selecting which elements to add.

Extra computational effort is exerted in SBPSO by using the $k$-tournament selection to find a "good" element to add to $X(t)$: an additional $k$ objective function evaluations are required. This is done because the set $A$ will, in general, contain many elements that lead to a worse objective function value when added to $X(t)$. Good elements to add to $X(t)$ will thus tend to be rare. The assumption made in this paper is that the extra effort to locate these good elements is worth the extra objective function evaluations.

### 3.3 Update Equations

Using the redefined operators from section 3.2, the velocity update equation for SBPSO used in this paper is

$$\begin{aligned} V_i(t+1) = {} & c_1 r_1 \otimes \left(Y_i(t) \ominus X_i(t)\right) \oplus c_2 r_2 \otimes \left(\widehat{Y}_i(t) \ominus X_i(t)\right) \\ & \oplus \left(c_3 r_3 \odot_k^+ A_i(t)\right) \oplus \left(c_4 r_4 \odot^- S_i(t)\right) \end{aligned} \tag{17}$$

where $A_i(t) = U \setminus \left(X_i(t) \cup Y_i(t) \cup \widehat{Y}_i(t)\right)$ and $S_i(t) = X_i(t) \cap Y_i(t) \cap \widehat{Y}_i(t)$. Parameters are $c_1, c_2 \in [0,1]$, $c_3, c_4 \in [0, |U|]$, and the random numbers $r_i$ are all independently drawn from the uniform distribution on $(0,1)$. Besides the additional velocity components involving $\odot^-$ and $\odot_k^+$, one more difference between equation (17) for SBPSO and equation (4) is the absence of an inertia term. This can be explained by looking at the position update equation for SBPSO:

$$X_i(t+1) = X_i(t) \boxplus V_i(t+1) \tag{18}$$

The velocity $V_i(t+1)$ is a set of operation pairs $\{(\pm, e_1), \ldots, (\pm, e_m)\}$ that is fully applied to the position $X_i(t)$, where each operation pair is an addition or a deletion. Once an element $e$ has been added to the position $X_i(t)$, adding the element again has no impact as a set can only contain a single instance of each element. Therefore, once the velocity has been applied to $X_i(t)$, each operation pair in $V_i(t+1)$ will have no impact if applied to $X_i(t+1)$. Hence, there is no need to include part of $V_i(t)$ in $V_i(t+1)$, which is what the inertia term would do. The SBPSO algorithm is given in algorithm 3.

Note that the order in which the operation pairs from $V_i(t+1)$ are applied to $X_i(t)$ is not relevant, because the individual additions and deletions $v_{i,j}$ in $V_i(t+1)$ from equation (17) can overlap, but can *not* cancel each other out. In other words, there can not be a $j_1 \neq j_2$ such that $v_{i,j_1} = (+, e)$ and $v_{i,j_2} = (-, e)$ are two operation pairs in $V_i(t+1)$ for the same element $e$. To illustrate, assume that $V_i(t+1)$ contains both $(+, e)$ and $(-, e)$ for some element $e$:

- Since attraction towards $Y_i(t)$ or $\widehat{Y}_i(t)$ can only create deletions for elements that are in $X_i(t) \setminus \left(Y_i(t) \cup \widehat{Y}_i(t)\right)$, while the $\odot^-$ operation can only create deletions for elements in $S_i(t)$, the presence of deletion $(-, e)$ in $V_i(t+1)$ implies that $e \in X_i(t)$.
- Since attraction towards $Y_i(t)$ or $\widehat{Y}_i(t)$ can only create additions for elements that are in $X_i(t) \setminus \left(Y_i(t) \cup \widehat{Y}_i(t)\right)$, while the $\odot_k^+$ operation can only create additions for elements in $A_i(t)$, the presence of addition $(+, e)$ in $V_i(t+1)$ implies that $e \notin X_i(t)$ or $e \in \left(U \setminus X_i(t)\right)$.
- For $e$ it must then hold that $e \in X_i(t) \cap \left(U \setminus X_i(t)\right) = \emptyset$. Therefore, such an $e$ can not exist in $V_i(t+1)$.

**Algorithm 3:** Set-Based PSO algorithm (SBPSO) for Maximization Problems

```
Set N equal to the number of particles in the swarm;
for i = 1,...,N do
    Initialize X_i as random subset of U ;
    Initialize V_i = ∅ ;
    Calculate f(X_i) ;
    Initialize f(Y_i) = −∞ ;
    Initialize f(Ŷ_i) = −∞ ;
end
while stopping condition is false do
    for i = 1,...,N do
        // set the personal best position ;
        if f(X_i) > f(Y_i) then
            Y_i = X_i;
        end
        // set the neighborhood best position ;
        for all neighbors l of particle i do
            if (f(Y_i) > f(Ŷ_l) then
                Ŷ_l = Y_i;
            end
        end
    end
    for i = 1,...,N do
        Update V_i according to equation (17);
        Update X_i according to equation (18);
        Calculate f(X_i);
    end
end
```

## 4 Multidimensional Knapsack Problem

The multidimensional knapsack problem (MKP), also called the multidimensional zero-one knapsack or rucksack problem, is a well-known NP-complete optimization problem (Gens and Levner, 1980). The aim is to maximize the total value of all items to be put in a knapsack,

$$\max \sum_{i=1}^{n} v_i x_i \tag{19}$$

subject to the zero-one constraints

$$x_i \in \{0,1\},\ \forall i \in \{1,\ldots,n\} \tag{20}$$

and weight constraints

$$\sum_{i=1}^{n} w_{i,j} x_i \leq C_j,\ \forall j \in \{1,\ldots,m\} \tag{21}$$

There are $n$ items in total, each with value $v_i$. The binary variable $x_i$ indicates whether the item $i$ is present in the knapsack or not. The problem has $m$ weight constraints, where for each constraint $j$ the item $i$ has a weight $w_{i,j}$ and for each constraint the total weight $\sum_i w_{i,j}\, x_i$ may not exceed the capacity $C_j$. In the remainder of this paper, all mention of the MKP's constraints refer to the weight constraints, as the zero-one constraints are considered part of the definition of the MKP as a class of problems.

A well-formulated multidimensional knapsack problem also adheres to the value constraints

$$v_i > 0, i \quad \forall i \in \{1,\ldots,n\} \tag{22}$$

and constraints on the total weight

$$w_{i,j} \leq C_j < \sum_{i=1}^{n} w_{i,j}, \quad \forall i \in \{1,\ldots,n\}, j \in \{1,\ldots,m\} \tag{23}$$

Note that any zero-one integer problem with non-negative coefficients can be formulated as a MKP. The first mention of such problems was with regards to capital budgeting (Lorie and Savage, 1955). A recent overview of exact methods and analytical approximations for the MKP can be found in (Puchinger et al., 2010). Population based optimization algorithms have also been applied to the MKP including genetic algorithms (GA) (Chu and Beasley, 1998; Khuri et al., 1994), ant colony optimization (Kong et al., 2008), as well as PSO. Kong and Tian (2006) used the binary PSO which includes a heuristic repair operator to avoid infeasible solutions, while Hembecker et al. (2007) used penalty functions to steer the search towards solutions that satisfy the MKP's constraints. Labed et al. (2011) proposed a hybrid GA binary PSO algorithm that includes a crossover operator and a separate repair operator that modifies positions to represent feasible solutions to the MKP. Wang et al. (2008) used the MKP to compare the binary PSO to two other discrete PSO variants, namely MBPSO and PBPSO.

Recent studies into the MKP frequently use the benchmark problems mentioned in Chu and Beasley (1998) to compare the performance of algorithms. These problems that are available on-line at the Operations Research Library (ORLib) at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html`, are divided into two sets: small MKP and large MKP. The small MKP is a collection of 55 problems that have been mentioned in literature prior to the paper by Chu and Beasley (1998). The large MKP is a collection of 270 randomly generated MKPs with number of items $n = 100, 250$, or 500, number of constraints $m = 5, 10$, or 30, and tightness ratio $0.25, 0.50$, or $0.75$. The tightness ratio, denoted $r$, was used in the construction of the problems as follows: first the weights $w_{i,j}$ and values $v_i$ were chosen randomly. Then the capacity constraint variables $C_j$ in equation (21) were set according to

$$C_j = r \sum_{i=1}^{m} w_{i,j}, \quad \forall j \in \{1,\ldots,m\} \tag{24}$$

The three choices for each of the three parameters $n, m$, and $r$ yield 27 different problem specifications. For each problem specification, ten problem instances are included in the problem set.

In general, these three problem parameters have the following effects on the MKP search space:

- a larger number of items, $n$, increases the search space and hence makes the problem of finding the optimum harder,
- a larger number of constraints, $m$, makes the feasible part of the search space smaller, and
- a larger tightness ratio, $r$, means that the weight constraints are *less* restrictive and that the feasible part of the search space becomes larger.

For the small MKPs, the optimal solutions are known, whilst this is not the case for all of the large MKPs. To be able to compare results for the large MKPs, Chu and Beasley (1998) obtained an upper bound for the objective function value by solving the linear programming (LP) relaxation of the large MKPs. The LP relaxation of the problem changes the zero-one constraint in equation (20) on $x_i$ from an integer constraint to a continuous constraint:

$$x_i \in [0,1],\ \forall i \in \{1,\ldots,n\} \tag{25}$$

thereby making the problem *easier* to solve and no longer NP-hard. The LP relaxed version of the MKP can efficiently be solved using standard linear programming solvers (Chu and Beasley, 1998).

## 5 Experimental Procedure

This section describes the experimental procedure followed for the purposes of this study. Section 5.1 describes the configuration of the algorithms used in the comparisons with SBPSO as well as the configuration of SBPSO itself. The problem sets with small and large MKPs mentioned in section 4 are then split into a set of tuning problems and a set of test problems, and the objective function is explicitly stated. Section 5.2 gives an explanation of the procedure used to tune the parameters of each algorithm, and provides the parameter values obtained from the tuning process.

### 5.1 Algorithm Configurations

*Algorithms:* The proposed SBPSO algorithm is compared to three other PSO algorithms: BPSO by Kennedy and Eberhart (1997), MBPSO by Shen et al. (2004), and PBPSO by Zhen et al. (2008). Refer to section 2.3 for detailed descriptions of these algorithms. These algorithms were chosen because they do no incorporate any domain specific methods such as a repair operator.

For BPSO, MBPSO, and PBPSO the candidate solution is directly represented by binary-valued particle positions: the bit values are directly interpreted as the $x_i$ values in equation (19). That is, a particle indicates the assignment of items to the knapsack. For SBPSO, in order to evaluate a solution, the $x_i$ from equation (19) are set to 1 for all items that are included in the particle position set, and set to 0 for all items that are not.

*Swarm size:* An important parameter in PSO algorithms is the number of particles in the swarm. While the optimal number of particles for a specific algorithm-problem pair can be problem dependent, this study used the same number of particles for all algorithms and for all problems in each problem set: for small MKPs the number of particles was set to 25, while for large MKPs the number of particles was set to 50.

*Topologies:* Each of the four PSO algorithms is used with each of the following three topologies: star, ring, and Von Neumann. This results in 12 algorithm-topology pairs. The pairs with a star topology are referred to as global best PSO shortened to GB in the tables in the remainder of this document. Similarly, the pairs with a ring topology are referred to as local best PSO shortened to LB, and the pairs with a Von Neumann topology are referred to as VN in the tables.

Particles organized in a swarm topology are considered connected if they are in each other's neighborhood. Particles that are not in each other's neighborhood are connected indirectly due to overlap between neighborhoods. If, for example, particle $i$ is not connected to particle $j$, but the two particles share a common neighbor $k$, then the path $i-k-j$ connects particles $i$ and $j$ in the topology. The distance between two particles in a topology is determined by the *shortest* path that connects the two particles. For particles $i$ and $j$ from the example, the $i-k-j$ path is the shortest path, and the distance between $i$ and $j$ thus is 2. The average distance across all possible pairs in a swarm, called the *average shortest path length*, is a measure of how connected the swarm is.

A swarm with the star topology always has an average shortest path length of 1, as each particle is in each other particle's neighborhood. For the Von Neumann topology, the average shortest path length depends on the number of particles in the swarm. For swarms of 25 and 50 particles, the Von Neumann topology leads to average shortest path lengths of 2.5 and 3.5 respectively. For the ring topology, the average shortest path length depends not only on the swarm size, but also on the neighborhood size. A neighborhood size of 4 was chosen for the experiments of this study, such that the swarms with a ring topology are less connected than those using either of the other two topologies. This resulted in average shortest path lengths for swarms with the ring topology of 3.5 for a swarm of 25 particles, and 6.6 for a swarm of 50 particles.

Therefore, in the experiments conducted, swarms with the star topology were the most connected, swarms with the ring topology were the least connected, and swarms with the Von Neumann topology had an intermediate level of connectedness.

*Problem set:* The MKPs used in the experiments consist of two main problem sets: 55 small MKPs and 270 large MKPs as described in section 4. The problem name reflects the filename from the ORLib source the problem comes from, plus a number indicating which problem from that file it refers to. For example, "mknap2-3" is the third problem found in the file *mknap2.txt*. The two sets of problems were each further split into a tuning set used to find the best parameters for the algorithms, and a test set that is used to compare the performance of the tuned algorithms.

For the small MKPs, a tuning set of 15 problems was manually chosen. The remaining 40 problems formed the test set. Which small MKPs were selected for the tuning set and which for the test set is summarized in table 1. The tuning set was chosen to reflect the range of problem sizes in the entire set of 55 problems, with the number of variables $n$ ranging from 20 to 90, and the number of constraints $m$ ranging from 2 to 30.

The three smallest problems (mknap1-1, mknap1-2, mknap1-3) were left out of the tuning set on purpose, as the search spaces for these problems are small ($2^6 = 64$, $2^{10} = 1024$, and $2^{15} = 32768$ possible solutions respectively) and hence the problems are quite simple to solve. For simple problems, little difference is to be expected in the performance of the algorithm control parameters, so the problems yield little information on which parameters are best.

For the large MKP, the total set of 270 problems consists of 27 subsets of problems, each of which contains 10 random instances for a given combination of problem parameters $n, m$, and tightness ratio $r$. For the tuning set, one problem was selected at random from each of the 27 subsets, and the remaining 243 problems formed the test set. The 27 tuning problems, each with the number of variables, the number of constraints, and tightness ratios are summarized in table 2.

**Table 1** Split of 55 small MKPs into 15 tuning and 40 test problems

| Tuning Set | | | Test Set | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| problem | n | m | problem | n | m | problem | n | m | problem | n | m |
| mknap1-4 | 20 | 10 | mknap1-1 | 6 | 10 | mknap2-23 | 50 | 5 | mknap2-42 | 34 | 4 |
| mknap1-5 | 28 | 10 | mknap1-2 | 10 | 10 | mknap2-24 | 60 | 5 | mknap2-43 | 18 | 2 |
| mknap2-10 | 71 | 2 | mknap1-3 | 15 | 10 | mknap2-25 | 60 | 5 | mknap2-44 | 20 | 10 |
| mknap2-15 | 30 | 5 | mknap1-6 | 39 | 5 | mknap2-27 | 60 | 5 | mknap2-46 | 37 | 30 |
| mknap2-17 | 40 | 5 | mknap1-7 | 50 | 5 | mknap2-29 | 70 | 5 | mknap2-47 | 28 | 4 |
| mknap2-2 | 60 | 30 | mknap2-1 | 60 | 30 | mknap2-3 | 24 | 2 | mknap2-5 | 24 | 2 |
| mknap2-20 | 50 | 5 | mknap2-11 | 30 | 5 | mknap2-30 | 70 | 5 | mknap2-6 | 24 | 2 |
| mknap2-26 | 60 | 5 | mknap2-12 | 30 | 5 | mknap2-31 | 70 | 5 | mknap2-7 | 24 | 2 |
| mknap2-28 | 70 | 5 | mknap2-13 | 30 | 5 | mknap2-32 | 80 | 5 | mknap2-8 | 24 | 2 |
| mknap2-33 | 80 | 5 | mknap2-14 | 30 | 5 | mknap2-34 | 80 | 5 | mknap2-9 | 71 | 2 |
| mknap2-39 | 90 | 5 | mknap2-16 | 40 | 5 | mknap2-35 | 80 | 5 | | | |
| mknap2-4 | 24 | 2 | mknap2-18 | 40 | 5 | mknap2-36 | 90 | 5 | | | |
| mknap2-41 | 27 | 4 | mknap2-19 | 40 | 5 | mknap2-37 | 90 | 5 | | | |
| mknap2-45 | 40 | 30 | mknap2-21 | 50 | 5 | mknap2-38 | 90 | 5 | | | |
| mknap2-48 | 35 | 4 | mknap2-22 | 50 | 5 | mknap2-40 | 90 | 5 | | | |

**Table 2** Large MKPs selected for tuning

| **problem** | $n$ | $m$ | $r$ | **problem** | $n$ | $m$ | $r$ | **problem** | $n$ | $m$ | $r$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| mknapcb1-6 | 100 | 5 | 0.25 | mknapcb4-3 | 250 | 5 | 0.25 | mknapcb7-1 | 500 | 5 | 0.25 |
| mknapcb1-17 | 100 | 5 | 0.50 | mknapcb4-12 | 250 | 5 | 0.50 | mknapcb7-19 | 500 | 5 | 0.50 |
| mknapcb1-27 | 100 | 5 | 0.75 | mknapcb4-27 | 250 | 5 | 0.75 | mknapcb7-30 | 500 | 5 | 0.75 |
| mknapcb2-7 | 100 | 10 | 0.25 | mknapcb5-7 | 250 | 10 | 0.25 | mknapcb8-10 | 500 | 10 | 0.25 |
| mknapcb2-11 | 100 | 10 | 0.50 | mknapcb5-20 | 250 | 10 | 0.50 | mknapcb8-16 | 500 | 10 | 0.50 |
| mknapcb2-22 | 100 | 10 | 0.75 | mknapcb5-21 | 250 | 10 | 0.75 | mknapcb8-26 | 500 | 10 | 0.75 |
| mknapcb3-3 | 100 | 30 | 0.25 | mknapcb6-7 | 250 | 30 | 0.25 | mknapcb9-8 | 500 | 30 | 0.25 |
| mknapcb3-20 | 100 | 30 | 0.50 | mknapcb6-16 | 250 | 30 | 0.50 | mknapcb9-18 | 500 | 30 | 0.50 |
| mknapcb3-24 | 100 | 30 | 0.75 | mknapcb6-23 | 250 | 30 | 0.75 | mknapcb9-26 | 500 | 30 | 0.75 |

*Objective function:* The MKP is defined as a maximization problem. The objective function used was the same for all the PSO algorithms. For particles that represent a feasible solution to the MKP, that is, which satisfy all $m$ constraints in equation (21), the objective function value was set equal to the sum of the values of the items in the particle. Particles that do not represent a feasible solution because they violate at least one of the constraints in equation (21), were assigned a objective function value of minus infinity. Since a particle uses its position to represent a solution, the objective function value of a particle is computed as $f\big(X(t)\big)$, defined as

$$f\big(X(t)\big) = \begin{cases} \sum_{i=1}^{n} v_i\, x_i & \text{if } \forall j \in \{1,\ldots,m\} : \sum_{i=1}^{n} w_{i,j} x_i \leq C_j \\ -\infty & \text{if } \exists j \in \{1,\ldots,m\} : \sum_{i=1}^{n} w_{i,j} x_i > C_j \end{cases} \tag{26}$$

In order to facilitate a comparison of results across different problems, the results in section 7 do not show the raw objective function values. For the small MKPs, the error between the best objective function value found and the known optimum is shown. Since the optimal solutions are not known for all the large MKPs, for these problems the error between

the best found objective function value and the LP relaxation bound is shown instead. The LP relaxation bounds were obtained using the Java wrapper of *lp_solve 5.5*, which is based on the revised simplex method[2].

*Initialization:* Particles were initialized randomly for each algorithm-topology pair. For the BPSO, MBPSO, and PBPSO algorithms, the positions were initialized randomly in $\{0,1\}^n$, while the velocities for BPSO and PBPSO were initialized randomly in $[-1,1]^n$, following (Eberhart and Shi, 2001). For PBPSO the continuous-valued positions, $\mathbf{x}'_i(0)$, were initialized as $\mathbf{0}$, to ensure that no initial bias was included in the discrete-valued positions, $\mathbf{x}_i(0)$. For the SBPSO algorithm, the positions were randomly initialized, such that each element had a 0.5 chance of being included, and all velocities were initialized as an empty set.

*Stopping conditions:* For each independent run of an algorithm, the same stopping conditions were applied:

1. the best objective function value in the swarm equaled the known optimum (in case of small MKPs) or equaled the LP relaxed bound (in case of large MKPs),
2. the best objective function value in the swarm had not improved for 2500 iterations, or
3. more than 5000 iterations had passed.

*Number of independent runs:* PSO is a stochastic optimization algorithm, and thus individual runs of the algorithm can have different results. Hence, multiple independent runs of the algorithms have to be executed and the average performance reported. For the small MKPs, 30 independent runs were used for tuning the algorithms and 100 independent runs were used to ascertain the average performance on the test problems. For the large MKPs, 30 independent runs were used both for tuning the algorithms and to determine the performance on the test problems.

## 5.2 Control Parameter Tuning

This section describes how each of the 12 algorithm-topology pairs was tuned on both problem sets separately. Section 5.2.1 describes how the parameter tuning was performed. Sections 5.2.2 and 5.2.3 summarize the resulting best control parameter values for each algorithm-topology pair.

### *5.2.1 Parameter Tuning Process*

While a number of efficient parameter tuning approaches exist, for example, F-Race (Birattari et al., 2002), the tuning process described in this section is more appropriate for the sensitivity analysis conducted in Section 6.

For each of the 12 algorithm-topology pairs, a similar process was used to tune the algorithm's parameters, although the number of control parameters differed: MBPSO has only two parameters, while BPSO has four, PBPSO has six, and SBPSO has five parameters. Each algorithm-topology was tuned twice: once on the tuning set of small MKPs and once on the large MKPs. The end result of the parameter tuning thus was a total of 24 tuned parameter combinations.

[2] M. Berkelaar, K. Eikland, P. Notebaert, *lpsolve version 5.5*, http://lpsolve.sourceforge.net/5.5/

Table 3 lists the ranges within which each parameter for each of the PSO algorithms was tuned. For each of the four PSO algorithms, 128 parameter combinations were generated that span the parameter space. Only static control parameters were considered. In order to generate the parameter combinations in a manner that ensures that the parameter space was covered well, sequences of Sobol pseudo-random numbers were used according to the method proposed by Franken (2009).

**Table 3** Parameter ranges used in tuning the four PSO algorithms

| algorithm | BPSO | PBPSO | algorithm | MBPSO | SBPSO |
|---|---|---|---|---|---|
| $\omega$ | [0.50, 0.99] | [0.50, 0.99] | $p_{\text{stat}}$ | [0.00, 1.00] | |
| $c_1$ | [0.00, 5.00] | [0.00, 5.00] | $p_{\text{reset}}$ | [0.00, 1.00] | |
| $c_2$ | [0.00, 5.00] | [0.00, 5.00] | $c_1$ | | [0.00, 1.00] |
| $V_{\max}$ | [1.00, 10.00] | [1.00, 10.00] | $c_2$ | | [0.00, 1.00] |
| $R$ | | [1.00, 100.00] | $c_3$ | | [0.50, 5.00] |
| $p_{\text{mut}}$ | | [0.00, 0.50] | $c_4$ | | [0.50, 5.00] |
| | | | $k$ | | $\{1,\ldots,9\}$ |

Even though the number of dimensions of the parameter space differs depending on the PSO algorithm, the same number of parameter combinations was used in tuning each of the algorithm-topology pairs on each of the problem sets. Hence, for the MBPSO algorithm, which has only two parameters, the parameter combinations provided a denser covering of the (smaller) parameter space than for the other PSO algorithms which each has at least four parameters.

Note that the tuning process used the same parameter combinations for each of the PSO algorithms for each of the three topologies, and on both problem sets. Thus, for example, in tuning BPSO using a star topology on the small MKPs, the same 128 parameter combinations were considered as in tuning BPSO using a Von Neumann topology on the large MKPs.

The next step in the tuning process was to determine the best parameter combination for each of the algorithm-topology pairs on each of the problem sets. To do this, 30 independent runs were conducted for each of the parameter combinations, on all the tuning problems in the problem set. For each problem, the average of the best objective function value achieved by each of the 128 parameter combinations over the 30 runs was determined. The parameter combinations were ranked in order of the average objective function value for each problem separately. Next, the *average rank* was determined for each parameter combination by averaging over all the problems. The parameter combination with the lowest average rank was deemed best and chosen as the tuning result. This method weighed the contribution of each tuning problem equally, and by using the rank of the objective function value instead of the objective function itself, a fair comparison was made using problems that have different optima and different search landscapes.

The results of tuning the 12 algorithm-topology pairs on the small MKPs and the large MKPs are discussed in sections 5.2.2 and 5.2.3 respectively.

### *5.2.2 Small Multidimensional Knapsack Problems*

Table 4 summarizes the best parameters found using the parameter tuning procedure described in section 5.2.1, for each of the algorithm-topology pairs on the small MKPs.

**Table 4** Tuned parameters for small MKPs

| algorithm | BPSO | | | PBPSO | | |
|---|---|---|---|---|---|---|
| topology | GB | LB | VN | GB | LB | VN |
| $\omega$ | 0.9211 | 0.9594 | 0.9709 | 0.6876 | 0.6455 | 0.6455 |
| $c_1$ | 4.6094 | 2.8125 | 4.4141 | 0.7422 | 4.2969 | 4.2969 |
| $c_2$ | 1.3281 | 1.5625 | 1.9922 | 0.5078 | 4.7656 | 4.7656 |
| $V_{\max}$ | 5.9219 | 7.1875 | 5.1484 | 3.3203 | 4.2344 | 4.2344 |
| $R$ | | | | 37.352 | 64.422 | 64.422 |
| $p_{\text{mut}}$ | | | | 0.0742 | 0.0391 | 0.0391 |

| algorithm | MBPSO | | | SBPSO | | |
|---|---|---|---|---|---|---|
| topology | GB | LB | VN | GB | LB | VN |
| $p_{\text{stat}}$ | 0.4844 | 0.4766 | 0.4766 | | | |
| $p_{\text{reset}}$ | 0.3906 | 0.3203 | 0.3203 | | | |
| $c_1$ | | | | 0.9297 | 0.5156 | 0.5156 |
| $c_2$ | | | | 0.2266 | 0.4531 | 0.4531 |
| $c_3$ | | | | 1.3086 | 1.8359 | 1.8359 |
| $c_4$ | | | | 2.1523 | 2.2578 | 2.2578 |
| $k$ | | | | 7 | 7 | 7 |

For BPSO, the attraction to the neighborhood best particle, $c_1$, increased as the swarm topology was less connected: highest for gbest BPSO, lowest for lbest BPSO. The attraction to the personal best, $c_2$, ranged from 1.3 for the star topology to 2.0 for the Von Neumann topology, and was clearly smaller than the values for $c_1$. The inertia weight $\omega$ was high for each of the three topologies, as was the $V_{\max}$, which was above 5 in all cases.

For PBPSO, the best parameter value combinations for lbest PBPSO and the Von Neumann topology were the same, but the best parameter values found for gbest PBPSO were quite different, mainly with much lower $c_1$ and $c_2$ values. Note that, compared to BPSO, the inertia weight for the best parameter value combinations for PBPSO was much smaller.

For MBPSO, the three values found for the static probability, $p_{\text{stat}}$, were similar and comparable to the value of 0.5 used by the original authors, Shen et al. (2004). The value of $p_{\text{reset}}$ of 32% to 39% was, however, more than triple the 10% used by Shen et al. (2004), indicating that a high proportion of random resets was beneficial.

For SBPSO, the parameter value combinations for the ring and Von Neumann topologies were the same, while for the star topology a different parameter value combination was optimal with a much higher $c_1$ and lower $c_2$. Section 6 gives a detailed analysis of the sensitivity of SBPSO's parameters using the tuning results.

### *5.2.3 Large Multidimensional Knapsack Problems*

Table 5 summarizes the best parameter values found using the parameter tuning procedure described in section 5.2 for each of the algorithm-topology pairs on the large MKPs.

For BPSO, the best parameter value combinations found on the large MKPs were exactly the same for each of the three topologies, characterized by a high inertia weight $\omega$, high $V_{\max}$ and $c_2 > c_1$. The latter inequality indicates a stronger attraction to the neighborhood best position than to the personal best position, which is the reverse of the results found for BPSO on the small MKPs, where $c_1 > c_2$.

For PBPSO, the ring and the Von Neumann topologies yielded the same best parameter value combination. For all three topologies, the values found for the inertia weight, $\omega$, were

**Table 5** Tuned parameters for large MKPs

| algorithm | BPSO | | | PBPSO | | |
|---|---|---|---|---|---|---|
| topology | GB | LB | VN | GB | LB | VN |
| $\omega$ | 0.9785 | 0.9785 | 0.9785 | 0.6263 | 0.7373 | 0.7373 |
| $c_1$ | 2.4609 | 2.4609 | 2.4609 | 3.8672 | 2.7344 | 2.7344 |
| $c_2$ | 4.1016 | 4.1016 | 4.1016 | 3.6328 | 1.9531 | 1.9531 |
| $V_{max}$ | 9.2266 | 9.2266 | 9.2266 | 8.9453 | 7.0469 | 7.0469 |
| $R$ | | | | 74.477 | 82.984 | 82.984 |
| $p_{mut}$ | | | | 0.0117 | 0.0078 | 0.0078 |
| algorithm | MBPSO | | | SBPSO | | |
| topology | GB | LB | VN | GB | LB | VN |
| $p_{stat}$ | 0.4531 | 0.2266 | 0.3828 | | | |
| $p_{reset}$ | 0.1094 | 0.0703 | 0.1016 | | | |
| $c_1$ | | | | 0.9297 | 0.3672 | 0.3672 |
| $c_2$ | | | | 0.2266 | 0.9141 | 0.9141 |
| $c_3$ | | | | 1.3086 | 1.5898 | 1.5898 |
| $c_4$ | | | | 2.1523 | 1.3086 | 1.3086 |
| $k$ | | | | 7 | 3 | 3 |

similar. These values are also very similar to the corresponding values found during tuning on the small MKPs: a relative difference of only 10%-14% was seen. For all three topologies, the parameter values found for $V_{max}$, $R$, and $p_{mut}$ showed some differences between those for gbest PBPSO and the other two topologies. But these differences are much smaller than the large difference for these parameter values compared to the tuning results on the small MKPs. On the large MKPs, the best values for $V_{max}$ and $R$ were much higher. Also, the values for $p_{mut}$ were lower, indicating that having many random mutations was less helpful on the large MKPs. For the gbest PBPSO, the best values for $c_1$ and $c_2$ resulted in much higher values than those found for the small MKPs, while lbest PBPSO and the Von Neumann topology yielded lower values than on the small MKPs.

For MBPSO there was some variation in the best values of $p_{stat}$ compared to the values found on the small MKPs: a lower value was found on the large MKPs for both the gbest and lbest MBPSO, while for the Von Neumann topology, $p_{stat}$ was higher on the large MKP. For $p_{reset}$, the best values found were close to the 10% used by Shen et al. (2004).

For SBPSO, the best parameter values found for gbest SBPSO were exactly the same as those found on the small MKPs. The best parameter values for lbest SBPSO and the Von Neumann topology matched, but were quite different than those found on the small MKP: the attraction to the personal best, $c_2$, was much higher for the larger MKPs, while the attraction to the neighborhood best, $c_1$, was lower.

## 6 Sensitivity Analysis of Set-Based Particle Swarm Optimization

This section analyzes the sensitivity of SBPSO to different values of its control parameters. Such sensitivity analysis is important, as little is yet known about what are good values for its control parameters.

The sensitivity analysis procedure is summarized in section 6.1, followed by the results for each of the three topologies, star, ring with neighborhood size of 4, and Von Neumann in sections 6.2, 6.3 and 6.4 respectively. A discussion of the relative performance of SBPSO parameters is given in section 6.5.

### 6.1 Sensitivity Analysis Procedure

The sensitivity of the performance of SBPSO to each individual control parameter was investigated using cumulative histograms. For each individual parameter, the horizontal axis of the histogram consists of bins which divide the parameter range into equally sized subranges. The vertical axis displays the number of parameter value combinations that fall in each bin, split into four groups based on the performance of the parameter value combination in the tuning process. If a particular bin for an individual parameter contains a large number of parameter combinations that are considered "good", this implies that the subrange for the individual parameter associated with the bin is good. This section describes how the histograms were constructed, resulting in a histogram for each of the three SBPSO-topology combinations, for each of the five control parameters. In total, 15 histograms were generated.

Note that a good parameter value combination for SBPSO requires that all five parameters individually have a good or at least reasonable value: if even one parameter has a bad value, the parameter value combination as a whole performs badly. The consequence of this is that, if a specific parameter value combination performs badly, this gives little information on whether the *individual* parameter values in that combination are good or bad: any single individual parameter value could be bad, or all values could be bad. Therefore, it is the parameter value combinations that perform well *as a whole* which contain information on the individual parameters. Hence, the sensitivity analysis focused on the 25% of parameter value combinations that performed best in the tuning process.

The performance of a parameter value combination was set equal to its average rank on the small MKPs and the large MKPs tuning sets combined, with each of the two tuning sets weighed equally. The full procedure to construct the histograms used in the following sections consisted of the following steps:

1. For each parameter value combination, the performance was set equal to 0.5 times the average rank on the small MKPs tuning set plus 0.5 times the average rank on the large MKPs tuning set.
2. The parameter value combinations were then themselves ranked based on the performance calculated in step 1.
3. The ranked parameter value combinations were split into quartiles, labeled A for the best 25%, B and C for the next two quartiles respectively, and D for the worst 25% of parameter value combinations[3].
4. Then, for each individual parameter, the parameter range was split into bins:
   (a) parameter values for $c_1$ and $c_2$ took values in the range $[0.0, 1.0]$, with the values grouped into the 10 bins, $[0.0, 0.1), [0.1, 0.2), \ldots, [0.9, 1.0]$;
   (b) parameter values for $c_3$ and $c_4$ took values in the range $[0.5, 5.0]$, with the values grouped into the nine bins, $[0.5, 1.0), [1.0, 1.5), \ldots, [4.5, 5.0]$; and
   (c) parameter values for $k$ took values in the range $1, \ldots, 9$, with the values grouped into nine bins containing one value each.

   These bins form the horizontal axis of the histogram.
5. For each individual parameter, the parameter value combinations were allocated one by one to a bin, based on the value of the individual parameter in the combination. In each bin, a count was kept of the number of parameter value combinations labeled A, B, C, and D separately. Consider, for example, the parameter value combination labeled A

[3] The parameter value combinations with label A are considered to be good combinations, those with label B are considered reasonable combinations.

with values $(0.95, 0.52, 2.03, 3.17, 3)$. Allocating this parameter value combination to a bin for the individual parameter $c_1$ entailed increasing by one the count of label A combinations in the sub-range bin $[0.9, 1.0]$.
6. For each individual parameter, a cumulative histogram was then constructed with, for each bin, the number of label A combinations at the bottom in black, on top of which the number of label B combinations is given in dark gray, and on top of that the number of label C combinations in light gray. The remaining parameter value combinations with label D were stacked at the top and "shown" in white.
7. As a final step, each of the bins was scaled to $[0, 1]$ for ease of comparison, as not all parameter value bins contained the same number of parameter value combinations[4].

Each histogram can be interpreted in the same manner: the black graph at the bottom shows the distribution of good parameter value combinations (labeled A for the best 25% combinations) for the individual parameter across the bins. The dark grey graph stacked on top of the black graph similarly shows the distribution of reasonable-but-not-good parameter value combinations (labeled B). Because the histogram is stacked, the top of the dark grey graph is the sum of the fractions of label A and label B combinations in each bin, indicating the fraction of parameter value combinations that are reasonable or better.

Note that, for the acceleration parameters $c_1$ to $c_4$, the bin labels on the horizontal axis of the histograms identify the *lower boundary* of the sub-range linked to that bin. For example, the bin for $c_1$ labeled 0.3 identifies the sub-range $[0.3, 0.4)$, and the bin for $c_3$ labeled 1.5 identifies the sub-range $[1.5, 2.0)$.

### 6.2 Global Best Set-Based Particle Swarm Optimization

Figure 3 shows the resulting histograms for the parameter sensitivity analysis on the gbest SBPSO.

For gbest SBPSO, high $c_1$ values led to better results: parameters in the range $c_1 \geq 0.8$ covered 20% of the parameter space but accounted for more than 45% of label A (the best quantile) parameter combinations. For the $c_1$ bins with $c_1 < 0.4$, only a few combinations were labeled A. For parameter $c_2$, the best results were found in the sub-range $[0.3, 0.6)$, which accounted for half the combinations labeled A. Values for $c_2$ up to 0.3 resulted in bad performance. For $c_3$ most of the best parameter values were in the range $[1.5, 3.5)$, while the performance of those four bins was approximately the same. For $c_4$ parameter values between 1.5 and 4.0 scored best, with higher bins performing slightly better, except for the $[3.0, 3.5)$ bin. Larger values of $k$ (indicating that a larger tournament was used to select each element to add based on marginal objective function values) led to better results, but the difference across the bins was quite small.

[4] Note that, by construction, the parameter value bins for an individual parameter contain almost the same number of parameter value combinations. For parameters $c_1$ and $c_2$, the 128 combinations were divided over 10 equally sized bins, resulting in 12 or 13 combinations in each bin. For parameters $c_3$, $c_4$, and $k$, the 128 combinations were divided over nine equally sized bins, resulting in 14 or 15 combinations in each bin. By dividing the results in each bin by the total number of combinations in the bin, the number of combinations with each label was changed instead into the fraction of all combinations with that label, so that results are better comparable across bins.

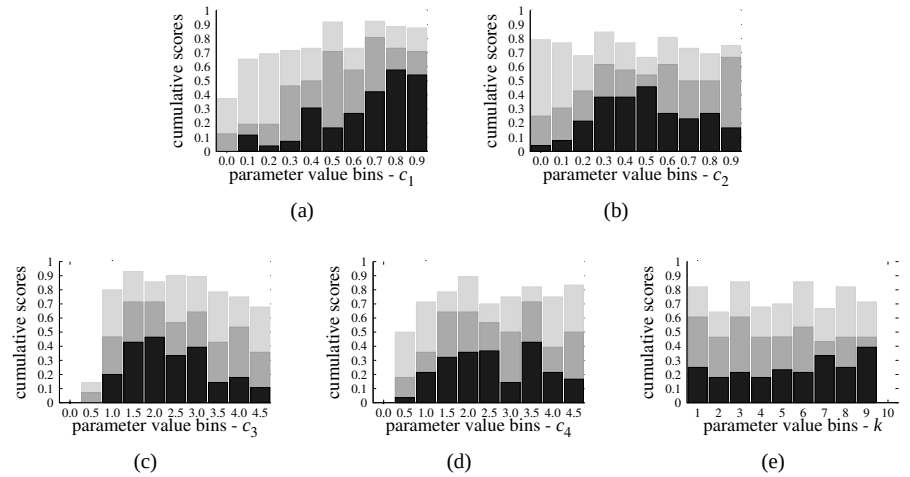


**Fig. 3** Sensitivity analysis of gbest SBPSO parameters: (a) $c_1$, (b) $c_2$, (c) $c_3$, (d) $c_4$, and (e) $k$.

## 6.3 Local Best Set-Based Particle Swarm Optimization

Figure 4 shows the resulting histograms for the parameter sensitivity analysis on the lbest SBPSO with neighborhood size 4.

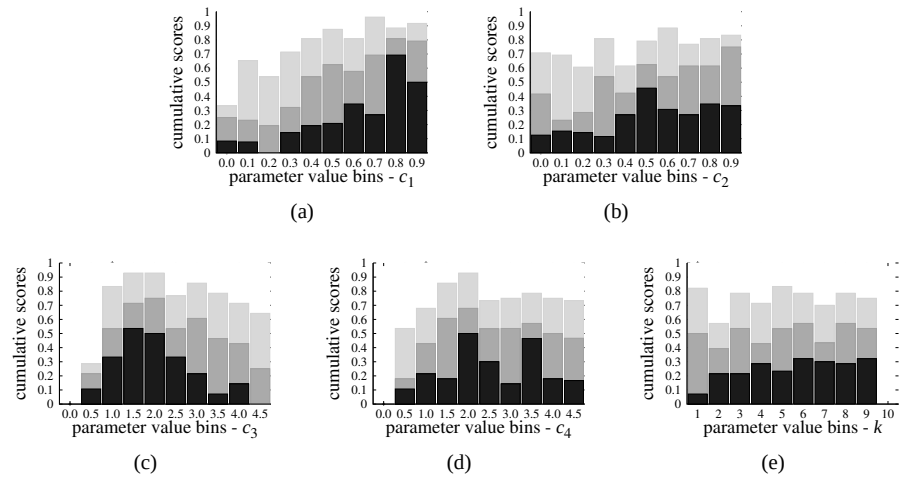


**Fig. 4** Sensitivity analysis of lbest SBPSO parameters: (a) $c_1$, (b) $c_2$, (c) $c_3$, (d) $c_4$, and (e) $k$.

For lbest SBPSO, high $c_1$ values led to better performance: parameters in the range $c_1 \geq 0.8$ covered 20% of the parameter space but accounted for more than 47% of label A parameter value combinations. Low $c_1$ values had few results labeled A, especially those for $c_1 < 0.3$. For parameter $c_2$, the best values were found in the range $[0.5, 0.6)$, but all bins with $c_2 > 0.4$ scored comparably well, while values $c_2 < 0.4$ clearly performed worse. The best

$c_3$ parameter values were in the range $[1.0, 2.5)$, and performance worsened proportionally for parameter values further away from 2.0. For $c_4$, the two bins $[2.0, 2.5)$ and $[3.5, 4.0)$ clearly had the most good results, while the parameter values between 2.5 and 3.5 scored worse. Larger values of $k$ led to more good results, but only $k = 1$ clearly performed worse based on the fraction of label A combinations. Combining label A and label B contributions resulted in no significant difference between the performance of each of the nine values of $k$: any value of $k$ led to the same number of reasonable parameter value combinations.

### 6.4 Von Neumann Set-Based Particle Swarm Optimization

Figure 5 shows the resulting histograms for the parameter sensitivity analysis on SBPSO with the Von Neumann topology.

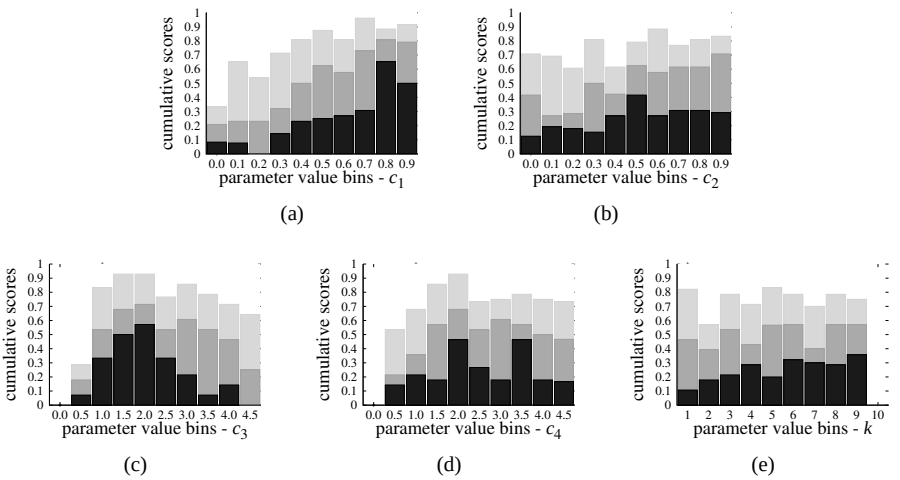


**Fig. 5** Sensitivity analysis of Von Neumann SBPSO parameters: (a) $c_1$, (b) $c_2$, (c) $c_3$, (d) $c_4$, and (e) $k$.

For SBPSO with the Von Neumann topology, high $c_1$ values led to better performance: parameters in the range $c_1 \geq 0.8$ covered 20% of the parameter space but accounted for more than 46% of good parameter value combinations. Low $c_1$ values had few good parameter value combinations, especially those for $c_1 < 0.3$. For parameter $c_2$, the best results were found in the range $[0.5, 0.6)$, but all bins with $c_2 > 0.4$ scored comparably well. For $c_3$, the best parameter values were in the range $[1.5, 2.5)$, and performance worsened proportionally for parameter values further away from 2.0. For $c_4$ the two bins $[2.0, 2.5)$ and $[3.5, 4.0)$ clearly had the best results, while the values between 2.5 and 3.5 scored worse. Combining label A and label B, the values $c_4 < 1.5$ scored worse, but all values $c_4 \geq 1.5$ performed at least reasonably. For parameter $k$, high values led to a higher proportion of label A results, but all values $k \geq 6$ scored comparably. Combining label A and label B contributions, there was no significant difference between the performance of each of the nine values of $k$: any value of $k$ led to the same number of reasonable parameter value combinations.

### 6.5 Relative Importance of Control Parameters

In general, not all control parameters for SBPSO are expected to have the same impact on performance. For example, the conclusion in sections 6.2 to 6.4 for parameter $k$ was that very little difference was seen between values 1 through 9 with respect to reasonable-to-good performance. In contrast, for parameter $c_1$, values of 0.8 or higher clearly were an indication of better performance, while values of 0.3 or lower were detrimental. Therefore, the performance of the SBPSO algorithm on the MKP is more sensitive to parameter $c_1$, than to parameter $k$. This section contains a systematic investigation of the relative sensitivity of the five SBPSO parameters.

A measure of the distribution of performance of an individual parameter can serve as an indication of the sensitivity of SBPSO to that parameter. As argued in section 6.1, most information about the performance of an individual parameter can be gained from looking at "good" parameter value combinations only, where good was defined as the best 25% (label A) parameter value combinations. Therefore, for each individual parameter, the distribution of the label A combinations was used as a proxy for the distribution of the performance.

For each parameter, and each of the three topologies, the distribution of label A combinations across bins was converted to a single measurement using the following steps:

1. For each bin, the fraction of label A parameter value combinations was obtained, and the fractions themselves were ordered from high to low.
2. The sum of the *highest* five fractions was labeled $fraction_{\text{high}}$.
3. The sum of the *lowest* five fractions was labeled $fraction_{\text{low}}$.
4. The sensitivity score was then defined as the difference, $fraction_{\text{high}} - fraction_{\text{low}}$.

Note that for parameters $c_3$, $c_4$, and $k$, only nine bins were used, such that the bin ranked fifth was included in both $fraction_{\text{high}}$ and $fraction_{\text{low}}$ and drops out of the sensitivity score.

The sensitivity score ranges between 0% and 100%. A score of 0% means that all bins contained exactly the same fraction of label A combinations, indicating that good parameter value combinations show little to no sensitivity to the individual parameter. A score of 100% means that at least five bins contained *zero* label A combinations, but that these combinations are instead concentrated in the remaining bins. For this case, good parameter value combinations show a high sensitivity to the individual parameter.

Table 6 summarizes the resulting sensitivity score for each individual parameter, split by the topology used, and ranks the sensitivity scores of the five parameters for each topology.

**Table 6** Performance distribution per individual control parameter

| parameter | GB SBPSO | rank GB | LB SBPSO | rank LB | VN SBPSO | rank VN |
|---|---|---|---|---|---|---|
| $c_1$ | 58% | (1) | 52% | (2) | 48% | (2) |
| $c_2$ | 31% | (4) | 25% | (4) | 16% | (5) |
| $c_3$ | 53% | (2) | 62% | (1) | 65% | (1) |
| $c_4$ | 41% | (3) | 39% | (3) | 33% | (3) |
| $k$ | 20% | (5) | 22% | (5) | 25% | (4) |

The sensitivity scores indicated that the performance of SBPSO had the highest sensitivity to control parameters $c_1$ (attraction to the personal best) and $c_3$ (the maximum number of elements to add to the solution set randomly). Hence, it can be concluded that, when applying SBPSO to the MKP, these two parameters are the most important to be tuned well.

This result held for all three topologies investigated. All three topologies were the least sensitive to parameters $c_2$ (attraction to the neighborhood best) and $k$ (the size of the tournament used).

Note that an equal amount of tuning effort was expended on all five SBPSO parameters: the process described in section 5.2 meant finding the best out of 128 randomly chosen parameter value combinations spread evenly across the five dimensional parameter space.

## 7 Results of Experiments

This section describes the results of the experiments conducted using the tuned algorithm-topology pairs. Section 7.1 explains the statistical procedure used to compare the performance of the algorithm-topology pairs. Sections 7.2 and 7.3 discuss the results of the experiments on the small and large MKPs respectively.

For both the small MKPs and the large MKPs, the respective results sections each contain five tables comparing the performance of the algorithm-topology pairs: the first three tables each summarize and compare the performance of the four PSO algorithms using a single topology. The fourth table compares the results of each of the four PSO algorithms, across all of the topologies. The final table has more detailed results per problem and compares the four PSO algorithms using each algorithm's best performing topology.

### 7.1 Procedure for Statistical Comparison

The algorithm-topology pairs were compared for statistically significant differences in performance using the Iman-Davenport test (ID-test) (Iman and Davenport, 1980), which is a refinement of the better known Friedman test (Friedman, 1937). The ID-test was used to analyze the performance, measured as the average error[5] on each of the test problems, which is equivalent to using the actual objective function values. The null hypothesis of the ID-test was that all algorithm-topology pairs had the same median performance. The significance level $\alpha$ was chosen as 0.05.

In case the ID-test rejected the null-hypotheses and showed a significant difference in the performance of the algorithm-topology pairs, further post-hoc tests were performed in order to determine which of the algorithm-topology pairs outperformed the other pairs. The post-hoc test used was that proposed by Nemenyi (1963), which considers the differences in the average rank of the performance over all problems.

For the Nemenyi test, the $Z$-score (the normalized distance in average rank of the average error) was used as input:

$$Z = \frac{|\overline{R}_1 - \overline{R}_2|}{\sqrt{\frac{k\,(k+1)}{6\,N}}} \tag{27}$$

where $\overline{R}_i$ is the average rank of the average error for algorithm-topology pair $i$, $k$ is the total number of algorithm-topology pairs being compared, and $N$ is the number of test problems on which the pairs were compared. This standard normally distributed $Z$-score was then translated into a $p$-value.

[5] The error is defined as the deviation from the known optimum for the small MKPs, and as the deviation from the LP relaxation bound for the large MKPs.

Because the post-hoc tests involved multiple pair-wise comparisons, the significance level needed to be adjusted in order to maintain equal family-wise error rates. For this purpose the Holm-Bonferroni method (Holm, 1979) was used: the largest difference in average rank found in the Nemenyi test was compared at significance level $\alpha$, the second largest difference was compared at significance level $\alpha/2$, and the $k$-th largest difference was compared at significance level $\alpha/k$.

The $Z$-score, the associated $p$-value, and the Holm-adjusted $\alpha$ are provided in the bottom rows of each table in the following two sections. If a $p$-value is smaller than the Holm $\alpha$ mentioned below it, the algorithm-topology pair *underperformed* the best pair in the comparison by a statistically significant margin. For the best performing algorithm-topology pair, the average error score is shown in **bold**. If the ID-test indicated a statistically significant difference in performance, but the post-hoc tests did not indicate a single best pair, all algorithm-topology pairs that were indistinguishable from the best are shown in bold.

### 7.2 Small Multidimensional Knapsack Problems

Results for the 40 small MKP test problems are summarized in tables 7, 8, and 9 for the star, ring, and Von Neumann topologies respectively. Each table lists the average and standard deviation of the error (the best objective function value found compared to the known optimum), and the average rank of the errors. This is followed by the average and standard deviation of the success rate (shortened SR in the tables), and the average rank of the success rate. The success rate of an algorithm-topology pair on a single MKP was defined as the percentage of independent runs that were successful in finding the optimum. The next two rows in each table shed light on the consistency of the algorithm: the row labeled "# perfect" reports the number of problems for which all independent runs found the optimum, and the row labeled "# failure" reports the number of problems for which all independent runs failed to find the optimum.

For the algorithm-topology comparisons that are reported in each of the tables in this section, the ID-test indicated that the median performance showed statistically significant differences. Hence, in all five cases, post-hoc tests were conducted and the results are reported at the bottom of the respective tables.

Table 7 shows that the gbest SBPSO outperformed the other three algorithms with a star topology by a statistically significant margin. If success rate was used as the performance measure instead of average error, gbest SBPSO also performed best in a statistically significant manner ($p$-values and $\alpha$'s are not shown). The average success rate of gbest SBPSO was 82.5%, while the second best performer was gbest PBPSO with an average success rate of 51.4%.

For all 40 problems, the success rate for the gbest SBPSO exceeded or matched that of the other three gbest PSO algorithms. Gbest SBPSO was also more consistent than the other gbest PSO algorithms, as the optimum was found in all independent runs for 21 out of 40 problems. For the other three algorithms, the optimum was found in all independent runs for at most five problems.

Table 8 shows that the lbest SBPSO outperformed the other three algorithms with a ring topology by a statistically significant margin. If success rate was used as the performance measure instead of average error, lbest SBPSO also performed best in a statistically significant manner. The average success rate of LB SBPSO was 81.9%, while the second best performer was lbest PBPSO, scoring an average success rate of 63.4%.

**Table 7** Summary of small MKP test results for the star topology. Bold face indicates statistically significant outperformance.

| problem | GB BPSO error | GB BPSO (rank) | GB MBPSO error | GB MBPSO (rank) | GB PBPSO error | GB PBPSO (rank) | GB SBPSO error | GB SBPSO (rank) |
|---|---|---|---|---|---|---|---|---|
| average error | 1.117 % | ( 2.80 ) | 1.089 % | ( 3.56 ) | 0.628 % | ( 2.45 ) | **0.444 %** | ( 1.19 ) |
| stdev error | 1.913 % | | 1.592 % | | 1.625 % | | 1.640 % | |
| average SR | 42.8 % | ( 2.81 ) | 29.9 % | ( 3.38 ) | 51.4 % | ( 2.50 ) | 82.5 % | ( 1.31 ) |
| stdev SR | 41.3 % | | 34.7 % | | 35.1 % | | 31.8 % | |
| # perfect | 5 | ( 2.5 ) | 3 | ( 4 ) | 5 | ( 2.5 ) | 21 | ( 1 ) |
| # failure | 11 | ( 4 ) | 4 | ( 2 ) | 4 | ( 2 ) | 4 | ( 2 ) |
| $Z$-score | | 5.58 | | 8.21 | | 4.36 | | |
| $p$-value | | 0.0000 | | 0.0000 | | 0.0000 | | |
| Holm $\alpha$ | | 0.0250 | | 0.0500 | | 0.0167 | | |

**Table 8** Summary of small MKP test results for the ring topology. Bold face indicates statistically significant outperformance.

| problem | LB BPSO error | LB BPSO (rank) | LB MBPSO error | LB MBPSO (rank) | LB PBPSO error | LB PBPSO (rank) | LB SBPSO error | LB SBPSO (rank) |
|---|---|---|---|---|---|---|---|---|
| average error | 0.841 % | ( 2.95 ) | 0.639 % | ( 3.35 ) | 0.521 % | ( 2.31 ) | **0.440 %** | ( 1.39 ) |
| stdev error | 1.716 % | | 1.620 % | | 1.634 % | | 1.641 % | |
| average SR | 50.3 % | ( 2.93 ) | 45.7 % | ( 3.24 ) | 63.4 % | ( 2.28 ) | 81.9 % | ( 1.56 ) |
| stdev SR | 43.3 % | | 37.4 % | | 36.8 % | | 33.2 % | |
| # perfect | 7 | ( 3 ) | 4 | ( 4 ) | 12 | ( 2 ) | 23 | ( 1 ) |
| # failure | 10 | ( 4 ) | 4 | ( 2 ) | 4 | ( 2 ) | 4 | ( 2 ) |
| $Z$-score | | 5.40 | | 6.79 | | 3.19 | | |
| $p$-value | | 0.0000 | | 0.0000 | | 0.0007 | | |
| Holm $\alpha$ | | 0.0250 | | 0.0500 | | 0.0167 | | |

For 38 out of 40 problems, the success rate for the lbest SBPSO exceeded or matched that for the other three lbest PSO algorithms. Lbest SBPSO was also more consistent than the other local best PSO algorithms, as the optimum was found in all independent runs for 23 out of the 40 problems. For the other three algorithms, the optimum was found in all independent runs for at most 12 problems. Note that the number of problems solved perfectly by lbest PBPSO (that is, 12) is significantly higher than was the case for the gbest PBPSO (that is, five).

Table 9 shows that SBPSO with a Von Neumann topology outperformed the other three PSO algorithms by a statistically significant margin. If success rate was used as the performance measure instead of average error, SBPSO with a Von Neumann topology also performed best in a statistically significant manner. The average success rate of the Von Neumann SBPSO was 82.7%, while the second best performer was the Von Neumann PBPSO with an average success rate of 64.8%.

For 37 out of the 40 problems the success rate for SBPSO with the Von Neumann topology exceeded or matched that for the other three PSO algorithms. SBPSO was also more consistent than the other PSO algorithms using the Von Neumann topology, as the optimum was found in all independent runs for 23 out of 40 problems. For the other three algorithms, the optimum was found in all independent runs for at most 12 problems. The

**Table 9** Summary of small MKP test results for the Von Neumann topology. Bold face indicates statistically significant outperformance.

| | VN BPSO | | VN MBPSO | | VN PBPSO | | VN SBPSO | |
|---|---|---|---|---|---|---|---|---|
| problem | BPSO error | VN (rank) | MBPSO error | VN (rank) | PBPSO error | VN (rank) | SBPSO error | VN (rank) |
| average error | 0.609 % | ( 2.81 ) | 0.613 % | ( 3.45 ) | 0.510 % | ( 2.28 ) | **0.439 %** | ( 1.46 ) |
| stdev error | 1.635 % | | 1.623 % | | 1.633 % | | 1.641 % | |
| average SR | 56.6 % | ( 2.76 ) | 48.6 % | ( 3.31 ) | 64.8 % | ( 2.36 ) | 82.7 % | ( 1.56 ) |
| stdev SR | 41.1 % | | 35.3 % | | 36.8 % | | 32.6 % | |
| # perfect | 9 | ( 3 ) | 4 | ( 4 ) | 12 | ( 2 ) | 25 | ( 1 ) |
| # failure | 6 | ( 4 ) | 4 | ( 2 ) | 4 | ( 2 ) | 4 | ( 2 ) |
| $Z$-score | | 4.68 | | 6.89 | | 2.84 | | |
| $p$-value | | 0.0000 | | 0.0000 | | 0.0023 | | |
| Holm $\alpha$ | | 0.0250 | | 0.0500 | | 0.0167 | | |

number of problems solved perfectly by PSO algorithms using the Von Neumann topology closely matched the results for the corresponding lbest PSO algorithms, with only lbest BPSO (seven out of 40) scoring differently than BPSO with the Von Neumann topology (nine out of 40).

Table 10 compares the performance of the three algorithm-topology pairs for each PSO algorithm separately. For BPSO, MBPSO, and PBPSO, the ID-tests yielded a $p$-value less than 0.0001, indicating that a statistically significant difference in performance existed. For all three algorithms, it was the star topology that underperformed, while no statistically significant difference in performance was seen between the ring topology and the Von Neumann topology.

For BPSO, the difference in performance between the ring topology and the Von Neumann topology yielded a $p$-value of 0.1314 using the Nemenyi post-hoc test at a Holm $\alpha$ of 0.0250. Therefore, although Von Neumann BPSO performed best, the difference in error with lbest BPSO was not statistically significant. The Von Neumann BPSO also scored best on the average success rate, the number of problems solved perfectly, and the number of problems on which the algorithm failed.

For MBPSO, the difference in performance between the ring topology and the Von Neumann topology yielded a $p$-value of 0.1635 using the Nemenyi post-hoc test at a Holm $\alpha$ of 0.0250. Therefore, although Von Neumann MBPSO performed best, the difference in error with lbest MBPSO was not statistically significant. There was little difference in the number of problems which the MBPSO algorithm-topology pairs solved perfectly, and no difference at all in the number of problems on which they failed. With reference to success rate, gbest MBPSO clearly underperformed lbest MBPSO and Von Neumann MBPSO.

For PBPSO, the difference in performance between the ring topology and the Von Neumann topology yielded a $p$-value of 0.1515 using the Nemenyi post-hoc test at a Holm $\alpha$ of 0.0250. Therefore, although Von Neumann PBPSO performed best, the difference in error with lbest PBPSO was not statistically significant. In all listed measures, gbest PBPSO clearly underperformed, while there was very little difference between lbest PBPSO and the Von Neumann PBPSO, with tied scores in the number of perfectly solved problems as well as the number of problems on which they both failed.

For SBPSO, the ID-test yielded a $p$-value of 0.5134, which indicated that the null hypothesis of equal performance of gbest SBPSO, lbest SBPSO, and Von Neumann SBPSO

**Table 10** Summary of small MKP test results compared across topologies. Bold face indicates statistically significant outperformance.

| Measure | GB BPSO error | GB BPSO ( rank ) | LB BPSO error | LB BPSO ( rank ) | VN BPSO error | VN BPSO ( rank ) |
|---|---|---|---|---|---|---|
| avg error | 1.117 % | ( 2.65 ) | **0.841 %** | ( 1.80 ) | **0.609 %** | ( 1.55 ) |
| stdev error | 1.913 % | | 1.716 % | | 1.635 % | |
| average SR | 42.8 % | ( 2.45 ) | 50.3 % | ( 2.03 ) | 56.6 % | ( 1.53 ) |
| stdev SR | 41.3 % | | 43.3 % | | 41.1 % | |
| # perfect | 5 | ( 3 ) | 7 | ( 2 ) | 9 | ( 1 ) |
| # failure | 11 | ( 3 ) | 10 | ( 2 ) | 6 | ( 1 ) |
| *Z*-score | | 4.92 | | 1.12 | | |
| *p*-value | | 0.0000 | | 0.1314 | | |
| Holm $\alpha$ | | 0.0500 | | 0.0250 | | |

| Measure | GB MBPSO error | GB MBPSO ( rank ) | LB MBPSO error | LB MBPSO ( rank ) | VN MBPSO error | VN MBPSO ( rank ) |
|---|---|---|---|---|---|---|
| avg error | 1.089 % | ( 2.93 ) | **0.639 %** | ( 1.65 ) | **0.613 %** | ( 1.43 ) |
| stdev error | 1.592 % | | 1.620 % | | 1.623 % | |
| average SR | 29.9 % | ( 2.78 ) | 45.7 % | ( 1.68 ) | 48.6 % | ( 1.55 ) |
| stdev SR | 34.7 % | | 37.4 % | | 35.3 % | |
| # perfect | 3 | ( 3 ) | 4 | ( 1.5 ) | 4 | ( 1.5 ) |
| # failure | 4 | ( 2 ) | 4 | ( 2 ) | 4 | ( 2 ) |
| *Z*-score | | 6.71 | | 0.98 | | |
| *p*-value | | 0.0000 | | 0.1635 | | |
| Holm $\alpha$ | | 0.0500 | | 0.0250 | | |

| Measure | GB PBPSO error | GB PBPSO ( rank ) | LB PBPSO error | LB PBPSO ( rank ) | VN PBPSO error | VN PBPSO ( rank ) |
|---|---|---|---|---|---|---|
| avg error | 0.628 % | ( 2.68 ) | **0.521 %** | ( 1.78 ) | **0.510 %** | ( 1.55 ) |
| stdev error | 1.625 % | | 1.634 % | | 1.633 % | |
| average SR | 51.4 % | ( 2.4 ) | 63.4 % | ( 1.9 ) | 64.8 % | ( 1.7 ) |
| stdev SR | 35.1 % | | 36.8 % | | 36.8 % | |
| # perfect | 5 | ( 3 ) | 12 | ( 1.5 ) | 12 | ( 1.5 ) |
| # failure | 4 | ( 2 ) | 4 | ( 2 ) | 4 | ( 2 ) |
| *Z*-score | | 5.05 | | 1.03 | | |
| *p*-value | | 0.0000 | | 0.1515 | | |
| Holm $\alpha$ | | 0.0500 | | 0.0250 | | |

| Measure | GB SBPSO error | GB SBPSO (rank) | LB SBPSO error | LB SBPSO (rank) | VN SBPSO error | VN SBPSO (rank) |
|---|---|---|---|---|---|---|
| avg error | 0.444 % | ( 2.13 ) | 0.440 % | ( 2.01 ) | 0.439 % | ( 1.86 ) |
| stdev error | 1.640 % | | 1.641 % | | 1.641 % | |
| average SR | 82.5 % | ( 2.09 ) | 81.9 % | ( 2.04 ) | 82.7 % | ( 1.88 ) |
| stdev SR | 31.8 % | | 33.2 % | | 32.6 % | |
| # perfect | 21 | ( 3 ) | 23 | ( 2 ) | 25 | ( 1 ) |
| # failure | 4 | ( 2 ) | 4 | ( 2 ) | 4 | ( 2 ) |
| *Z*-score | | 1.21 | | 0.67 | | |
| *p*-value | | 0.1131 | | 0.2514 | | |
| Holm $\alpha$ | | 0.0500 | | 0.0250 | | |

was *not* rejected. Therefore, no statistically significant difference in performance could be found between the three topologies for SBPSO. The listed measures for SBPSO all indicated that there was little difference in performance between the three SBPSO algorithm-topology pairs: the relative difference in the average errors of the three pairs was 1.1%, while the relative difference in the average success rate of the three pairs was 1.0%. Only the number of problems solved perfectly showed some differentiation, as gbest SBPSO solved 21 out of the 40 problems perfectly, while lbest SBPSO completely solved 23 problems, and Von Neumann SBPSO 25 problems.

Table 11 shows a problem-by-problem comparison of the four PSO algorithms. Each PSO algorithm is paired with the topology that performed best for that PSO algorithm, which was the Von Neumann topology in each case. The statistical comparison of the four algorithm-topology pairs is therefore the same as that shown in table 9 and not repeated in table 11.

The four problems for which SBPSO with the Von Neumann topology failed to find the optimum in all independent runs are mknap2-6, mknap2-11, mknap2-13, and mknap2-18. The other 11 algorithm-topology pairs all similarly failed for these four problems. For the algorithm-topology pairs combining the Von Neumann topology with SBPSO, PBPSO, and MBPSO respectively, these four problems were also the only failures. For the Von Neumann-BPSO pair, additionally problems mknap2-43 and mknap2-47 caused failures.

Excluding the four problems on which SBPSO completely failed to find the optimum (a success rate of 0%), the *lowest success rate* recorded for SBPSO on any of the remaining 36 problems was reasonable: 50% for gbest SBPSO (average success rate on the 36 problems of 89.2%), 20% for lbest SBPSO (average success rate of 88.5%), and 30% for SBPSO using the Von Neumann topology (average success rate of 89.4%).

### 7.3 Large Multidimensional Knapsack Problems

Results for the 243 large MKPs are summarized in tables 12, 13, and 14 for the star, ring, and Von Neumann topology respectively. Each table lists the average and standard deviation of the error (the best objective function value found compared to the LP relaxation bound), and the average rank of the errors. The average error is shown on three different cross-sections of the problem set (refer to section 4 for details on these parameters and the problem set):

1. The number of items, $n$, with values 100, 250, and 500.
2. The number of constraints, $m$, with values 5, 10, and 30.
3. The tightness ratio, $r$, with values 0.25, 0.50, and 0.75.

The ID-test indicated that, for the algorithm-topology comparisons that are reported in each of the tables, the median performance showed statistically significant differences. Hence, for all five cases post-hoc tests were conducted and the results are reported at the bottom of the respective tables.

Table 12 summarizes the large MKP results for the four PSO algorithms, each using the star topology. The table shows that the gbest SBPSO was the best performing algorithm: it scored the smallest average error of 1.74%, and the average rank of the error shown on the same line was exactly 1, meaning that gbest SBPSO was the best performing algorithm on each of the 243 test problems. The post-hoc tests showed that the outperformance of gbest SBPSO was also statistically significant: pair-wise comparisons with the three other PSO algorithms yielded $Z$-scores above 10, which resulted in $p$-values smaller than $10^{-22}$. Gbest PBPSO was the second best performer on 193 problems, gbest BPSO performed second

**Table 11** Small MKP test results for the best algorithm-topology pairs per algorithm. Bold face indicates statistically significant outperformance.

| problem | $n$ | $m$ | VN BPSO error | VN BPSO (rank) | VN MBPSO error | VN MBPSO (rank) | VN PBPSO error | VN PBPSO (rank) | VN SBPSO error | VN SBPSO (rank) |
|---|---|---|---|---|---|---|---|---|---|---|
| mknap1-1 | 6 | 10 | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) |
| mknap1-2 | 10 | 10 | 0.212 % | ( 4 ) | 0 % | ( 2 ) | 0 % | ( 2 ) | 0 % | ( 2 ) |
| mknap1-3 | 15 | 10 | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) |
| mknap1-6 | 39 | 5 | 0.093 % | ( 1.5 ) | 0.278 % | ( 4 ) | 0.093 % | ( 1.5 ) | 0.104 % | ( 3 ) |
| mknap1-7 | 50 | 5 | 0.235 % | ( 3 ) | 0.280 % | ( 4 ) | 0.097 % | ( 2 ) | 0.054 % | ( 1 ) |
| mknap2-1 | 60 | 30 | 0.087 % | ( 3 ) | 0.285 % | ( 4 ) | 0.030 % | ( 2 ) | 0 % | ( 1 ) |
| mknap2-3 | 28 | 2 | 0 % | ( 2 ) | 0.143 % | ( 4 ) | 0 % | ( 2 ) | 0 % | ( 2 ) |
| mknap2-5 | 28 | 2 | 0.308 % | ( 3 ) | 0.335 % | ( 4 ) | 0.095 % | ( 2 ) | 0.054 % | ( 1 ) |
| mknap2-6 | 28 | 2 | 3.943 % | ( 4 ) | 3.820 % | ( 3 ) | 3.792 % | ( 2 ) | 3.698 % | ( 1 ) |
| mknap2-7 | 28 | 2 | 0.647 % | ( 2 ) | 0.799 % | ( 4 ) | 0.689 % | ( 3 ) | 0.278 % | ( 1 ) |
| mknap2-8 | 28 | 2 | 0.359 % | ( 3 ) | 0.883 % | ( 4 ) | 0.268 % | ( 2 ) | 0.110 % | ( 1 ) |
| mknap2-9 | 105 | 2 | 0.206 % | ( 1 ) | 0.453 % | ( 4 ) | 0.225 % | ( 2 ) | 0.247 % | ( 3 ) |
| mknap2-11 | 30 | 5 | 0.399 % | ( 2 ) | 0.451 % | ( 4 ) | 0.402 % | ( 3 ) | 0.348 % | ( 1 ) |
| mknap2-12 | 30 | 5 | 0.157 % | ( 3 ) | 0.141 % | ( 2 ) | 0.198 % | ( 4 ) | 0 % | ( 1 ) |
| mknap2-13 | 30 | 5 | 3.209 % | ( 2.5 ) | 3.209 % | ( 2.5 ) | 3.209 % | ( 2.5 ) | 3.209 % | ( 2.5 ) |
| mknap2-14 | 30 | 5 | 0.038 % | ( 3 ) | 0.031 % | ( 2 ) | 0.116 % | ( 4 ) | 0 % | ( 1 ) |
| mknap2-16 | 40 | 5 | 0.066 % | ( 2 ) | 0.118 % | ( 4 ) | 0.072 % | ( 3 ) | 0 % | ( 1 ) |
| mknap2-18 | 40 | 5 | 9.409 % | ( 3 ) | 9.465 % | ( 4 ) | 9.408 % | ( 2 ) | 9.407 % | ( 1 ) |
| mknap2-19 | 40 | 5 | 0 % | ( 2 ) | 0.017 % | ( 4 ) | 0 % | ( 2 ) | 0 % | ( 2 ) |
| mknap2-21 | 50 | 5 | 0.002 % | ( 3 ) | 0.025 % | ( 4 ) | 0 % | ( 1.5 ) | 0 % | ( 1.5 ) |
| mknap2-22 | 50 | 5 | 0 % | ( 2 ) | 0.095 % | ( 4 ) | 0 % | ( 2 ) | 0 % | ( 2 ) |
| mknap2-23 | 50 | 5 | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) | 0 % | ( 2.5 ) |
| mknap2-24 | 60 | 5 | 0.009 % | ( 3 ) | 0.157 % | ( 4 ) | 0.006 % | ( 2 ) | 0 % | ( 1 ) |
| mknap2-25 | 60 | 5 | 0.005 % | ( 2 ) | 0.055 % | ( 4 ) | 0.006 % | ( 3 ) | 0 % | ( 1 ) |
| mknap2-27 | 60 | 5 | 0 % | ( 2 ) | 0.035 % | ( 4 ) | 0 % | ( 2 ) | 0 % | ( 2 ) |
| mknap2-29 | 70 | 5 | 0.007 % | ( 2 ) | 0.333 % | ( 4 ) | 0.012 % | ( 3 ) | 0 % | ( 1 ) |
| mknap2-30 | 70 | 5 | 0 % | ( 2 ) | 0.140 % | ( 4 ) | 0 % | ( 2 ) | 0 % | ( 2 ) |
| mknap2-31 | 70 | 5 | 0 % | ( 2 ) | 0.125 % | ( 4 ) | 0 % | ( 2 ) | 0 % | ( 2 ) |
| mknap2-32 | 80 | 5 | 0.030 % | ( 3 ) | 0.133 % | ( 4 ) | 0 % | ( 1.5 ) | 0 % | ( 1.5 ) |
| mknap2-34 | 80 | 5 | 0.002 % | ( 3 ) | 0.043 % | ( 4 ) | 0 % | ( 1.5 ) | 0 % | ( 1.5 ) |
| mknap2-35 | 80 | 5 | 0.144 % | ( 3 ) | 0.263 % | ( 4 ) | 0.029 % | ( 2 ) | 0 % | ( 1 ) |
| mknap2-36 | 90 | 5 | 0.069 % | ( 3 ) | 0.125 % | ( 4 ) | 0.022 % | ( 2 ) | 0 % | ( 1 ) |
| mknap2-37 | 90 | 5 | 0.165 % | ( 3 ) | 0.279 % | ( 4 ) | 0.048 % | ( 2 ) | 0 % | ( 1 ) |
| mknap2-38 | 90 | 5 | 0.452 % | ( 4 ) | 0.420 % | ( 3 ) | 0.149 % | ( 2 ) | 0.008 % | ( 1 ) |
| mknap2-40 | 90 | 5 | 0.351 % | ( 4 ) | 0.208 % | ( 3 ) | 0.180 % | ( 2 ) | 0.001 % | ( 1 ) |
| mknap2-42 | 34 | 4 | 0.280 % | ( 4 ) | 0.201 % | ( 3 ) | 0.126 % | ( 2 ) | 0.010 % | ( 1 ) |
| mknap2-43 | 29 | 2 | 0.883 % | ( 4 ) | 0.340 % | ( 3 ) | 0.311 % | ( 2 ) | 0.017 % | ( 1 ) |
| mknap2-44 | 20 | 10 | 1.183 % | ( 4 ) | 0.359 % | ( 3 ) | 0.248 % | ( 2 ) | 0 % | ( 1 ) |
| mknap2-46 | 37 | 30 | 0.985 % | ( 4 ) | 0.290 % | ( 2 ) | 0.351 % | ( 3 ) | 0 % | ( 1 ) |
| mknap2-47 | 28 | 4 | 0.437 % | ( 4 ) | 0.181 % | ( 2 ) | 0.205 % | ( 3 ) | 0.002 % | ( 1 ) |
| average | | | 0.609 % | ( 2.81 ) | 0.613 % | ( 3.45 ) | 0.510 % | ( 2.28 ) | **0.439 %** | ( 1.46 ) |
| # perfect | | | 9 | ( 3 ) | 4 | ( 4 ) | 12 | ( 2 ) | 25 | ( 1 ) |
| # failure | | | 6 | ( 4 ) | 4 | ( 2 ) | 4 | ( 2 ) | 4 | ( 2 ) |

best for the remaining 50 problems, and gbest MBPSO usually ranked last out of the four algorithm-topology pairs.

The relative performance of the four PSO algorithms using the star topology was stable across each of the three splits of the problem set, with gbest SBPSO $>$ gbest PBPSO $>$ gbest BPSO $>$ gbest MBPSO in each individual split except one: for the $243/3 = 81$ problems

**Table 12** Summary of large MKP test results for the star topology. Bold face indicates statistically significant outperformance.

| Measure | | GB BPSO error | ( rank ) | GB MBPSO error | ( rank ) | GB PBPSO error | ( rank ) | GB SBPSO error | ( rank ) |
|---|---|---|---|---|---|---|---|---|---|
| average error | | 4.679 % | ( 2.909 ) | 5.619 % | ( 3.885 ) | 3.250 % | ( 2.206 ) | **1.740 %** | ( 1.000 ) |
| stdev error | | 3.468 % | | 2.723 % | | 1.718 % | | 1.170 % | |
| n | 100 | 3.831 % | ( 2.877 ) | 5.160 % | ( 3.889 ) | 2.568 % | ( 2.235 ) | **1.260 %** | ( 1.000 ) |
| n | 250 | 4.679 % | ( 2.877 ) | 5.663 % | ( 3.889 ) | 3.286 % | ( 2.235 ) | **1.758 %** | ( 1.000 ) |
| n | 500 | 5.526 % | ( 2.975 ) | 6.034 % | ( 3.877 ) | 3.896 % | ( 2.148 ) | **2.201 %** | ( 1.000 ) |
| m | 5 | 3.037 % | ( 2.383 ) | 4.354 % | ( 4.000 ) | 3.134 % | ( 2.617 ) | **1.875 %** | ( 1.000 ) |
| m | 10 | 3.942 % | ( 3.012 ) | 5.521 % | ( 3.988 ) | 2.763 % | ( 2.000 ) | **1.553 %** | ( 1.000 ) |
| m | 30 | 7.057 % | ( 3.333 ) | 6.983 % | ( 3.667 ) | 3.853 % | ( 2.000 ) | **1.791 %** | ( 1.000 ) |
| $r$ | 0.25 | 8.253 % | ( 3.122 ) | 8.664 % | ( 3.659 ) | 5.264 % | ( 2.220 ) | **3.141 %** | ( 1.000 ) |
| $r$ | 0.50 | 3.751 % | ( 2.831 ) | 5.344 % | ( 4.000 ) | 2.799 % | ( 2.169 ) | **1.355 %** | ( 1.000 ) |
| $r$ | 0.75 | 1.909 % | ( 2.769 ) | 2.712 % | ( 4.000 ) | 1.613 % | ( 2.231 ) | **0.676 %** | ( 1.000 ) |
| $Z$-score | | | 16.30 | | 24.63 | | 10.30 | | |
| $p$-value | | | 0.0000 | | 0.0000 | | 0.0000 | | |
| Holm $\alpha$ | | | 0.0250 | | 0.0500 | | 0.0167 | | |

with $m = 5$, gbest BPSO (average rank 2.383) scored better than gbest PBPSO (average rank 2.617). Here the symbol '$>$' is used to mean "has a lower (better) average rank than".

A difference in performance was seen with regards to the split of the problems based on the number of items, $n$: a larger number of items led to a higher average error for each of the gbest PSO algorithms. However, this effect was not equally strong for each of the algorithms: for problems with $n = 500$ compared to those with $n = 100$, the average error of gbest SBPSO was 75% higher, while for gbest MBPSO the increase in average error was only 16%.

Problems with tightness ratio $r = 0.25$ were most challenging for all gbest PSO algorithms, with the average error substantially higher than for problems with $r = 0.50$ or 0.75. A smaller $r$ means that each of the $m$ weight constraints is more restrictive (lower capacity), which, *in general*, has two effects on the optimal solution compared to that for problems with a higher tightness ratio:

1. the optimal solution using a small $r$ contains fewer items, and
2. the objective function value at the optimum using a small $r$ is lower, as fewer items are included in the knapsack.

Table 13 summarizes the large MKP results for the four PSO algorithms, each using the ring topology. The table shows that the lbest SBPSO was the best performing algorithm with an average rank of 1.333. The ID-test and post-hoc tests confirmed that lbest SBPSO outperformed each of the other three pairs, but the difference in performance between lbest SBPSO and lbest PBPSO was smaller than that seen between gbest SBPSO and gbest PBPSO in table 12.

The relative performance of the four PSO algorithms using the ring topology was stable across each of the three splits of the problem set into three subsets, with lbest SBPSO $>$ lbest PBPSO $>$ lbest MBPSO $>$ lbest BPSO, except for two cases:

1. for the problems with $m = 5$, lbest PBPSO (average rank 1.000) scored better than lbest SBPSO (average rank 2.000) on all 81 problems in the subset, while lbest BPSO (average rank 3.210) scored better than lbest MBPSO (average rank 3.790), and

**Table 13** Summary of large MKP test results for the ring topology. Bold face indicates statistically significant outperformance.

| Measure | | LB BPSO error | ( rank ) | LB MBPSO error | ( rank ) | LB PBPSO error | ( rank ) | LB SBPSO error | ( rank ) |
|---|---|---|---|---|---|---|---|---|---|
| average error | | 7.006 % | ( 3.737 ) | 3.922 % | ( 2.959 ) | 3.650 % | ( 1.971 ) | **2.292 %** | ( 1.333 ) |
| stdev error | | 5.037 % | | 2.059 % | | 2.591 % | | 1.331 % | |
| n | 100 | 6.348 % | ( 3.778 ) | 3.044 % | ( 2.852 ) | 3.101 % | ( 2.037 ) | **1.767 %** | ( 1.333 ) |
| n | 250 | 6.951 % | ( 3.753 ) | 3.917 % | ( 2.938 ) | 3.626 % | ( 1.975 ) | **2.366 %** | ( 1.333 ) |
| n | 500 | 7.719 % | ( 3.679 ) | 4.805 % | ( 3.086 ) | 4.221 % | ( 1.901 ) | **2.743 %** | ( 1.333 ) |
| m | 5 | 3.091 % | ( 3.210 ) | 3.289 % | ( 3.790 ) | **1.994 %** | ( 1.000 ) | 2.334 % | ( 2.000 ) |
| m | 10 | 7.520 % | ( 4.000 ) | 3.654 % | ( 2.963 ) | 3.112 % | ( 2.037 ) | **2.075 %** | ( 1.000 ) |
| m | 30 | 10.407 % | ( 4.000 ) | 4.824 % | ( 2.123 ) | 5.842 % | ( 2.877 ) | **2.468 %** | ( 1.000 ) |
| $r$ | 0.25 | 11.961 % | ( 3.829 ) | 6.185 % | ( 2.817 ) | 6.059 % | ( 2.024 ) | **3.893 %** | ( 1.329 ) |
| $r$ | 0.50 | 5.817 % | ( 3.723 ) | 3.608 % | ( 3.060 ) | 3.066 % | ( 1.892 ) | **1.957 %** | ( 1.325 ) |
| $r$ | 0.75 | 3.063 % | ( 3.654 ) | 1.878 % | ( 3.000 ) | 1.738 % | ( 2.000 ) | **0.966 %** | ( 1.346 ) |
| $Z$-score | | | 20.53 | | 13.88 | | 5.45 | | |
| $p$-value | | | 0.0000 | | 0.0000 | | 0.0000 | | |
| Holm $\alpha$ | | | 0.0500 | | 0.0250 | | 0.0167 | | |

2. for the problems with $m = 30$, lbest MBPSO (average rank 2.123) scored better than lbest PBPSO (average rank 2.877).

The relative performance of the lbest MBPSO and lbest PBPSO algorithm-pairs was correlated with the number of constraints, $m$: lbest MBPSO performed relatively better for an increasing number of constraints, while lbest PBPSO performed relatively worse with increasing $m$. For both lbest PBPSO and lbest MBPSO the average error increased when $m$ increased, but for lbest PBPSO this deterioration was worse. For all the lbest PSO algorithms, the average error was most sensitive to changes in $r$.

A possible explanation for lbest PBPSO having outperformed lbest SBPSO on problems with $m = 5$, is that the lbest SBPSO algorithm was better tuned to the problems with a larger number of constraints ($m = 10$ or 30), while the lbest PBPSO algorithm was better tuned for problems with fewer constraints. An alternative explanation is that the $k$-tournament selection used in LB SBPSO helped the particles stay in the feasible part of the solution space. This feature has extra value in the case of a larger number of constraints, where particles will encounter the edge of the feasible part of the solution space more often.

Table 14 shows that the Von Neumann SBPSO was the best performing algorithm with an average rank of 1.342. The ID-test and post-hoc tests confirmed that the Von Neumann SBPSO outperformed each of the other three pairs, with the Von Neumann PBPSO scoring second best. The difference in performance between the Von Neumann SBPSO and the Von Neumann PBPSO was approximately the same as seen between lbest SBPSO and lbest PBPSO in table 13.

The relative behavior of the four PSO algorithms using the Von Neumann topology was the same as that seen for the lbest PSO algorithms in table 13: across each of the three splits of the problem set, the result was Von Neumann SBPSO $>$ Von Neumann PBPSO $>$ Von Neumann MBPSO $>$ Von Neumann BPSO in each individual split, except for two cases:

1. for the problems with $m = 5$, the Von Neumann PBPSO (average rank 1.000) performed best on all 81 problems in the subset, with the Von Neumann SBPSO (average rank

**Table 14** Summary of large MKP test results for the von Neumann topology. Bold face indicates statistically significant outperformance.

| Measure | | VN BPSO error | VN BPSO ( rank ) | VN MBPSO error | VN MBPSO ( rank ) | VN PBPSO error | VN PBPSO ( rank ) | VN SBPSO error | VN SBPSO ( rank ) |
|---|---|---|---|---|---|---|---|---|---|
| average error | | 6.973 % | ( 3.823 ) | 3.403 % | ( 2.811 ) | 3.348 % | ( 2.025 ) | **2.249 %** | ( 1.342 ) |
| stdev error | | 5.039 % | | 1.742 % | | 2.533 % | | 1.275 % | |
| n | 100 | 6.291 % | ( 3.815 ) | 2.647 % | ( 2.790 ) | 2.762 % | ( 2.049 ) | **1.772 %** | ( 1.346 ) |
| n | 250 | 6.920 % | ( 3.864 ) | 3.418 % | ( 2.765 ) | 3.330 % | ( 2.037 ) | **2.294 %** | ( 1.333 ) |
| n | 500 | 7.707 % | ( 3.790 ) | 4.145 % | ( 2.877 ) | 3.954 % | ( 1.988 ) | **2.680 %** | ( 1.346 ) |
| m | 5 | 3.076 % | ( 3.469 ) | 2.980 % | ( 3.506 ) | **1.783 %** | ( 1.000 ) | 2.433 % | ( 2.025 ) |
| m | 10 | 7.465 % | ( 4.000 ) | 3.191 % | ( 2.914 ) | 2.847 % | ( 2.086 ) | **2.046 %** | ( 1.000 ) |
| m | 30 | 10.377 % | ( 4.000 ) | 4.039 % | ( 2.012 ) | 5.416 % | ( 2.988 ) | **2.266 %** | ( 1.000 ) |
| $r$ | 0.25 | 11.943 % | ( 3.976 ) | 5.382 % | ( 2.610 ) | 5.739 % | ( 2.085 ) | **3.789 %** | ( 1.329 ) |
| $r$ | 0.50 | 5.775 % | ( 3.855 ) | 3.089 % | ( 2.819 ) | 2.693 % | ( 2.000 ) | **1.917 %** | ( 1.325 ) |
| $r$ | 0.75 | 3.023 % | ( 3.628 ) | 1.658 % | ( 3.013 ) | 1.533 % | ( 1.987 ) | **0.981 %** | ( 1.372 ) |
| Z-score | | | 21.18 | | 12.54 | | 5.83 | | |
| P-value | | | 0.0000 | | 0.0000 | | 0.0000 | | |
| Holm $\alpha$ | | | 0.0500 | | 0.0250 | | 0.0167 | | |

2.025) scoring second best. Also the Von Neumann BPSO (average rank 3.469) narrowly outperformed the Von Neumann MBPSO (average rank 3.506), and

2. for the problems with $m = 30$, the Von Neumann MBPSO (average rank 2.012) scored better than the Von Neumann PBPSO (average rank 2.988).

Table 15 compares the performance of the three topologies for each algorithm over all the large MKPs. For each PSO algorithm, the ID-tests yielded a $p$-value below 0.0001, indicating that a statistically significant difference in performance existed between the three topologies. Due to space restrictions, the $Z$-scores, $p$-values, and Holm $\alpha$'s for the Nemenyi post-hoc tests have been excluded from table 15. However, the largest of these $p$-values was 0.0011 with a Holm $\alpha$ of 0.0167. Therefore, it was confirmed that, for each PSO algorithm, a single topology performed best by a statistically significant margin: for MBPSO and PBPSO the Von Neumann topology scored best, while for BPSO and SBPSO it was the star topology that scored best.

For BPSO, the gbest BPSO performed much better than BPSO using either of the other two topologies. The average error was 4.68% for gbest BPSO, with lbest BPSO and the Von Neumann BPSO scoring 7.01% and 6.97% respectively. The gbest BPSO scored best on 198 out of 243 problems, but was outperformed on problems with few constraints ($m = 5$) combined with a high tightness ratio of $r = 0.75$. Here gbest BPSO performed worst out of the three BPSO pairs on the entire subset of 27 problems. For problems with $m = 5$ and $r = 0.5$, gbest BPSO's performance was comparable to the other two pairs and yielded an average rank of 1.944.

For MBPSO, the relative performance of the three topologies was very stable across the entire problem set with the Von Neumann MBPSO scoring the best (with an average rank of 1.010), lbest MBPSO achieved an average rank of 1.990, and gbest MBPSO scored worst on all problems. The Von Neumann MBPSO failed to outperform lbest MBPSO on only three of the 243 problems.

For the PBPSO, the Von Neumann PBPSO performed best with reference to the average rank of errors, with an average rank of 1.5. However, gbest PBPSO achieved a lower average

**Table 15** Summary of large MKP test results compared across topologies. Bold face indicates statistically significant outperformance.

| | | GB BPSO | | LB BPSO | | VN BPSO | |
|---|---|---|---|---|---|---|---|
| Measure | | error | ( rank ) | error | ( rank ) | error | ( rank ) |
| average error | | **4.679 %** | ( 1.340 ) | 7.006 % | ( 2.510 ) | 6.973 % | ( 2.150 ) |
| stdev error | | 3.468 % | | 5.000 % | | 5.000 % | |
| n | 100 | **3.831 %** | ( 1.296 ) | 6.348 % | ( 2.537 ) | 6.291 % | ( 2.167 ) |
| n | 250 | **4.679 %** | ( 1.327 ) | 6.951 % | ( 2.531 ) | 6.920 % | ( 2.142 ) |
| n | 500 | **5.526 %** | ( 1.395 ) | 7.719 % | ( 2.451 ) | 7.707 % | ( 2.154 ) |
| m | 5 | **3.037 %** | ( 2.019 ) | 3.091 % | ( 2.179 ) | **3.076 %** | ( 1.802 ) |
| m | 10 | **3.942 %** | ( 1.000 ) | 7.520 % | ( 2.704 ) | 7.465 % | ( 2.296 ) |
| m | 30 | **7.057 %** | ( 1.000 ) | 10.407 % | ( 2.636 ) | 10.377 % | ( 2.364 ) |
| *r* | 0.25 | **8.253 %** | ( 1.037 ) | 11.961 % | ( 2.555 ) | 11.943 % | ( 2.409 ) |
| *r* | 0.50 | **3.751 %** | ( 1.307 ) | 5.817 % | ( 2.530 ) | 5.775 % | ( 2.163 ) |
| *r* | 0.75 | **1.909 %** | ( 1.692 ) | 3.063 % | ( 2.429 ) | **3.023 %** | ( 1.878 ) |

| | | GB MBPSO | | LB MBPSO | | VN MBPSO | |
|---|---|---|---|---|---|---|---|
| Measure | | error | ( rank ) | error | ( rank ) | error | ( rank ) |
| average error | | 5.619 % | ( 3.000 ) | 3.922 % | ( 1.990 ) | **3.403 %** | ( 1.010 ) |
| stdev error | | 2.723 % | | 2.100 % | | 1.700 % | |
| n | 100 | 5.160 % | ( 3.000 ) | 3.044 % | ( 1.988 ) | **2.647 %** | ( 1.012 ) |
| n | 250 | 5.663 % | ( 3.000 ) | 3.917 % | ( 2.000 ) | **3.418 %** | ( 1.000 ) |
| n | 500 | 6.034 % | ( 3.000 ) | 4.805 % | ( 1.975 ) | **4.145 %** | ( 1.025 ) |
| m | 5 | 4.354 % | ( 3.000 ) | 3.289 % | ( 1.963 ) | **2.980 %** | ( 1.037 ) |
| m | 10 | 5.521 % | ( 3.000 ) | 3.654 % | ( 2.000 ) | **3.191 %** | ( 1.000 ) |
| m | 30 | 6.983 % | ( 3.000 ) | 4.824 % | ( 2.000 ) | **4.039 %** | ( 1.000 ) |
| *r* | 0.25 | 8.664 % | ( 3.000 ) | 6.185 % | ( 2.000 ) | **5.382 %** | ( 1.000 ) |
| *r* | 0.50 | 5.344 % | ( 3.000 ) | 3.608 % | ( 1.988 ) | **3.089 %** | ( 1.012 ) |
| *r* | 0.75 | 2.712 % | ( 3.000 ) | 1.878 % | ( 1.974 ) | **1.658 %** | ( 1.026 ) |

| | | GB PBPSO | | LB PBPSO | | VN PBPSO | |
|---|---|---|---|---|---|---|---|
| Measure | | error | ( rank ) | error | ( rank ) | error | ( rank ) |
| average error | | 3.250 % | ( 1.860 ) | 3.650 % | ( 2.650 ) | **3.348 %** | ( 1.500 ) |
| stdev error | | 1.718 % | | 2.600 % | | 2.500 % | |
| n | 100 | **2.568 %** | ( 1.790 ) | 3.101 % | ( 2.667 ) | **2.762 %** | ( 1.543 ) |
| n | 250 | 3.286 % | ( 1.864 ) | 3.626 % | ( 2.654 ) | **3.330 %** | ( 1.481 ) |
| n | 500 | 3.896 % | ( 1.914 ) | 4.221 % | ( 2.617 ) | **3.954 %** | ( 1.469 ) |
| m | 5 | 3.134 % | ( 3.000 ) | 1.994 % | ( 2.000 ) | **1.783 %** | ( 1.000 ) |
| m | 10 | **2.763 %** | ( 1.568 ) | 3.112 % | ( 2.951 ) | **2.847 %** | ( 1.481 ) |
| m | 30 | **3.853 %** | ( 1.000 ) | 5.842 % | ( 2.988 ) | 5.416 % | ( 2.012 ) |
| *r* | 0.25 | **5.264 %** | ( 1.695 ) | 6.059 % | ( 2.646 ) | **5.739 %** | ( 1.659 ) |
| *r* | 0.50 | 2.799 % | ( 1.904 ) | 3.066 % | ( 2.663 ) | **2.693 %** | ( 1.434 ) |
| *r* | 0.75 | 1.613 % | ( 1.974 ) | 1.738 % | ( 2.628 ) | **1.533 %** | ( 1.397 ) |

| | | GB SBPSO | | LB SBPSO | | VN SBPSO | |
|---|---|---|---|---|---|---|---|
| Measure | | error | ( rank ) | error | ( rank ) | error | ( rank ) |
| average error | | **1.740 %** | ( 1.000 ) | 2.292 % | ( 2.570 ) | 2.249 % | ( 2.430 ) |
| stdev error | | 1.170 % | | 1.300 % | | 1.300 % | |
| n | 100 | **1.260 %** | ( 1.000 ) | 1.767 % | ( 2.519 ) | 1.772 % | ( 2.481 ) |
| n | 250 | **1.758 %** | ( 1.000 ) | 2.366 % | ( 2.636 ) | 2.294 % | ( 2.364 ) |
| n | 500 | **2.201 %** | ( 1.000 ) | 2.743 % | ( 2.543 ) | 2.680 % | ( 2.457 ) |
| m | 5 | **1.875 %** | ( 1.000 ) | 2.334 % | ( 2.173 ) | 2.433 % | ( 2.827 ) |
| m | 10 | **1.553 %** | ( 1.000 ) | 2.075 % | ( 2.549 ) | 2.046 % | ( 2.451 ) |
| m | 30 | **1.791 %** | ( 1.000 ) | 2.468 % | ( 2.975 ) | 2.266 % | ( 2.025 ) |
| *r* | 0.25 | **3.141 %** | ( 1.000 ) | 3.893 % | ( 2.659 ) | 3.789 % | ( 2.341 ) |
| *r* | 0.50 | **1.355 %** | ( 1.000 ) | 1.957 % | ( 2.566 ) | 1.917 % | ( 2.434 ) |
| *r* | 0.75 | **0.676 %** | ( 1.000 ) | 0.966 % | ( 2.468 ) | 0.981 % | ( 2.532 ) |

error, scoring 3.25% while the Von Neumann PBPSO had an average error of 3.35%. This can be explained by the more consistent behavior of gbest PBPSO: its standard deviation of the error was 1.72%, while for the Von Neumann PBPSO this was 2.5%. The Von Neumann PBPSO scored well for problems with $m = 5$, but scored badly for problems with $m = 30$: the difference in average error on the two subsets was $5.42\% - 1.78\% = 3.63\%$. For gbest PBPSO the sensitivity to the problem parameter $m$ was much smaller, and the difference between the subset on which it performed best ($m = 10$) and worst ($m = 30$) was only $3.85\% - 2.76\% = 1.09\%$.

For SBPSO, the star topology was most successful, with gbest SBPSO performing best on all 243 problems. Little difference in performance was observed between lbest SBPSO and the Von Neumann SBPSO, which is probably related to the fact that the same control parameter values were used for both pairs (refer to table 5 for the parameter values). Hence, the only difference between the pairs was that the Von Neumann SBPSO has a more closely connected swarm compared to lbest SBPSO. Only for the split of the problem set based on the number of constraints, $m$, some difference in performance was seen between lbest SBPSO and the Von Neumann SBPSO, where lbest SBPSO performed better on problems with $m = 5$, and the Von Neumann PBPSO performed better on problems with $m = 30$. Considering the number of constraints, both lbest SBPSO and the Von Neumann SBPSO performed best on the subset of problems with $m = 10$. Having a more closely connected swarm helped the Von Neumann SBPSO on problems with more constraints.

A detailed comparison of the four PSO algorithms, each using its best performing topology, is given in table 16. The four best performing algorithm-topology pairs are gbest BPSO, Von Neumann MBPSO, Von Neumann PBPSO, and gbest SBPSO. With an average error of 1.72%, gbest SBPSO scored better than the other three pairs, with the second best pair, Von Neumann PBPSO, scoring an average error of 3.32 %. The ID-test followed by post-hoc tests indicated that gbest SBPSO outperformed the other three pairs by a statistically significant margin.

For gbest SBPSO, the average rank was 1.26, followed by Von Neumann PBPSO, Von Neumann MBPSO, and gbest BPSO with average ranks of 2.13, 2.81, and 3.80 respectively. The gbest SBPSO had the lowest error on 179 of the 243 problems, and was second best on the remaining 64, for which the Von Neumann PBPSO scored best each time. Gbest BPSO performed worst on 194 problems, and the second worst on the remaining 47.

Each of the first 27 rows of table 16 represents results for the subset of nine problems that correspond to the given MKP parameters $n, m$, and $r$. For all 27 problem subsets, the ID-test indicated a difference in performance across the four algorithm-topology pairs. However, in only two cases was a single algorithm-topology pair shown to outperform the other three[6]: Gbest SBPSO statistically outperformed for $n = 100$, $m = 10$, $r = 0.25$ and $n = 250$, $m = 10$, $r = 0.25$. For each of the remaining 25 problems, the post-hoc tests did not indicate a single best algorithm-topology pair, but instead resulted in two best pairs with indistinguishable performance: no significant difference could be seen between the two best performing pairs, while the two worst pairs underperformed the best two in a statistically significant manner. For nine out of 25 problem specifications, all with $m = 30$, gbest SBPSO and the Von Neumann MBPSO performed best, while gbest BPSO and the Von Neumann PBPSO underperformed. For the remaining 16 out of 25 cases, gbest SBPSO and the Von Neumann PBPSO performed best, while gbest BPSO and the Von Neumann MBPSO underperformed.

[6] Even if all problems yield the same ranks, resulting in average rankings of 1, 2, 3, and 4 for the four algorithm-topology pairs, the post-hoc Nemenyi test did *not* show a statistically significant difference between ranks 1 and 2, at a confidence level of $\alpha = 0.05$, which led to a Holm $\alpha$ of 0.0167 for the comparison of the two best performing pairs.

**Table 16** Large MKP results for the best algorithm-topology pairs per algorithm. Bold face indicates statistically significant outperformance.

| | | | GB BPSO | | VN MBPSO | | VN PBPSO | | GB SBPSO | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $m$ | $r$ | error | ( rank ) | error | ( rank ) | error | ( rank ) | error | ( rank ) |
| 100 | 5 | 0.25 | 3.772 % | ( 3.67 ) | 3.625 % | ( 3.33 ) | **1.757 %** | ( 1 ) | **1.951 %** | ( 2 ) |
| 100 | 5 | 0.50 | 1.835 % | ( 3.56 ) | 1.774 % | ( 3.44 ) | **0.796 %** | ( 1 ) | **0.873 %** | ( 2 ) |
| 100 | 5 | 0.75 | 1.155 % | ( 3.89 ) | 1.077 % | ( 3.11 ) | **0.504 %** | ( 1.56 ) | **0.505 %** | ( 1.44 ) |
| 100 | 10 | 0.25 | 5.334 % | ( 4 ) | 4.064 % | ( 2.56 ) | 4.052 % | ( 2.44 ) | **2.181 %** | ( 1 ) |
| 100 | 10 | 0.50 | 2.384 % | ( 4 ) | 2.160 % | ( 3 ) | **1.868 %** | ( 2 ) | **0.853 %** | ( 1 ) |
| 100 | 10 | 0.75 | 1.217 % | ( 3.56 ) | 1.206 % | ( 3.44 ) | **1.031 %** | ( 2 ) | **0.399 %** | ( 1 ) |
| 100 | 30 | 0.25 | 11.396 % | ( 4 ) | **5.132 %** | ( 2 ) | 8.411 % | ( 3 ) | **2.773 %** | ( 1 ) |
| 100 | 30 | 0.50 | 4.786 % | ( 4 ) | **3.057 %** | ( 2 ) | 3.902 % | ( 3 ) | **1.195 %** | ( 1 ) |
| 100 | 30 | 0.75 | 2.327 % | ( 3.13 ) | **1.566 %** | ( 2 ) | 2.364 % | ( 3.88 ) | **0.540 %** | ( 1 ) |
| 250 | 5 | 0.25 | 5.182 % | ( 3.67 ) | 4.995 % | ( 3.33 ) | **3.066 %** | ( 1 ) | **3.294 %** | ( 2 ) |
| 250 | 5 | 0.50 | 2.680 % | ( 3.67 ) | 2.617 % | ( 3.33 ) | **1.437 %** | ( 1 ) | **1.541 %** | ( 2 ) |
| 250 | 5 | 0.75 | 1.390 % | ( 3.67 ) | 1.362 % | ( 3.33 ) | **0.740 %** | ( 1.56 ) | **0.735 %** | ( 1.44 ) |
| 250 | 10 | 0.25 | 6.917 % | ( 4 ) | 4.895 % | ( 2.67 ) | 4.801 % | ( 2.33 ) | **2.845 %** | ( 1 ) |
| 250 | 10 | 0.50 | 3.161 % | ( 3.95 ) | 2.904 % | ( 3.05 ) | **2.312 %** | ( 2 ) | **1.186 %** | ( 1 ) |
| 250 | 10 | 0.75 | 1.546 % | ( 3.75 ) | 1.522 % | ( 3.25 ) | **1.237 %** | ( 2 ) | **0.561 %** | ( 1 ) |
| 250 | 30 | 0.25 | 12.765 % | ( 4 ) | **6.333 %** | ( 2 ) | 9.356 % | ( 3 ) | **3.451 %** | ( 1 ) |
| 250 | 30 | 0.50 | 5.556 % | ( 4 ) | **3.921 %** | ( 2 ) | 4.308 % | ( 3 ) | **1.471 %** | ( 1 ) |
| 250 | 30 | 0.75 | 2.738 % | ( 4 ) | **2.055 %** | ( 2 ) | 2.595 % | ( 3 ) | **0.670 %** | ( 1 ) |
| 500 | 5 | 0.25 | 6.349 % | ( 3.67 ) | 6.299 % | ( 3.33 ) | **4.565 %** | ( 1 ) | **4.749 %** | ( 2 ) |
| 500 | 5 | 0.50 | 3.134 % | ( 3.33 ) | 3.219 % | ( 3.67 ) | **2.026 %** | ( 1.44 ) | **2.066 %** | ( 1.56 ) |
| 500 | 5 | 0.75 | 1.837 % | ( 3.5 ) | 1.849 % | ( 3.5 | **1.157 %** | ( 1.33 ) | **1.160 %** | ( 1.67 ) |
| 500 | 10 | 0.25 | 8.357 % | ( 4 ) | 5.943 % | ( 3 ) | **5.652 %** | ( 2 ) | **3.404 %** | ( 1 ) |
| 500 | 10 | 0.50 | 3.776 % | ( 4 ) | 3.522 % | ( 3 ) | **2.677 %** | ( 2 ) | **1.455 %** | ( 1 ) |
| 500 | 10 | 0.75 | 1.889 % | ( 3.5 ) | 1.905 % | ( 3.5 | **1.399 %** | ( 2 ) | **0.726 %** | ( 1 ) |
| 500 | 30 | 0.25 | 14.189 % | ( 4 ) | **7.089 %** | ( 2 ) | 10.003 % | ( 3 ) | **3.592 %** | ( 1 ) |
| 500 | 30 | 0.50 | 6.398 % | ( 4 ) | **4.655 %** | ( 2.11 ) | 4.822 % | ( 2.89 ) | **1.593 %** | ( 1 ) |
| 500 | 30 | 0.75 | 3.083 % | ( 4 ) | **2.378 %** | ( 2 ) | 2.811 % | ( 3 ) | **0.764 %** | ( 1 ) |
| average | | | 4.635 % | ( 3.80 ) | 3.375 % | ( 2.81 ) | 3.320 % | ( 2.13 ) | **1.723 %** | ( 1.26 ) |
| $Z$-score | | | | 21.68 | | 13.22 | | 7.34 | | |
| $p$-value | | | | 0.0000 | | 0.0000 | | 0.0000 | | |
| Holm $\alpha$ | | | | 0.0500 | | 0.0250 | | 0.0167 | | |

The performance of the Von Neumann PBPSO deteriorated for larger values of $m$, compared to the other algorithm-topology pairs in table 16. The Von Neumann PBPSO outperformed the other three pairs on problems with $m = 5$, but the difference with gbest SBPSO became smaller for larger values of $r$. For problems with $m = 10$, the Von Neumann PBPSO performed second best on 74 out of 81 problems. However, for problems with $m = 30$, the Von Neumann PBPSO ranked better than third only once out of 81 problems, and performed worse than both gbest SBPSO and the Von Neumann MBPSO. As mentioned in the discussion of the results in table 13, the parameters chosen for the Von Neumann PBPSO (which were the same as for lbest PBPSO) were probably better suited to problems with a lower number of constraints.

## 8 Conclusions and Future Work

This paper introduced set-based particle swarm optimization (SBPSO) as a generic particle swarm optimization (PSO) algorithm for use on discrete optimization problems that can be described as set-based problems. In addition to the attraction to personal best and neighborhood best positions, two mechanisms were included in SBPSO to ensure that the algorithm could explore the entire search space. These mechanisms were described in general terms, and a specific implementation of each was chosen for use in the experiments. The first mechanism stated that elements were removed randomly from the intersection of the current position, $X(t)$, the personal best position, $Y(t)$, and the neighborhood best position, $\widehat{Y}(t)$. The second mechanism stated that elements outside the union of $X(t)$, $Y(t)$, and $\widehat{Y}(t)$, were chosen to be added to the position via a *k*-tournament selection and using marginal objective function values.

The multidimensional knapsack problem (MKP) was chosen as the optimization problem to evaluate the performance of SBPSO, using a large number of benchmark problems from literature. SBPSO was compared to three existing discrete PSO algorithms, namely binary PSO (BPSO), modified binary PSO (MBPSO), and probability binary PSO (PBPSO). Each algorithm was evaluated using one of three swarm topologies, that is, the star topology, the ring topology with neighborhood size 4, and the Von Neumann topology. This resulted in 12 algorithm-topology pairs.

A Sobol pseudo-random number generator was used to generate low-discrepancy sequences in the parameter space to help tune each algorithm-topology pair separately, once on a set of small MKP, and once on a set of large MKP, for a total of 24 different tuning tasks. The same number of parameter value combinations were evaluated for each tuning task, and the best performing parameter value combination yielded the parameter values used in the testing phase.

A sensitivity analysis of SBPSO with respect to different values of its control parameters was done. This showed that the performance of SBPSO was most sensitive to $c_1$ (the attraction to personal best) and $c_3$ (the number of elements to add from outside the union of $X(t)$, $Y(t)$, and $\widehat{Y}(t)$).

For both the small MKPs as well as the large MKPs, and for each of the three swarm topologies used, the results showed that SBPSO outperformed the other three algorithms by a statistically significant margin. These results also held when the best performing topology was chosen for each PSO algorithm. The results also showed that the Von Neumann topology was the best topology to use for each of the algorithms on the small MKPs. For the large MKPs, the star topology was best for BPSO and SBPSO, while the Von Neumann topology was best for MBPSO and PBPSO.

The overall conclusion is that SBPSO performed better than the three other discrete PSO algorithms over a range of MKPs using different swarm topologies. These results were statistically significant at a significance level of $\alpha = 0.05$.

The goal of this paper was *not* to find the best algorithm for solving the MKP, but to propose a generic set-based PSO. State-of-the-art algorithms for solving the MKP perform better, for example the genetic algorithm described by Chu and Beasley (1998), which incorporates a domain specific repair operator. This algorithm perfectly solved all small MKP, while on the large MKP it recorded an average error of 0.54%. Of the algorithm-pairs tested in this paper, gbest SBPSO performed best on the large MKP with an average error of 1.72%. However, it is emphasized that SBPSO does not make use of any domain specific operators to improve performance.

The next steps will be to evaluate the SBPSO algorithm on different problems, including feature selection. Also, the contribution of the specific implementation chosen for the operators $\odot^-$ and $\odot^+$ to the performance of SBPSO will be investigated. The performance of SBPSO, should domain specific operators be included, will also be investigated.

## References

Abraham, A., Liu, H., Zhang, W., and Chang, T.-G. (2006). Scheduling jobs on computational grids using fuzzy particle swarm algorithm. In Gabrys, B., Howlett, R., and Jain, L., editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4252 of *Lecture Notes in Computer Science*, pages 500–507. Springer, Berlin/Heidelberg.

Benameur, L., Alami, J., and El Imrani, A. (2009). A new discrete particle swarm model for the frequency assignment problem. In *Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications*, pages 139–144, Piscataway, NJ. IEEE Press.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Confernece*, pages 11–18, San Francisco. Morgan Kaufmann.

Bock, J. and Hettenhausen, J. (2012). Discrete particle swarm optimisation for ontology alignment. *Information Sciences*, 192(0):152–173.

Chandrasekaran, S., Ponnambalam, S., Suresh, R., and Vijayakumar, N. (2006). A hybrid discrete particle swarm optimization algorithm to solve flow shop scheduling problems. In *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–6, Piscataway, NJ. IEEE Press.

Chen, W.-N., Zhang, J., Chung, H., Zhong, W.-L., Wu, W.-G., and Shi, Y. (2010). A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on Evolutionary Computation*, 14(2):278–300.

Chu, P. and Beasley, J. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86.

Clerc, M. (2004). Discrete particle swarm optimization illustrated by the traveling salesman problem. In Onwubolu, G. and Babu, B., editors, *New Optimization Techniques in Engineering*, pages 219–239. Springer, Berlin/Heidelberg.

Correa, E., Freitas, A., and Johnson, C. (2006). A new discrete particle swarm optimization algorithm applied to attribute selection in a bioinformatics data set. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 35–42, New York, NY. ACM Press.

Du, J.-X., Huang, D.-S., Zhang, J., and Wang, X.-F. (2005). Shape matching using fuzzy discrete particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 405–408, Piscataway, NJ. IEEE Press.

Eberhart, R. C., Kennedy, J., and Shi, Y. (2001). *Swarm intelligence*. Morgan Kaufmann series in evolutionary computation. Elsevier, Amsterdam.

Eberhart, R. C. and Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 81–86, Piscataway, NJ. IEEE Press.

Eberhart, R. C., Simpson, P. K., and Dobbins, R. W. (1996). *Computational Intelligence PC tools*. AP Professional, Boston, MA.

Franken, N. (2009). Visual exploration of algorithm parameter space. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 389–398, Piscataway, NJ. IEEE Press.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

Gao, F., Cui, G., Zhao, Q., and Liu, H. (2006). Application of improved discrete particle swarm algorithm in partner selection of virtual enterprise. *International Journal of Computer Science and Network Security*, 6(3A):208–212.

García, A., Pastor, R., and Corominas, A. (2006). Solving the response time variability problem by means of metaheuristics. *Frontiers in artificial intelligence and applications*, 146:187–196.

Gens, G. and Levner, E. (1980). Complexity of approximation algorithms for combinatorial problems: a survey. *Special Interest Group on Algorithms and Computation Theory News*, 12:52–65.

Hembecker, F., Lopes, H. S., and Godoy, J. W. (2007). Particle swarm optimization for the multidimensional knapsack problem. In *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms, Part I*, pages 358–365, Berlin/Heidelberg. Springer.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):pp. 65–70.

Iman, R. and Davenport, J. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics Part A - Theory and Methods*, 9(6):571–595.

Kennedy, J. and Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, volume 5, pages 4101–4109, Piscataway, NJ. IEEE Press.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimisation. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ. IEEE Press.

Kennedy, J. and Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1671–1676, Piscataway, NJ. IEEE Press.

Khan, S. and Engelbrecht, A. (2010). A fuzzy particle swarm optimization algorithm for computer communication network topology design. *Applied Intelligence*, 36:1–17.

Khanesar, M., Teshnehlab, M., and Shoorehdeli, M. (2007). A novel binary particle swarm optimization. In *Proceedings of the Mediterranean Conference on Control and Automation*, Piscataway, NJ. IEEE Press.

Khuri, S., Bäck, T., and Heitkötter, J. (1994). The zero/one multiple knapsack problem and genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing*, pages 188–193, New York, NY. ACM Press.

Kong, M. and Tian, P. (2006). Apply the particle swarm optimization to the multidimensional knapsack problem. In Rutkowski, L., Tadeusiewicz, R., Zadeh, L., and Zurada, J., editors, *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, volume 4029 of *Lecture Notes in Computer Science*, pages 1140–1149. Springer, Berlin/Heidelberg.

Kong, M., Tian, P., and Kao, Y. (2008). A new ant colony optimization algorithm for the multidimensional knapsack problem. *Computers & Operations Research*, 35(8):2672–2683.

Labed, S., Gherboudj, A., and Chikhi, S. (2011). A modified hybrid particle swarm optimization algorithm for multidimensional knapsack problem. *International Journal of*

*Computer Applications*, 34(2):11–16.

Langeveld, J. and Engelbrecht, A. P. (2011). A generic set-based particle swarm optimization algorithm. In *Proceedings of the International Conference on Swarm Intelligence*, Cergy, France. EISTI.

Li, B.-B., Wang, L., and Liu, B. (2008). An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 38(4):818–831.

Liu, B., Wang, L., and Jin, Y.-H. (2007). An effective PSO-based memetic algorithm for flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1):18–27.

Liu, H. and Abraham, A. (2007). An hybrid fuzzy variable neighborhood particle swarm optimization algorithm for solving quadratic assignment problems. *Journal of Universal Computer Science*, 13(9):1309–1331.

Liu, H., Abraham, A., and Hassanien, A. E. (2010). Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Generation Computer Systems*, 26(8):1336–1343.

Lorie, J. H. and Savage, L. J. (1955). Three problems in rationing capital. *The Journal of Business*, 28:229.

Ma, C.-X., Qian, L., Wang, L., Menhas, M. I., and Fei, M.-R. (2010). Determination of the PID controller parameters by modified binary particle swarm optimization algorithm. In *Proceedings of the Chinese Control and Decision Conference*, pages 2689–2694, Piscataway, NJ. IEEE Press.

Menhas, M. I., Wang, L., Fei, M.-R., and Ma, C.-X. (2011). Coordinated controller tuning of a boiler turbine unit with new binary particle swarm optimization algorithm. *International Journal of Automation and Computing*, 8:185–192.

Neethling, C. and Engelbrecht, A. (2006). Determining RNA secondary structure using set-based particle swarm optimization. In Yen, G., Lucas, S., Fogel, G., Kendall, G., Salomon, R., Zhang, B.-T., Coello, C., and Runarsson, T., editors, *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1670–1677, Piscataway. NJ. IEEE Press.

Nemenyi, P. (1963). *Distribution-free multiple comparisons*. PhD thesis, Princeton University, Princeton, NJ, USA.

Pampara, G., Franken, N., and Engelbrecht, A. (2005). Combining particle swarm optimisation with angle modulation to solve binary problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 89–96, Piscataway, NJ. IEEE Press.

Pang, W., Wang, K.-P., Zhou, C.-G., and Dong, L.-J. (2004a). Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In *Proceedings of the IEEE International Conference on Computer and Information Technology*, pages 796–800, Piscataway, NJ. IEEE Press.

Pang, W., Wang, K.-P., Zhou, C.-G., Dong, L.-J., Liu, M., Zhang, H.-Y., and Wang, J.-Y. (2004b). Modified particle swarm optimization based on space transformation for solving traveling salesman problem. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 4, pages 2342–2346, Piscataway, NJ. IEEE Press.

Puchinger, J., Raidl, G. R., and Pferschy, U. (2010). The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, 22:250–265.

Shen, B., Yao, M., and Yi, W. (2006). Heuristic information based improved fuzzy discrete PSO method for solving TSP. In *Proceedings of the Pacific Rim International Conference on Artificial intelligence*, pages 859–863, Berlin/Heidelberg. Springer.

Shen, Q., Jiang, J.-H., Jiao, C.-X., li Shen, G., and Yu, R.-Q. (2004). Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists. *European Journal of Pharmaceutical Sciences*, 22(2-3):145–152.

Shi, Y. and Eberhart, R. C. (1998). A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69–73, Piscataway, NJ. IEEE Press.

Shi, Y. and Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 101–106, Piscataway, NJ. IEEE Press.

Tasgetiren, M. F., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G. (2004). Particle swarm optimization algorithm for permutation flowshop sequencing problem. In Dorigo, M., Birattari, M., Blum, C., M.Gambardella, L., Mondada, F., and Stützle, T., editors, *Ant Colony, Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 366–385. Springer, Berlin/Heidelberg.

Tu, C.-J., Chuang, L.-Y., Chang, J.-Y., and Yang, C.-H. (2008). Feature selection using PSO-SVM. *IAENG International Journal of Computer Science*, 33:111–116.

Veenhuis, C. (2008). A set-based particle swarm optimization method. In Rudolph, G., Jansen, T., Lucas, S., Poloni, C., and Beume, N., editors, *Proceedings of the Parallel Problem Solving from Nature Conference*, volume 5199 of *Lecture Notes in Computer Science*, pages 971–980. Springer, berlin/Heidelberg.

Wang, K.-P., Huang, L., Zhou, C.-G., and Pang, W. (2003). Particle swarm optimization for traveling salesman problem. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 3, pages 1583–1585, Piscataway, NJ. IEEE Computer Society.

Wang, L., Wang, X., Fu, J., and Zhen, L. (2008). A novel probability binary particle swarm optimization algorithm and its application. *Journal of Software*, 3(9):28–35.

Wu, Z., Ni, Z., Gu, L., and Liu, X. (2010). A revised discrete particle swarm optimization for cloud workflow scheduling. In *Proceedings of the International Conference on Computational Intelligence and Security*, pages 184–188, Piscataway, NJ. IEEE Press.

Yang, S., Wang, M., and Jiao, L. (2004). A quantum particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 320–324, Piscataway, NJ. IEEE Press.

Zhang, C., Sun, J., Wang, Y., and Yang, Q. (2007). An improved discrete particle swarm optimization algorithm for TSP. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 35–38, Piscataway, NJ. IEEE Computer Society.

Zhen, L., Wang, L., Wang, X., and Huang, Z. (2008). A novel PSO-inspired probability-based binary optimization algorithm. In *Proceedings of the International Symposium on Information Science and Engineering*, volume 2, pages 248–251, Oulu. Academy Publisher.

Zhong, W.-L., Zhang, J., and Chen, W.-N. (2007). A novel discrete particle swarm optimization to solve traveling salesman problem. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3283–3287, Piscataway, NJ. IEEE Press.