## Particle Swarm Optimization (PSO) algorithm

Arguably since the invention of the electronic computer (possibly even earlier), scientists and philosophers alike pondered over the similarities between computer programs and minds. “Computers can process symbolic information, can derive conclusions from premises, can store information and recall it when it is appropriate, and so on—all things that minds do” - (J. F. Kennedy et al., 2001). They reasoned that the capability of minds to host intelligence gave direction to the possibilities for computers, hence, birthing the great quest for Artificial Intelligence (AI).

Progressing from the latter quarter of the nineties marked revolutionary findings in the development of AI technologies like the GA (Holland, 1992), evolutionary computation(Back et al., 1997), and the Artificial Neural Network (ANN) (Jain et al., 1996). However, the social psychologist James Kennedy and his associates (2001) observed a stereotype creeping into the general understanding of AI at the time that was limiting the understanding of AI at the time. They noted that early AI researchers understood the measure of intelligence as the ability to solve large, complex, and sometimes multipart, problems quickly. Due to the variety in methods to approach many problems, building an intelligent computer program that finds the best choice required and motivated them to think of a number of clever methods. They developed ‘logical shortcuts’, called *heuristics*, that speed up the process in a manner that was applicably reusable. The programs developed by the researchers were simply outstanding at problem-solving, calculation and memory retention but were found to fail at simpler things like conversation and face recognition. This was due to the continuously growing number of variables still needing to be addressed in the problem domains it was being applied to. There was always something else that could go wrong.

“The early AI researchers had made an important assumption, so fundamental that it was never stated explicitly nor consciously acknowledged” (J. F. Kennedy et al., 2001). AI at the time was modelled on the vision of a single disconnected person capable of coolheadedly handling the situations posed to them using the information and logical reasoning stored in their brain. However, they argued that if human intelligence was the intended model, then this model of understanding was devoid of an important comportment/behaviour involved in human reasoning and development: *Socialization*.

In real social interaction, not only information but also rules, tips, and also methods of processing information are exchanged. They observed further social behaviours which were “the norm throughout the animal kingdom” in other organisms like Fish schooling, birds flocking and bugs swarming. These behaviours occurred not only for copulation purposes but included important necessities for the population like “having a thousand eyes” to keep watch for predators and searching for food, among other advantages.

Their book “Swarm Intelligence” cited earlier, introduced the concept of exploiting social behaviours by splitting the computational requirements of a system across a group or ‘swarm’ of intercommunicating individuals. Gaining inspiration from the natural examples mentioned earlier, they proposed a model called *Particle Swarm Optimization* which differed from the popular evolutionary computation methods at the time because its population members, named *particles*, were first initialized with stochastically assigned positions and velocities, and then flown through the problem space in search of a solution. The algorithm uses two important variables *pbest* and *gbest*, both respectively meaning the particle’s personal best position found on the search space and the global best position found across all particles. Sometimes the *gbest* may even be calculated using a heuristic like K nearest neighbours or clustering techniques(Ahmadyfard & Modares, 2008). With each iteration, the velocities associated with each particle were stochastically accelerated towards its previous best-found location and the neighbour-hood best location in the search space. These stochastic mechanisms, changing with each iteration and being applied to the vectors, gave the algorithm a “lifelike” appearance as the particles buzzed around the search space (J. Kennedy & Eberhart, 1995). Their initial programs aimed to model the coordinated behaviour of bird flocks and schools of fish, but as they ran the algorithm on a two-dimensional plane, the particles’ movements began to resemble swarms of mosquitoes. This fact, along with the point that each particle was, in essence, a mass-less and volume-less mathematical abstraction that can be called a point when stationary, they deemed the terms *particle* and *Particle Swarm Optimization* as fitting descriptors.

Due to its simplistic and effective design, PSO has gained a lot of popularity over the years (Blum & Li, 2008; Eberhart & Shi, 2001; J. Kennedy & Eberhart, 1995) and has found applications in many domains such as scalable optimization for social learning (Cheng & Jin, 2015), clustering for high dimensional data sets (Esmin et al., 2015), and multi-objective optimization (Delgarm et al., 2016). With regards to the TSP, studies were undertaken to adapt the algorithm to these discrete domains (Zhong et al., 2007). One interesting algorithm adaptation was the one proposed by Wang et al. (2003) and further improved by Yousefikhoshbakht (2021), where a series of swap sequences were used to represent vectors. This adaptation was chosen as the PSO focus for this study.

### Defining Problem domain

In the PSO, each member of the swarm is composed of 3 D-dimensional vectors (Poli et al., 2007). These vectors store the current position , the previous best position and the velocity . D stands for the number of dimensions within the given search space. At the start of the algorithm, the particles are initiated at random locations within the search space and, using these 3 variables along with the global best position , the particle navigates its search space. Of course, with regards to the TSP, there is no continuous search space for the particles to use. There are a discrete number of solutions that can be given to any given map (based on its size) and further validation that filters acceptable solutions (based on the TSP conditions 1 and 2). For this reason, the approach taken towards the understanding of ‘vectors’ needs to be adapted.

In keeping consistent and to ease the future hybridization steps, it was decided to have the PSO operating on the same search space as the ACO and GA. This meant that similar to the GA, the location of a particle at any given time was a complete and valid proposed solution to the TSP represented as a sequence of cities to visit , where is the number of cities on the map. The movement vector would then need to be an operation capable of altering without breaking is validity. Wang et al. (2003) proposed the representation of these movement vectors as swap operators, defined as , that when applied to swap the location of the cities and occuring within the sequence. This creates a completely new sequence which can be treated as the new location vector for the particle after the movement vector was applied to it . The velocity vector can contain any number of swap operators compiled together as *Swap Sequence* , which can then be applied to any location vector to bring it to another position within the search space. With this understanding in mind, the velocity needed to bring an example particle from its current location to its personal best-found location since the algorithm began , can be understood as the question: What swaps to my current sequence of cities is needed until it becomes the *pbest* sequence?

Diagram

Description automatically generated with medium confidence

Figure 1 - Applying Velocity (SS) vector to Position vector

Another big consideration in this application of the algorithm is how to the weights are represented. In normal velocity vectors, simple vector scaling is done by multiplying it by the weights assigned. However, with our new representation of the velocity vector, the application of weights needs to be rethought. Continuing with their proposed model, Wang et al. (2003) repurposed the weights used in the algorithm to stand for probabilities for each of the Swap Operators being maintained in the Sequence after the weight is applied.

Where is an operator that returns with a probability of , otherwise, it returns nothing.

### Initial Population

At the start of the algorithm, each particle is initialized with a random solution/ position on the search space. Then for each iteration of the algorithm, the particles undergo a 2-step process. First, the particles are evaluated using the standard route evaluation method detailed in equation x to update each particle’s *pbest* and the algorithm’s *gbest* values. After this, the velocities for each particle are calculated and applied to its position to traverse the search space.

#### Inertia weight

Because velocities were used in the original algorithm, the concept of momentum comes into play when designing an algorithm model. The momentum of the particle is dictated by the inertia weight and, because of the adapted nature of weights in this algorithm, it ranges between 0-1. This inertia weight is used as the probability of swap operators persisting in the velocity after it is applied to the position for particle movement. An inertia weight of 0 completely refreshes the velocity vector after each movement and renders the concept of velocity obsolete, while a value of 1 places the particle in a frictionless environment where its velocity is ever-increasing with each calculation. Balancing the right levels for the inertia weight is important because it dictates the capabilities of the algorithm when it comes to exploration (higher ) vs exploitation (lower ). Some authors have found the technique of linearly decreasing from to over the whole run; prioritising exploration nearer the start of the algorithm and slowly focusing more on exploitation as time goes by, was an effective way to handle that weight. The experiments done showed that (retention probability of 50%) was the best weight score and it outperformed the linear scaling method when used in this study.

#### Stochasticity

J. Kennedy & Eberhart, (1995) noted, unfortunately, that using the simple rules where the particle simply followed his best direction with an aspect of inertia and skewed with ‘advice’ of the global best direction found, the flock quickly settled on a unanimous, unchanging direction. To solve this, they included a variable they named *craziness* to randomly change the velocities, introducing variation into the system, and giving it a “lifelike” appearance. This then became the common PSO algorithm where, for each iteration, the velocity applied to move the ant an accumulation of:

1. The current velocity with an inertia weight
2. The velocity towards our personal best position -> ; where is the importance weight of this velocity and is the *craziness* function returning a random number for this iteration.
3. The velocity towards the global best position found -> ; where is the importance weight of this velocity and is the *craziness* function returning a random number for this iteration.

For the adaptation used in this study, because the weights attached to a velocity only represent the retention of operators within that velocity, both the and can be merged with their counterparts (Wang et al., 2003). This means the final decision-making algorithm to calculate the velocity of a given particle for each iteration is:

Where and are random weights between 0-1 generated each iteration, velocity towards the personal best position, and is the velocity towards the global best position.

#### MMPSO

Yousefikhoshbakht, (2021) found optimization problems with the application of this algorithm to industry services, one of the weaknesses highlighted being premature convergence on local optimums. Some of the application challenges highlighted in that domain were: the large size of problems that managers face daily, the importance rankings of the different problems based on user/customer attention, and the consistency in answers returned from the various manager and customer problems. A balance needed to be found between local searches for susceptible areas and global best searches, which they tackled through their proposed PSO variant named the MPSO. There introduced another important variable called *gcbest* tracking the global best location found *for that iteration only*. To track the use of *gcbest,* variable bound between a min () and max () was used along with an accompanying inverse . is the probability that *gbest* will be used for this iteration’s vector calculation step, while its inverse is the probability of using *gcbest* instead.

At the start of the algorithm, , representing a probability of for using gbest this iteration, and as the iterations increase, linearly progresses towards . The method is quite simple, and the experiments detailed in chapter 4 reveal success with using this methodology when compared to the classic PSO. No adaptation needed to be done to this model when implementing it in the study.

## Bibliography

Ahmadyfard, A., & Modares, H. (2008). Combining PSO and k-means to enhance data clustering. *2008 International Symposium on Telecommunications*, 688–691. https://doi.org/10.1109/IS℡.2008.4651388

Back, T., Hammel, U., & Schwefel, H.-P. (1997). Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, *1*(1), 3–17. https://doi.org/10.1109/4235.585888

Blum, C., & Li, X. (2008). Swarm Intelligence in Optimization. In C. Blum & D. Merkle (Eds.), *Swarm Intelligence: Introduction and Applications* (pp. 43–85). Springer. https://doi.org/10.1007/978-3-540-74089-6\_2

Cheng, R., & Jin, Y. (2015). A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, *291*, 43–60. https://doi.org/10.1016/j.ins.2014.08.039

Delgarm, N., Sajadi, B., Kowsary, F., & Delgarm, S. (2016). Multi-objective optimization of the building energy performance: A simulation-based approach by means of particle swarm optimization (PSO). *Applied Energy*, *170*, 293–303. https://doi.org/10.1016/j.apenergy.2016.02.141

Eberhart, & Shi, Y. (2001). Particle swarm optimization: Developments, applications and resources. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, *1*, 81–86 vol. 1. https://doi.org/10.1109/CEC.2001.934374

Esmin, A. A. A., Coelho, R. A., & Matwin, S. (2015). A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artificial Intelligence Review*, *44*(1), 23–45. https://doi.org/10.1007/s10462-013-9400-4

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, *29*(3), 31–44. https://doi.org/10.1109/2.485891

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN’95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Kennedy, J. F., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence*. Morgan Kaufmann Publishers.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence*, *1*(1), 33–57. https://doi.org/10.1007/s11721-007-0002-0

Wang, K.-P., Huang, L., Zhou, C.-G., & Pang, W. (2003). Particle swarm optimization for traveling salesman problem. *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, *3*, 1583-1585 Vol.3. https://doi.org/10.1109/ICMLC.2003.1259748

Yousefikhoshbakht, M. (2021). Solving the Traveling Salesman Problem: A Modified Metaheuristic Algorithm. *Complexity*, *2021*, e6668345. https://doi.org/10.1155/2021/6668345

Zhong, W., Zhang, J., & Chen, W. (2007). A novel discrete particle swarm optimization to solve traveling salesman problem. *2007 IEEE Congress on Evolutionary Computation*, 3283–3287. https://doi.org/10.1109/CEC.2007.4424894