Particle Swarm Optimization (PSO) Performance in Solving the Train Location Problem at Transshipment Yard

Alhossein Mohamed, Qiyuan Peng

School of Transportation and Logistics, Southwest Jiaotong University, Chengdu, China
Email: mhmdhosen@gmail.com

Received 20 July 2014; revised 2 September 2014; accepted 6 October 2014

Abstract

Particle swarm optimization (PSO) is an evolutionary computation technique; it has shown its effectiveness as an efficient, fast and simple method of optimization. In this paper, the mathematical model represents NP-hard in the strong sense; since any instance of the quadratic assignment problem (QAP), I will implement the particle swarm optimization (PSO) for the quadratic assignment problem (QAP). The results show that the PSO is an appropriate optimization tool for use in determining the train location in the transshipment yard by comparing it with previous studies to know the PSO's performance.

Keywords

Particle Swarm Optimization, Transshipment Yards, Train Location

Subject Areas: **High Performance Computing, Information and Communication Theory and Algorithms**

1. Introduction

With increasing volume of rail cargo an efficient operation of railway systems obviously becomes more and more important. In this context, modern rail-rail transshipment yards are an efficient alternative to traditional classification (or marshaling) yards, which both allow for a consolidation of different freight trains with equal destination. In this way, hub-and-spoke systems, like they are traditionally applied for road transport, become available for railway systems as well, and replace cost-intensive point-to-point transport of multiple small freight trains. Whereas traditional classification yards require a time-consuming reshuffling of railway cars via a system of track switches, in a modern transshipment yard container handling is conducted by huge gantry cranes spanning the railway tracks (**Figure 1**). The PSO algorithm is applied in determining the freight train location in transshipment and detailed results are shown and compared with other evolutionary algorithms results.

2. Problem Description

The train location problem (TLP) has to be solved perpetually for every pulse of trains arriving at a transshipment yard. The TLP assigns each train of the given pulse a track (vertical parking position) and additionally decides on each train's horizontal parking position along the yard. It is the aim of the TLP to evenly spread the overall workload among the given gantry cranes of the yard, so that by minimizing the maximum workload of cranes train processing of a given pulse is accelerated. This basic decision problem relies on some premises [1].

With suited parameters d_{ipjqc} on hand and the additional notation summarized in Table 1 the train location problem (TLP) can be formalized by a quadratic program consisting of objective function (1) subject to constraints (2) to (4):

$$(\text{TLP})\,\text{Minimize } F(X) = \max_{c\in C}\left\{\sum_{i\in I}\sum_{j\in I}\sum_{p\in P_i}\sum_{q\in P_j} d_{ipjqc}\cdot x_{ip}\cdot x_{jq}\right\} \qquad (1)$$

Subject to:

$$\sum_{p\in P_i} x_{ip} = 1 \quad \forall i \in I \qquad (2)$$

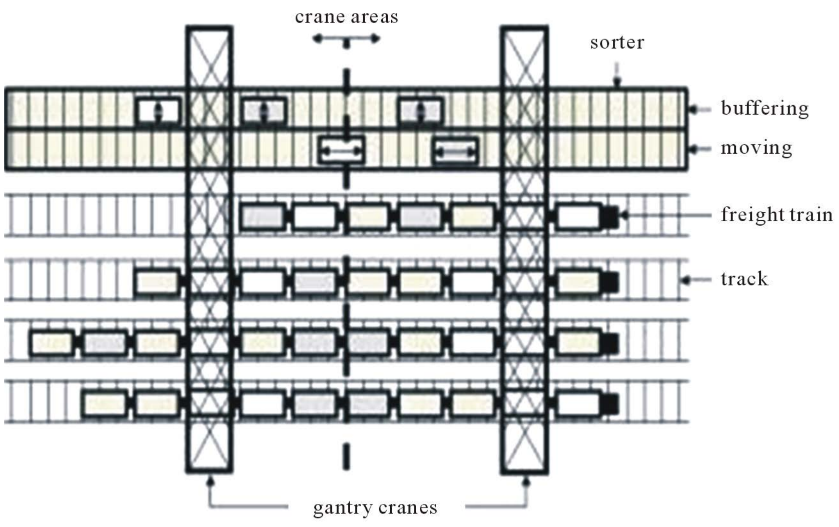

Figure 1. Rail-rail transshipment yard.

Table 1. Notation.

I	set of trains (indices i and j)
C	set of cranes (index c)
G	set of parallel tracks within transshipment yard (index g)
P	set of all parking positions (indices p and q)
P_g	set of parking positions on track g
P_i	set of park positions available for train i
d_{ipjqc}	total distance (or time span) to be processed by crane c between train i at position p and train j at position q
x_{ip}	binary variable: 1, if train i is assigned at position p; 0, otherwise

$$\sum_{i \in I} \sum_{p \in P_q} x_{ip} = 1 \quad \forall g \in G \tag{3}$$

$$x_{ip} \in \{0,1\} \quad \forall i \in I,\ p \in P_i \tag{4}$$

In objective function (1) the maximum workload of all cranes $c \in C$ is to be minimized, where each crane's workload is determined by summing up the respective workload parameters d_{ipjqc} whenever train i is parked on position p (with $x_{ip} = 1$) and another train j is parked on position q (with $x_{jq} = 1$). Equalities (2) ensure that each train i of the given pulse I is assigned to exactly one parking position out of its possible positions P_i. On the other hand, it is to be ensured (constraints (3)) that each track g receives exactly one train, where $P_g \subset P$ comprises all parking positions of track g. Finally, binary variables x_{ip} are defined by (4).

Obviously, TLP with facultative weights d_{ipjqc} is NP-hard in the strong sense, since any instance of the quadratic assignment problem (QAP) can be easily reduced to an instance of TLP with a single crane $C = \{1\}$ where the length of all trains is equal to the length of the yard, so that only one parking position per track is feasible. QAP was proven to be NP-hard in the strong sense by Sahni and Gonzalez (1976).

This problem had been studies by Michael Kellner, Nils Boysen, Malte Fliedner [1] using efficient heuristic solution procedures to solve it, which they used a myopic heuristic procedure and two meta-heuristics, a simulated annealing approach and a genetic algorithm. And today I am going to solve it by particle swarm optimization (PSO). Appling the particle swarm optimization tools in mat lab programming software.

3. Particle Swarm Optimization Implementation

3.1. Particle Swarm Model

The classical PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search the new solutions, where the fitness, f, can be calculated as the certain qualities measure [2]. Each particle has a position represented by a position-vector x_i (i is the index of the particle), and a velocity represented by a velocity-vector v_i. Each particle remembers its own best position so far in a vector $x_i^{\#}$, and its j-th dimensional value is $x_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector x^*, and its j-th dimensional value is x_j^*. During the iteration time t, the update of the velocity from the previous velocity to the new velocity is determined by Equation (5). The new position is then determined by the sum of the previous position and the new velocity by Equation (6).

$$v_{ij}(t) = w v_{ij}(t-1) + c_1 r_1 \left(x_{ij}^{\#}(t-1) - x_{ij}(t-1) \right) + c_2 r_2 \left(x_j^*(t-1) - x_{ij}(t-1) \right) \tag{5}$$

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \tag{6}$$

where r_1 and r_2 are the random numbers in the interval [0, 1]. c_1 is a positive constant, called as coefficient of the self-recognition component, c_2 is a positive constant, called as coefficient of the social component. The variable w is called as the inertia factor, which value is typically setup to vary linearly from 1 to near 0 during the iterated processing. From Equation (5), a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm.

In the PSO model, the particle searches the solutions in the problem space within a range $[-s; s]$ (if the range is not symmetrical, it can be translated to the corresponding symmetrical range). In order to guide the particles effectively in the search space, the maximum moving distance during one iteration is clamped in between the maximum velocity $[-v_{\max}; v_{\max}]$ given in Equation (7), and similarly for its moving range given in Equation (8):

$$v_{i,j} = \operatorname{sign}(v_{i,j}) \min\left(|v_{i,j}|, v_{\max} \right) \tag{7}$$

$$x_{i,j} = \operatorname{sign}(x_{i,j}) \min\left(|x_{i,j}|, x_{\max} \right) \tag{8}$$

The value of $v_{\max}$ is $\rho \times s$, with $0.1 \le \rho \le 1.0$ and is usually chosen to be s, *i.e.* $\rho = 1$. The pseudo-code for particle swarm optimization algorithm is illustrated in algorithm.

Algorithm: Particle Swarm Optimization Algorithm

```
01. Initialize the size of the particle swarm $n$, and other parameters.
02. Initialize the positions and the velocities for all the particles randomly.
03. While (the end criterion is not met) do
04.     $t = t + 1$
05.     Calculate the fitness value of each particle;
06.     $x^* = \arg\min_{i=1}^{n}\left(f\left(x^*(t-1)\right), f\left(x_1(t)\right), f\left(x_2(t)\right), \cdots, f\left(x_i(t)\right), \cdots, f\left(x_n(t)\right)\right)$
07.     For $i$ = 1 to $n$
08.         $x_i^\#(t) = \arg\min_{i=1}^{n}\left(f\left(x_i^\#(t-1)\right), f\left(x_i(t)\right)\right)$;
09.         For $j$ = 1 to $d$
10.             Update the $j$-th dimension value of $x_i$ and $v_i$
10.                 according to Eqs.
12.         Next $j$
13.     Next $i$
14. End While.
```

3.2. Experiment Settings

3.2.1. Yard Layout

With regard to the yard layout, we base the study on the typical setting of German transshipment yards. A typical yard length is 700 meters and slots are adjusted to accommodate standard railcars with a total length of 14 meters. Thus, we assume a yard length of T = 50 slots, with a horizontal distance of d^h = 14 meters between any two adjacent slots. Furthermore, we assume a vertical distance of d^v = 7 meters between neighboring tracks and the sorter, which is located below the final track. The number $|G|$ of parallel tracks and the number $|C|$ of gantry cranes are varied as follows: $|G| \in \{4,6,8\}$ and $|C| \in \{2,4,6\}$, so that differently sized transshipment yards are investigated.

3.2.2. Container Moves

The train length (in slots) is randomly determined, where different scenarios (short trains only, long trains only and mixed train lengths) are generated by drawing l_i out of intervals [15 - 25], [35 - 45] and [15 - 45], respectively. Furthermore, we aim to investigate different workload settings, namely low, medium and high workload.

Therefore, $\gamma = r \cdot M^{\max}$ container moves are determined by randomly drawing trains i and j and train positions k and l (counted from the engine) for each move, where $i \neq j$ must hold. $M^{\max}$ and r denote the maximum number of container moves (for given train lengths) and a fraction value randomly drawn from intervals [0.1, 0.2] (low), [0.4, 0.6] (medium) and [0.8, 0.9] (high workload), respectively.

3.2.3. Technical Crane Parameters

The gantry cranes move in horizontal and vertical direction simultaneously driven by independent engines. In horizontal direction the whole crane moves on special rail tracks, whereas vertically merely the steeple cab carrying the spreader is moved. Thus, the maximum time span for executing the vertical and horizontal movement determines the processing time of a container move. We assume a velocity of crane and steeple cab of v^e = 3 meters per second, if the crane moves empty, whereas the velocity reduces to v^l = 2 meters per second, if a container is carried. Once positioned, picking and dropping of containers requires additional processing time. Especially, locating the spreader is precision work, so that we assume a typical time span of t^d = 45 seconds for picking or dropping a container. See Alicke (2002) and Martinez *et al.* (2004) for comparable parameters.

3.2.4. Crane Movement

A typical real-world policy to determine the division of labor among cranes is to partition the yard into equally sized areas (Boysen and Fliedner, 2009). Thus, we follow this widespread approach and assign each given crane the same number of slots (accounting for rounding differences). If a bundle of trains is parked and, thus, all container moves are fixed and assigned to gantry cranes, sequencing moves per crane remains a complex optimization problem.

In our case, for each crane an asymmetric traveling salesman problem would need to be solved while considering the interdependencies among cranes resulting from split moves.

However, the sequence of container moves is typically not optimized by a scheduling procedure but locally determined by the respective crane operator. Thus, to simulate a human decision rule we apply a simple nearest neighbor heuristic. Each crane's starting position is the left hand border of its area, while the steeple cab is positioned over the sorter. From there it consecutively executes the container move closest to its current position. Split moves, e.g., from yard area A to B, are considered by updating crane B's list of unprocessed container moves, not before the respective container arrived in the sorter-segment of crane B. Thus, the list is updated just after crane A processed the first part of the split move (from train into the sorter) and a vehicle (with sorter velocity $v^s = 3$) moved the container into yard area B. With regard to the sorting system it is assumed that vehicles are not a bottleneck and congestions do not occur.

The aforementioned parameters of instance generation are summarized in **Table 2**. All parameters are combined in a full-factorial design and for each parameter constellation 10 replications are generated, so that 3 × 3 × 3 × 3 × 10 = 810 different instances were obtained.

For each of these instances a preprocessing has to be executed, which determines weights d_{ijklc} of total workload resulting from loaded moves for each train pair i and j and feasible parking positions of trains. If feasible positions of a train pair are presumed, the workload w^D of a direct container move from track t and slot s to track t' and slot s' can be determined as follows:

$$w^D\left((t,s),(t',s')\right) = 2\cdot t^p + \max\left\{\frac{|s-s'|\cdot d^h}{v^l};\frac{|t-t'|\cdot d^v}{v^l}\right\}$$

If processed as a direct move, a container move $\left((t,s),(t',s')\right)$ requires a pick and a drop operation $\left(2\cdot t^p\right)$. Additionally, the time for the actual move is to be added, which amounts to the maximum of the crane's horizontal and vertical distance each weighted with velocity v^l (for a loaded move). Split moves into (with weight w^{IN}) and out of (with weight w^{OUT}) the sorter also require a pick and drop operation and the actual movement time, which is required for the vertical movement between rail track and sorter track t_s:

$$w^{\text{IN}}\left((t,s),(t_s,s)\right) = 2\cdot t^p + \frac{(t_s - t)\cdot d^v}{v^l}$$

$$w^{\text{OUT}}\left((t_s,s'),(t',s')\right) = 2\cdot t^p + \frac{(t_s - t')\cdot d^v}{v^l}$$

The sum of resulting workload weights w for the respective container moves between two trains amount to weights d_{ijklc}. Note that the workload of empty moves required for the yard simulation can be calculated in the same fashion. Then, the four heuristic solution procedures are applied to determine parking positions of trains. First, with regard to the solution performance these results are compared to optimal TLP objective values, so that the gap of our heuristic procedures can be determined. Then, resulting parking positions of trains are passed over to our yard simulation, where the schedules of cranes and sorter are determined. This way, the workload of loaded moves (as considered in model TLP) can be compared to the real-world workload (approximated by the simulation), so that the suitability of our surrogate objective and the acceleration of container processing obtained by optimized parking positions can be determined.

3.3. Yard Simulation

The results of our yard simulation, Here, parking positions of trains obtained by TLP are passed over and real-world crane and sorter operations are simulated. This way, the results of TLP can be compared with the resulting

Table 2. Parameters for instance generation.

Symbol	Description	Values		
$\lvert G\rvert$	Number of tracks	4	6	8
$\lvert C\rvert$	Number of cranes	2	4	6
l_i	Length of trains	Short [15 - 25]	Long [35 - 45]	Mixed [15 - 45]
r	Fraction of container moves	Small workload [0.1, 0.2]	Medium workload [0.4, 0.6]	Small workload [0.8, 0.9]

(approximate) real-world workload of cranes. While TLP minimizes the cranes' workload merely on the basis of loaded moves (surrogate objective, denoted as SURR), the yard simulation provides the actual workload consisting of loaded and empty crane moves (actual objective, denoted as ACT). On average over all 810 instances, the workload of SURR determined by our best-performing PSO procedure already makes up 89.01% of the value of ACT, which results from the over proportional influence of time consuming pick and drop operations. Moreover, the coefficient of correlation (Pearson's product-moment correlation) between both approaches amounts to a remarkable 0.9992. Thus, the conclusion can be drawn, that our surrogate objective of merely considering loaded moves is a suited simplification. On the one hand, the actual objective of reducing the overall workload is strongly supported and, on the other hand, the solution process is considerably alleviated by excluding a detailed crane scheduling.

Furthermore, we aim at investigating the question whether optimized parking positions enable a considerable reduction of train processing time compared to real-world policy RWP. Recall that RWP assigns tracks according to a first-come-first-serve policy and parks all trains in slot 1. As performance measures, we report the average absolute deviation (labeled *avg abs*) between both policies with regard to the makespan of processing the current bundle of trains as approximated by our simulation study. *Avg abs* denominates the acceleration of train processing if improved parking positions are applied instead of RWP in minutes averaged over all instances of the respective parameter constellation. Furthermore, the average relative deviation (labeled *avg rel*) of both policies in percent is reported, where the deviation is measured by $\frac{F\left(\text{RWP}\right)-F\left(\delta\right)}{F\left(\delta\right)}*100$ with $F\left(\text{RWP}\right)$ and $F\left(\delta\right)$ being the make span of our yard simulation when parking positions are determined by a real-world rule of thumb or improved with one of our heuristic procedures $\delta\in\left\{\text{MSP},\text{SA},\text{GA},\text{PSO}\right\}$ respectively. **Table 3**

Table 3. Absolute and relative speed-up of container processing time depending on yard size.

$\lvert G\rvert$	$[C]$			Average
	2	4	6	
4				
MPS	15.81/32.11	8.78/35.99	6.73/33.23	10.44/33.78
SA	20.78/51.38	10.33/44.05	8.06/42.30	13.06/45.91
GA	20.74/51.14	10.23/43.63	8.30/45.22	13.09/46.67
PSO	20.54/58.06	11.19/47.23	7.95/46.24	13.23/50.51
6				
MPS	24.31/36.31	15.25/43.59	11.54/45.21	17.03/41.70
SA	31.84/56.15	17.03/53.19	13.27/54.58	20.71/54.64
GA	31.87/56.91	17.48/55.33	13.52/57.70	20.96/56.65
PSO	33.78/55.08	23.48/50.96	17.20/59.77	24.82/55.27
8				
MPS	33.09/39.28	21.09/47.75	15.73/47.13	23.30/44.72
SA	42.39/59.94	24.26/59.02	17.85/56.61	28.17/58.52
GA	43.22/62.49	25.08/64.63	18.84/64.07	29.05/63.73
PSO	47.38/69.94	29.19/68.77	26.35/70.03	34.31/69.58
Average				
MPS	24.40/35.90	15.04/42.54	11.33/41.86	16.93/40.07
SA	31.67/55.82	17.20/52.09	13.06/51.16	20.64/53.02
GA	31.94/56.85	17.60/54.53	13.55/55.66	21.30/55.68
PSO	33.90/61.03	21.29/55.65	17.17/58.68	24.12/58.45

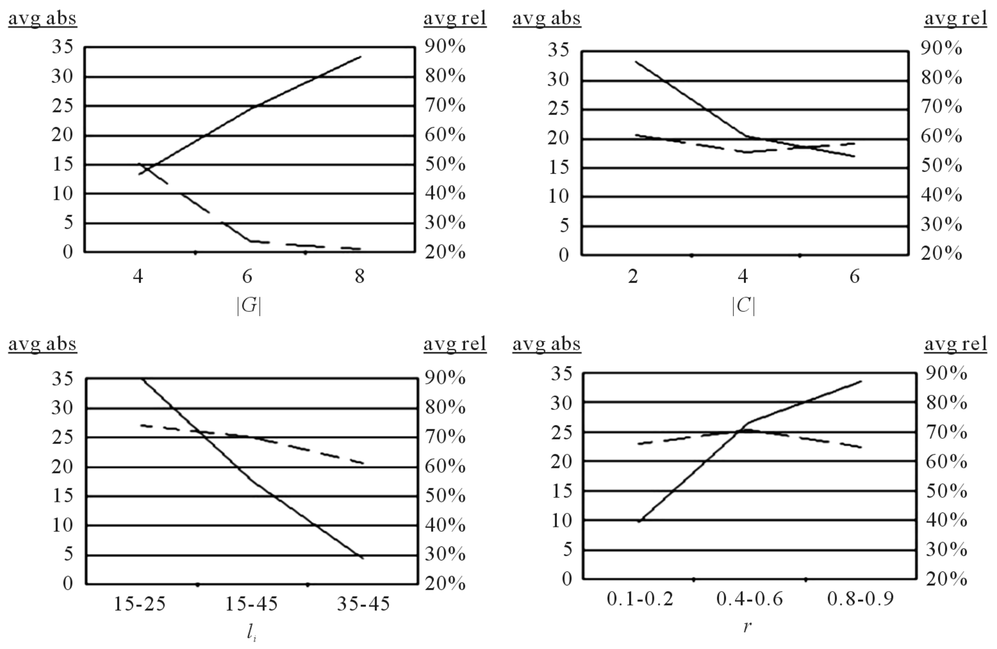

Figure 2. Absolute and relative speed-up in dependency of parameters of instance generation.

lists both performance measures in dependency of the parameters: number $|G|$ of tracks and number $|C|$ of cranes, which together reflect the size of a transshipment yard. The results reveal a remarkable potential for accelerating train processing. Depending on the size of the yard, possible absolute accelerations (*avg abs*) of our best-performing PSO procedure deviate between 47.38 minutes with eight tracks (high overall workload) and two cranes (low division of labor) and 7.95 minutes with four tracks (low overall workload) and six cranes (high division of labor). Interestingly, the relative acceleration (*avg rel*) of train processing performs somewhat contrarily. This is explained by the fact, that with a high division of labor the average make span tends to be lower in value, because a pulse of trains is processed much faster.

As a consequence a comparable absolute reduction in makespan leads to a higher relative reduction. In relative terms, train processing is accelerated by between 47.23% and 70.03% whenever parking positions are determined by PSO.

Further conclusions (in terms of a sensitivity analysis) can be drawn if the speed-up of improved parking positions (determined with PSO) is related to the parameters of instance generation. Therefore, **Figure 2** displays the average relative deviation (*avg rel in* %) and the average absolute deviation (*avg abs in minutes*) in dependency of the parameters: number $|G|$ of tracks, number $|C|$ of cranes, interval of train lengths l_i and interval of fraction value r, which determines the number of containers (workload) to be processed, respectively.

4. Conclusions

- With an increasing number $|G|$ of tracks the overall workload is increased.
- The higher the division of labor (more cranes $|C|$), the lower the absolute (*avg abs*) speed-ups of optimal parking positions, whereas relative speed-up remains almost unaffected by additional cranes.
- With shorter trains more degrees of freedom exist with regard to finding feasible parking positions within a yard.
- With an increasing fraction of value r, the number of containers transshipped rises.

The performance of particle swarm optimization (PSO) method was studied in this paper compared with traditional and other evolutionary optimization methods and gave the best results in solving the problem.

References

[1] Kellner, M., Boysen, N. and Fliedner, M. (2009) How to Park Freight Trains on Rail-Rail Transshipment Yards. Frie-

drich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Germany.

[2] Liu, H. and Abraham, A. (2007) A Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems. *Journal of Universal Computer Science*, **13**, 1032-1054.