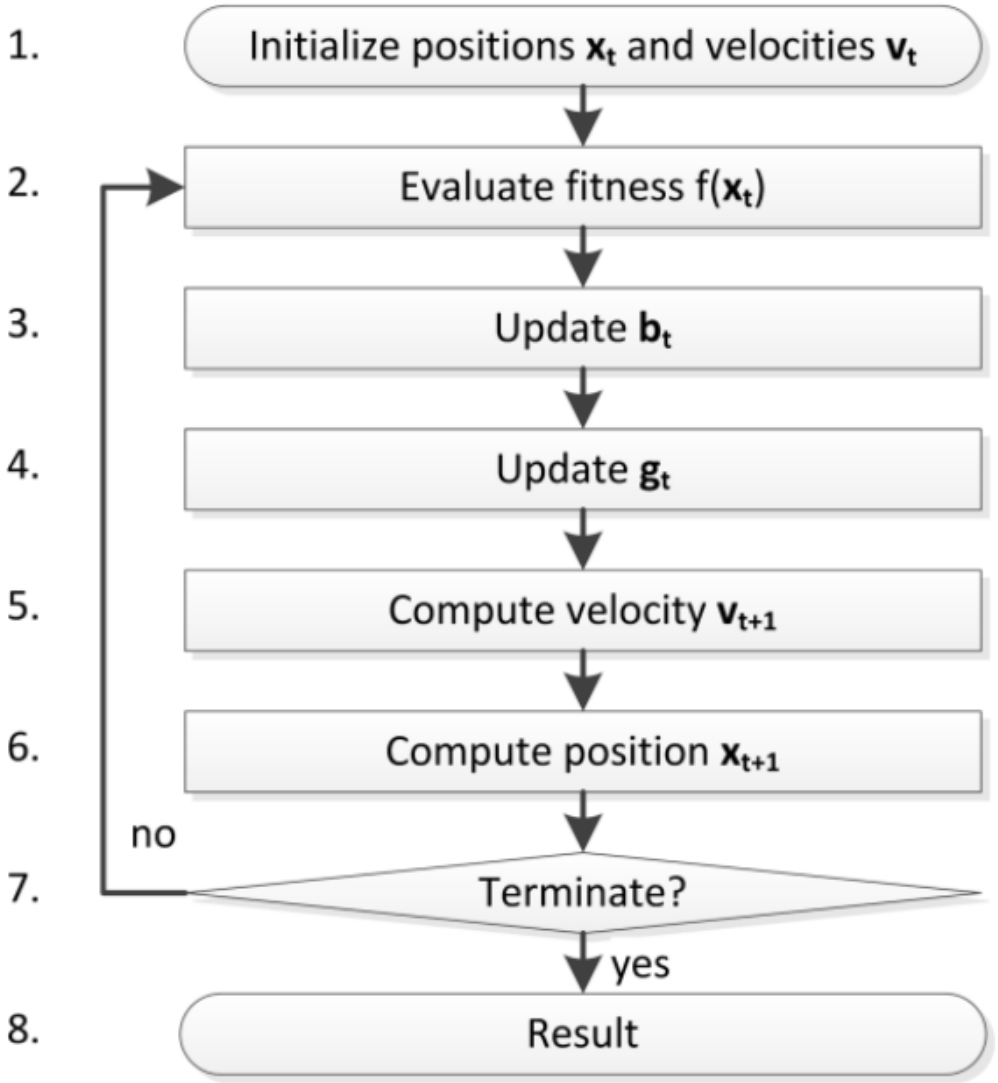
Методы распараллеливания популяционных алгоритмов оптимизации

Уровни распараллеливания

- Уровень задач
- **Уровень данных**
- **Уровень алгоритмов**
- Уровень инструкций

PSO

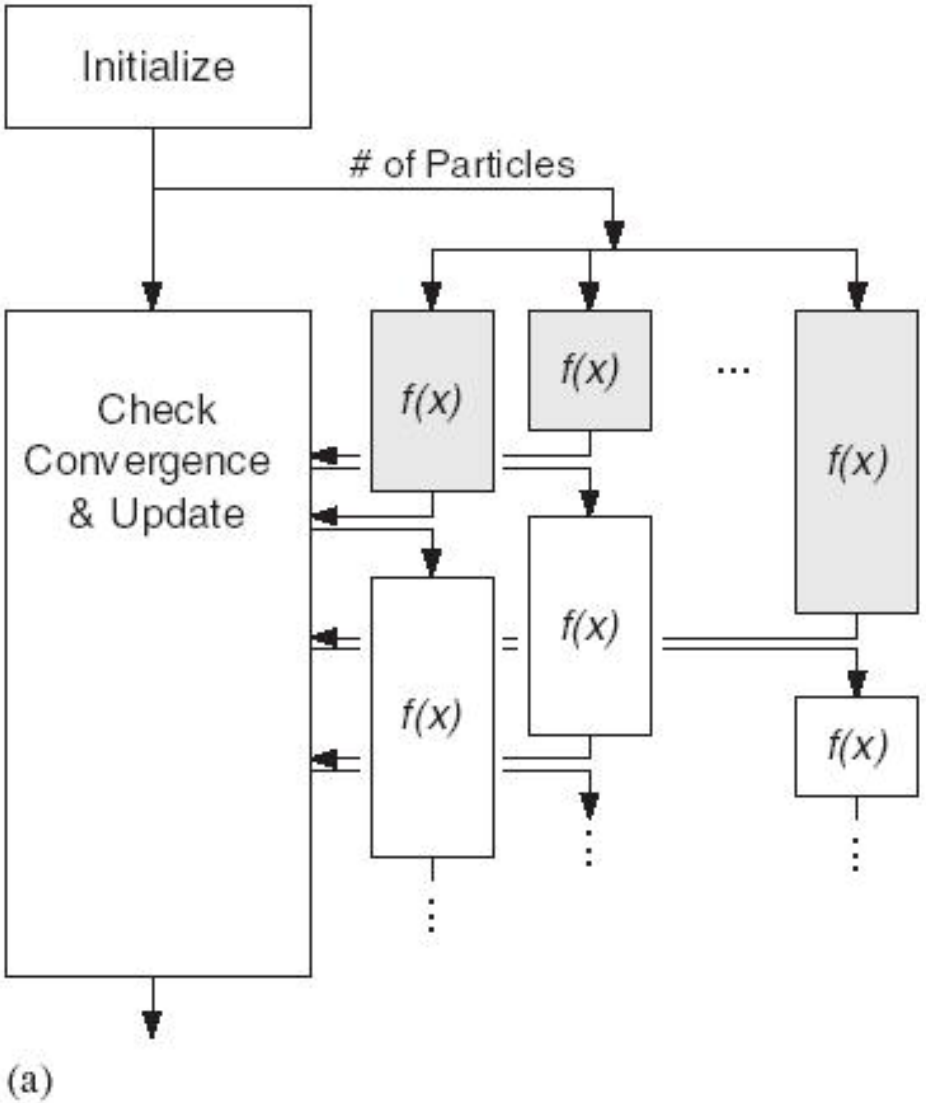


Flowchart of the particle swarm optimization algorithm

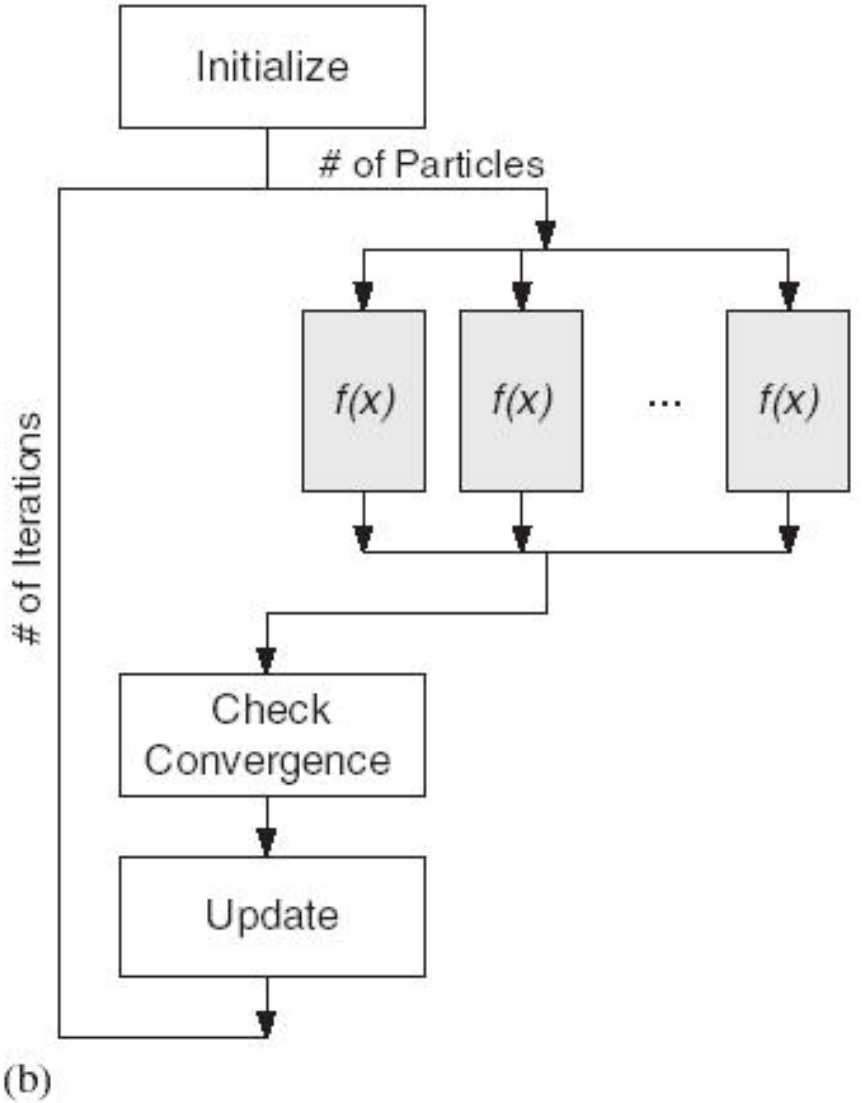
Particle
swarm
optimization

Метод
роя
частиц

Глобальная модель



Параллельный **асинхронный** PSO



Параллельный **синхронный** PSO

Особенности алгоритма

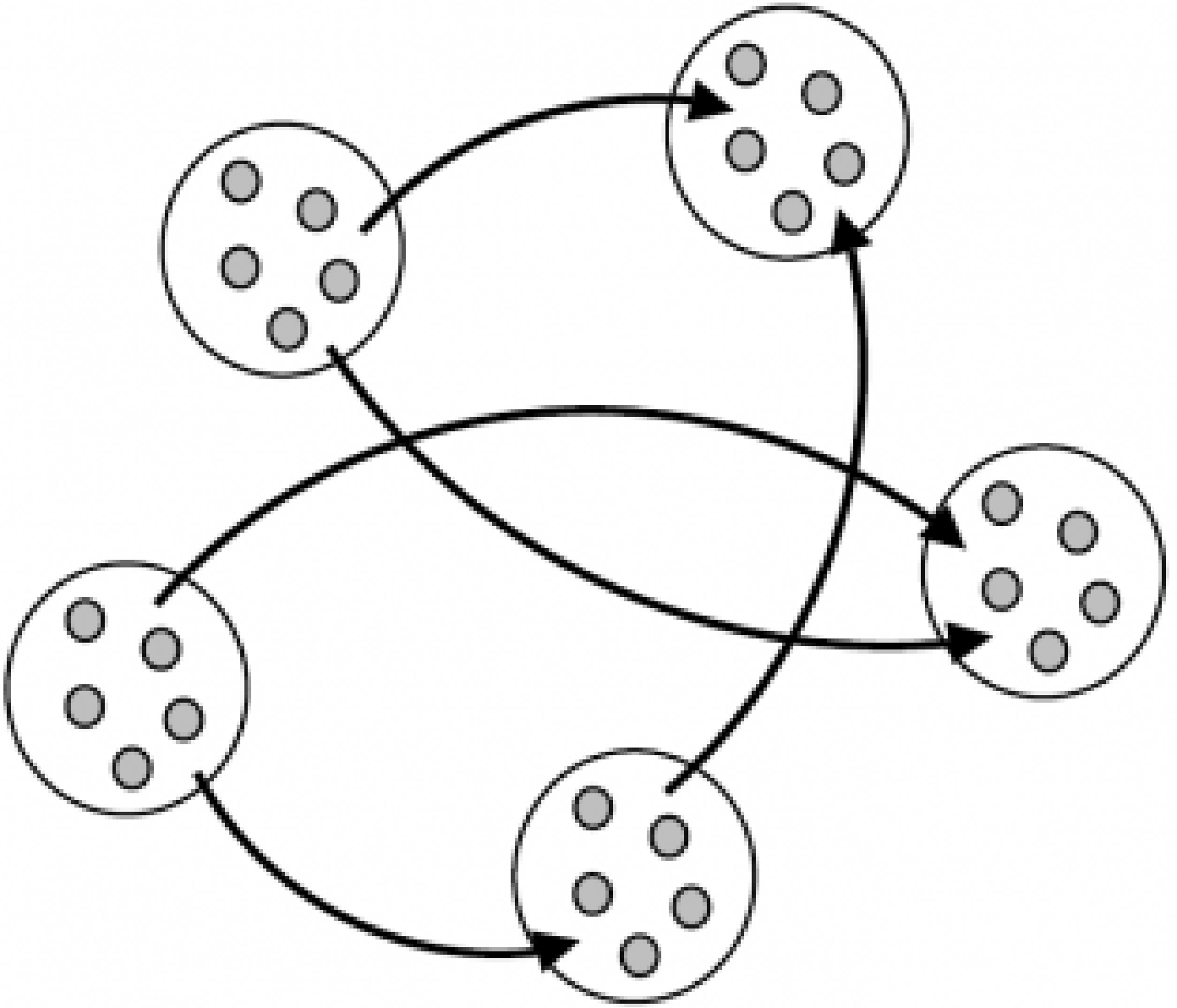
+ Возможность получения и использования глобальной информации о популяции (например информации о глобально лучшей частице)

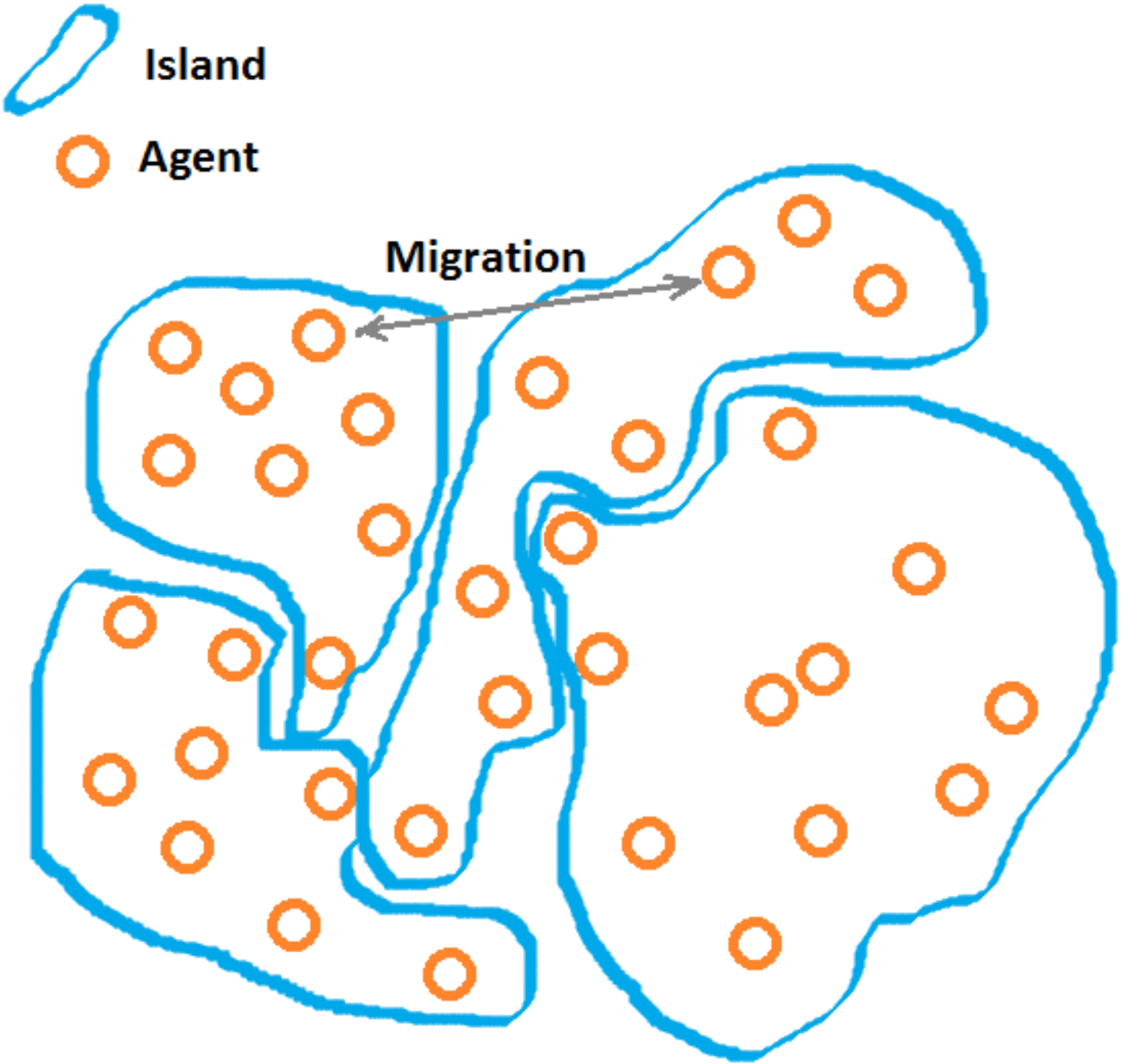
- Высокие накладные расходы на коммуникации

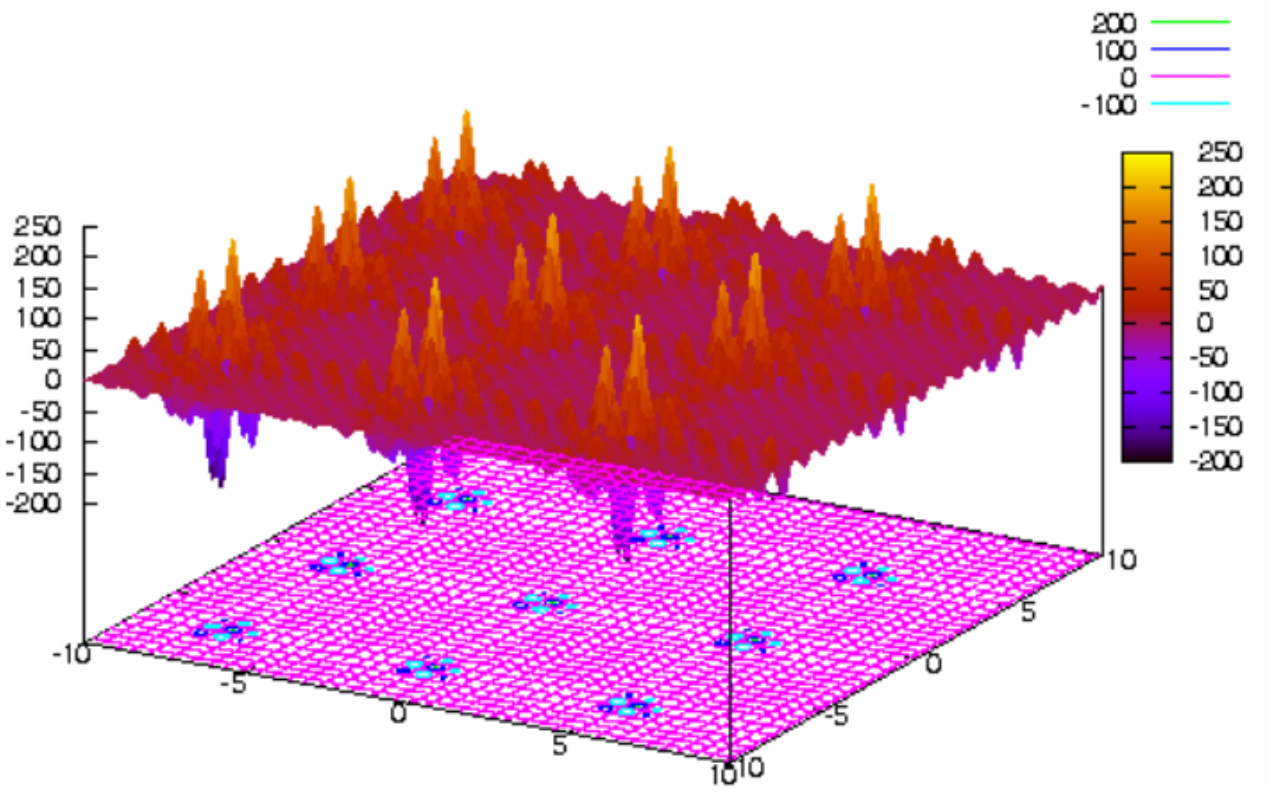
---

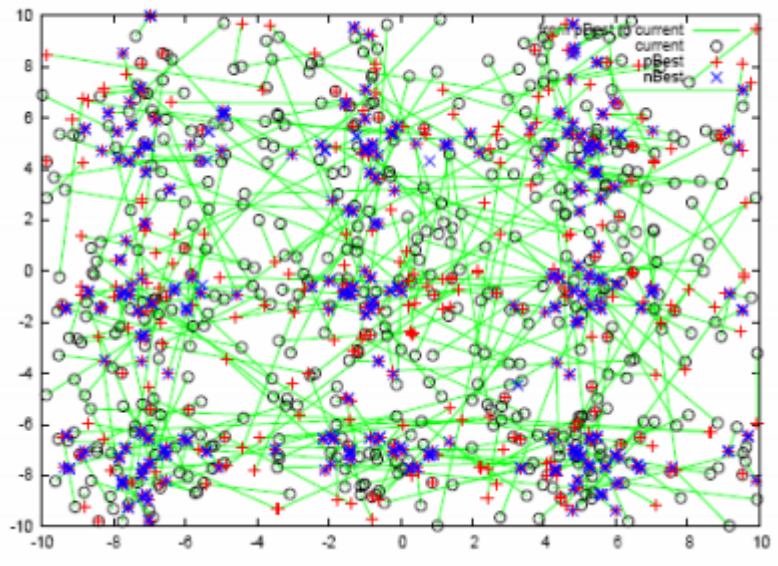
* Влияние дисперсии вычислительной сложности фитнесс функции и гомогенности вычислительной системы на эффективность синхронного и асинхронного алгоритмов

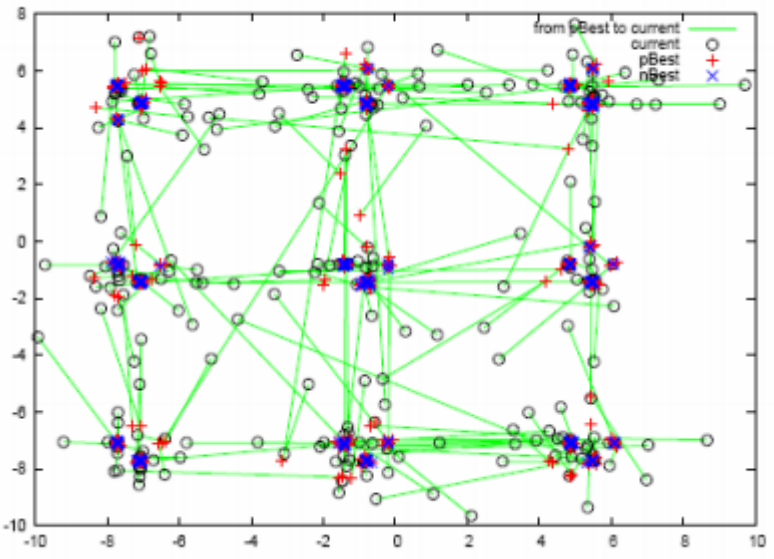
Островная модель



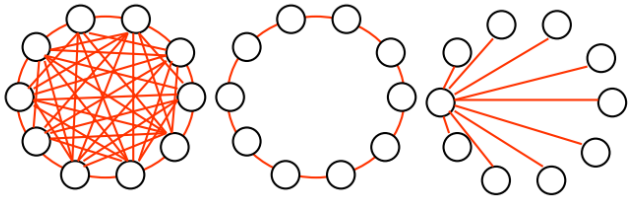

Island
Agent
Migration


200
100
0
-100
250
200
150
100
50
0
-50
-100
-150
-200
-10
-5
0
5
10


from pBest to current
current
pBest
nBest

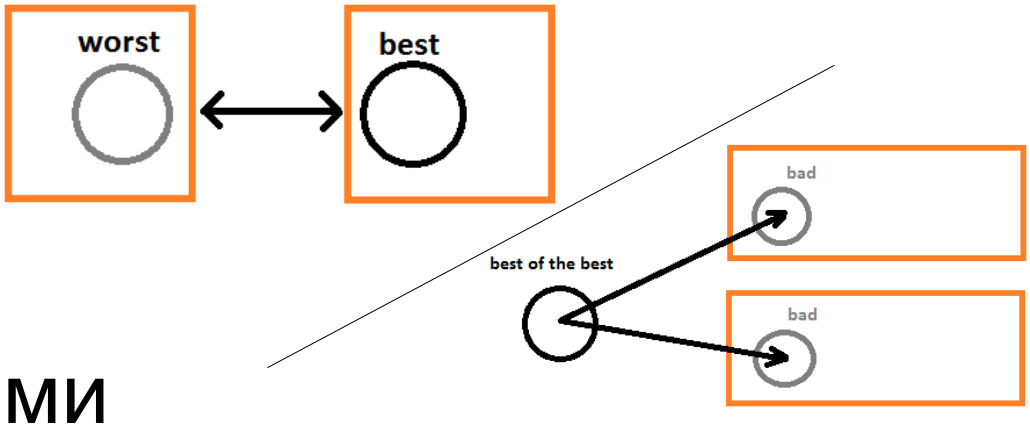

from pBest to current
current
pBest
nBest

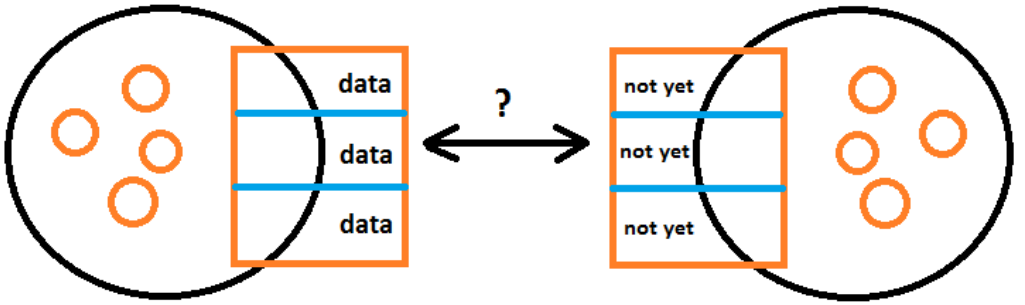
Свойства островной модели



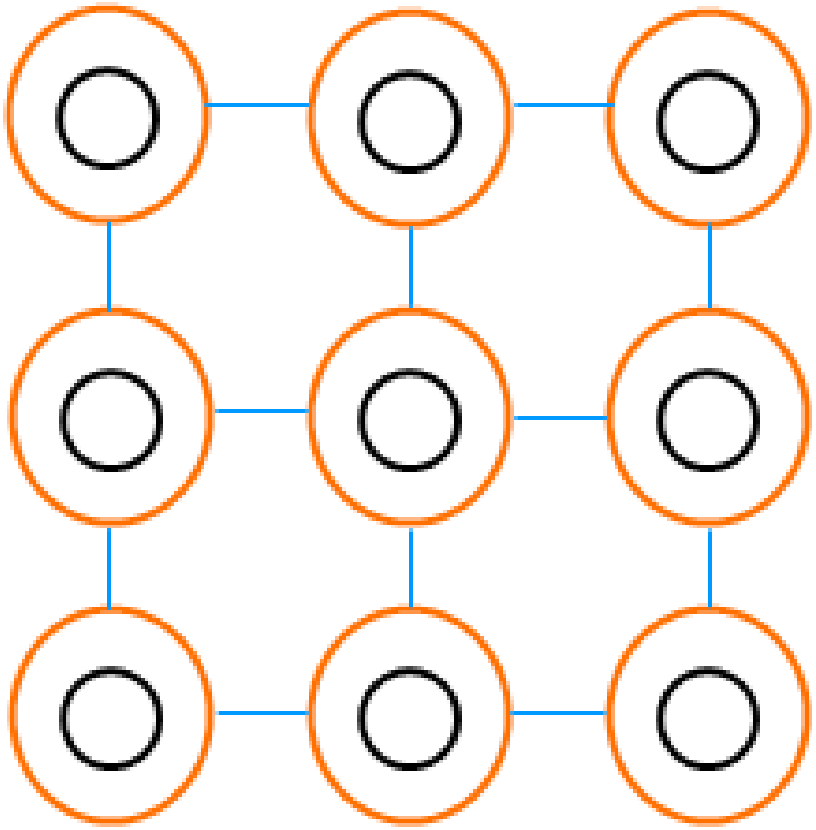
- Топология связей островов
- Длительность сезона t^{mig}

- Стратегия миграции
  - Перемещением
  - Репликацией
- Режим обмена агентами
  - Синхронный
  - Асинхронный

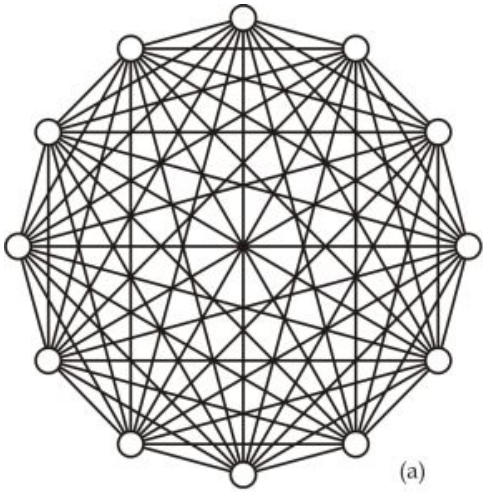


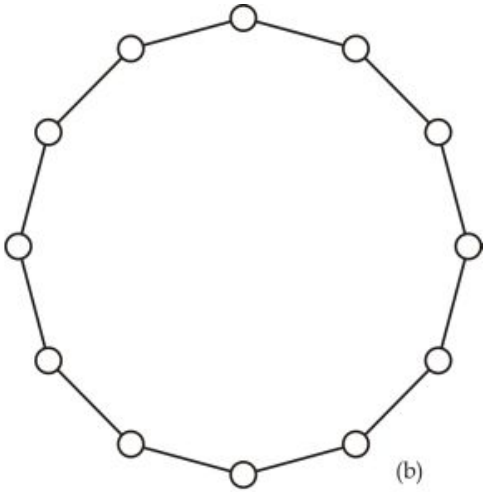


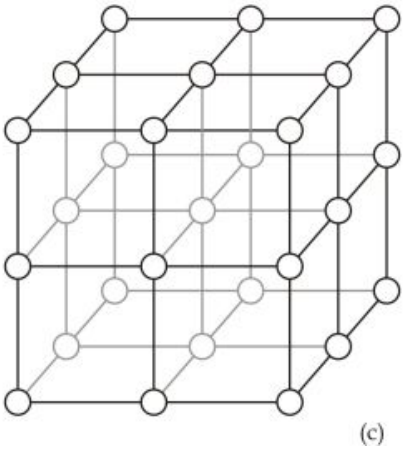
Диффузная модель

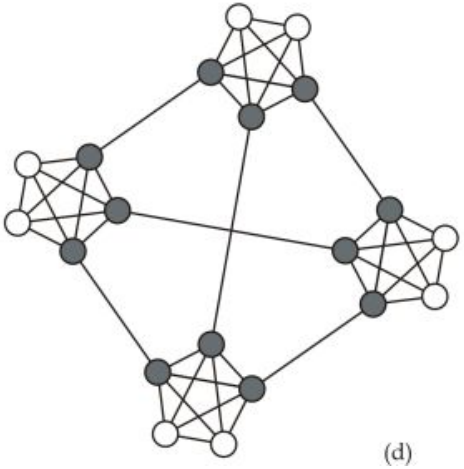


Агент на острове







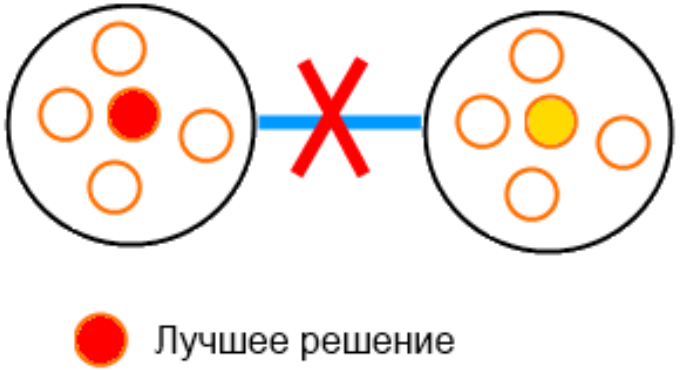


Особенности алгоритма

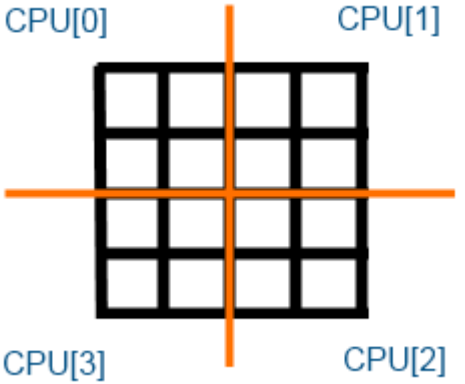
- Высокая производительность только при невысокой связанности графа связей агентов и / или высокой вычислительной сложности фитнес - функции

Другие модели

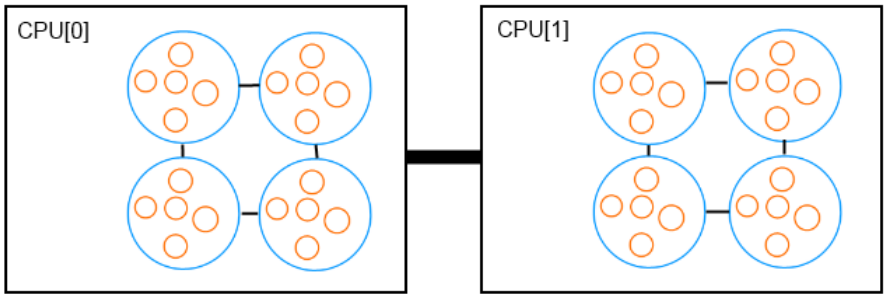
Некоммутирующая мультипопуляция



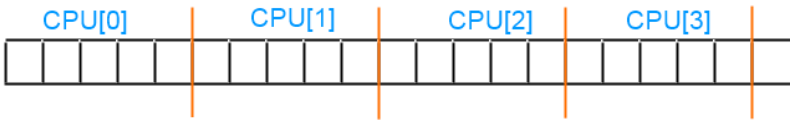
Пространственно – вложенная модель параллелизма



Коэволюционная модель

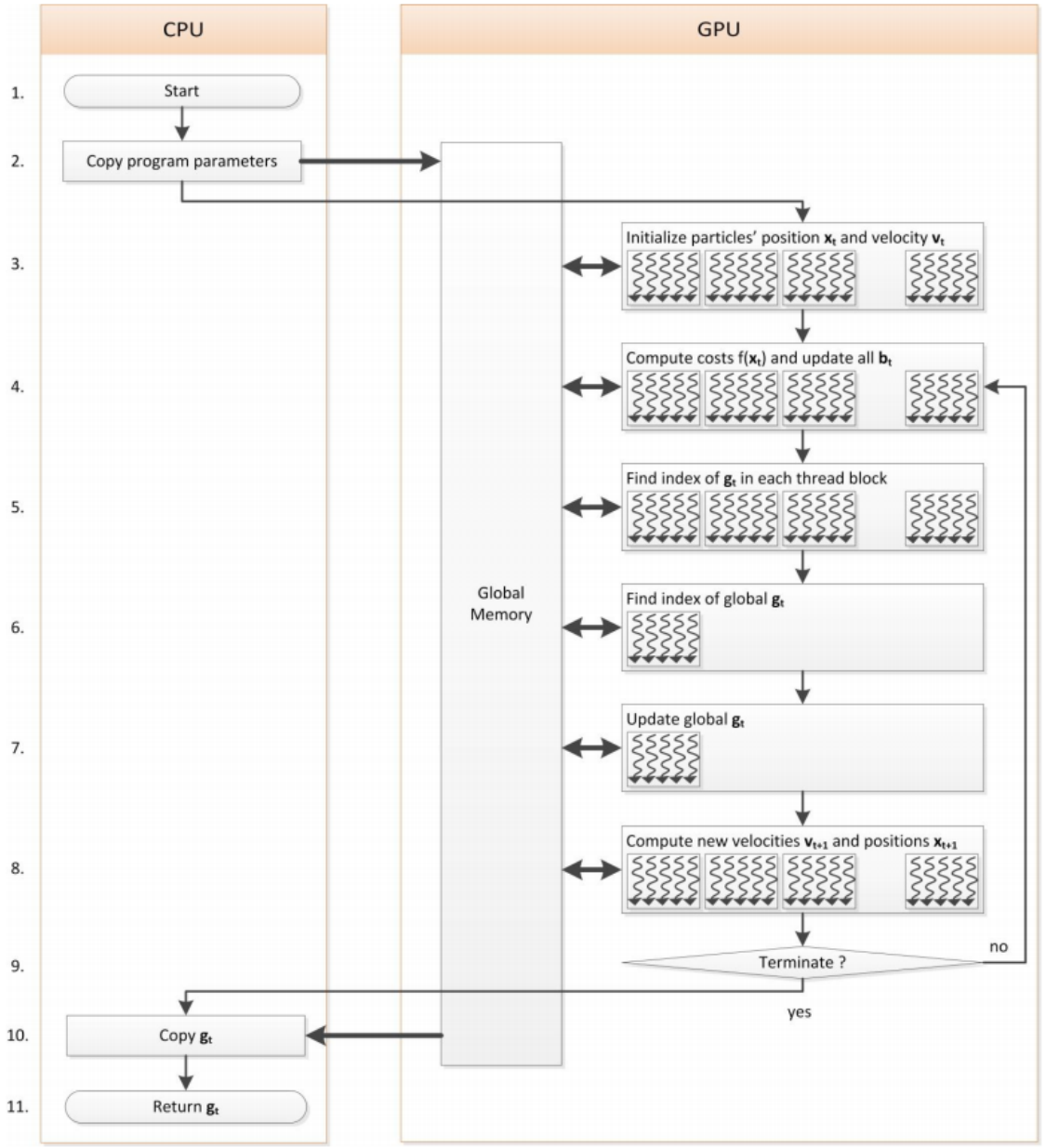


Параллелизм расщиплением



Примеры параллельного решения задач оптимизации

GPU Island PSO



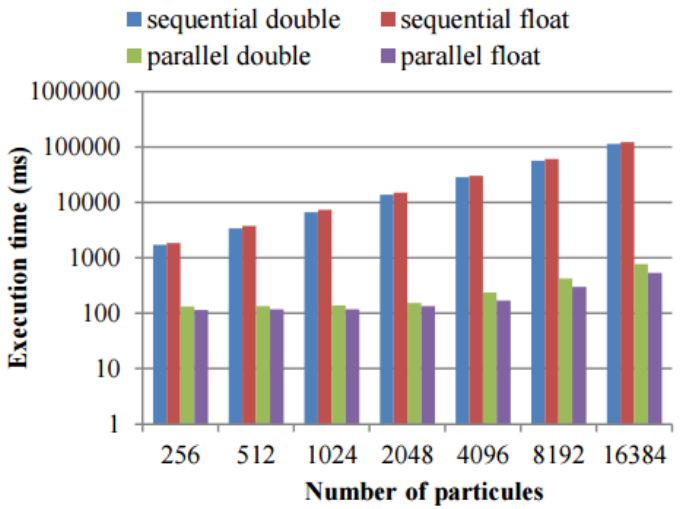


Fig. 8. Execution time of our sequential and parallel PSO for different precisions and number of particles (20 dim, 2000 iterations, avg of 20 trials)

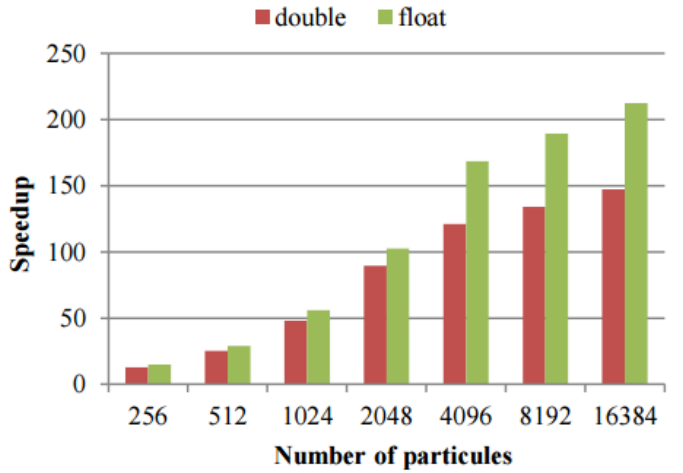


Fig. 9. Speedup of our parallel PSO for different precisions and number of particles (20 dimensions, 2000 iterations, average of 20 trials)

GIPSO vs PSO

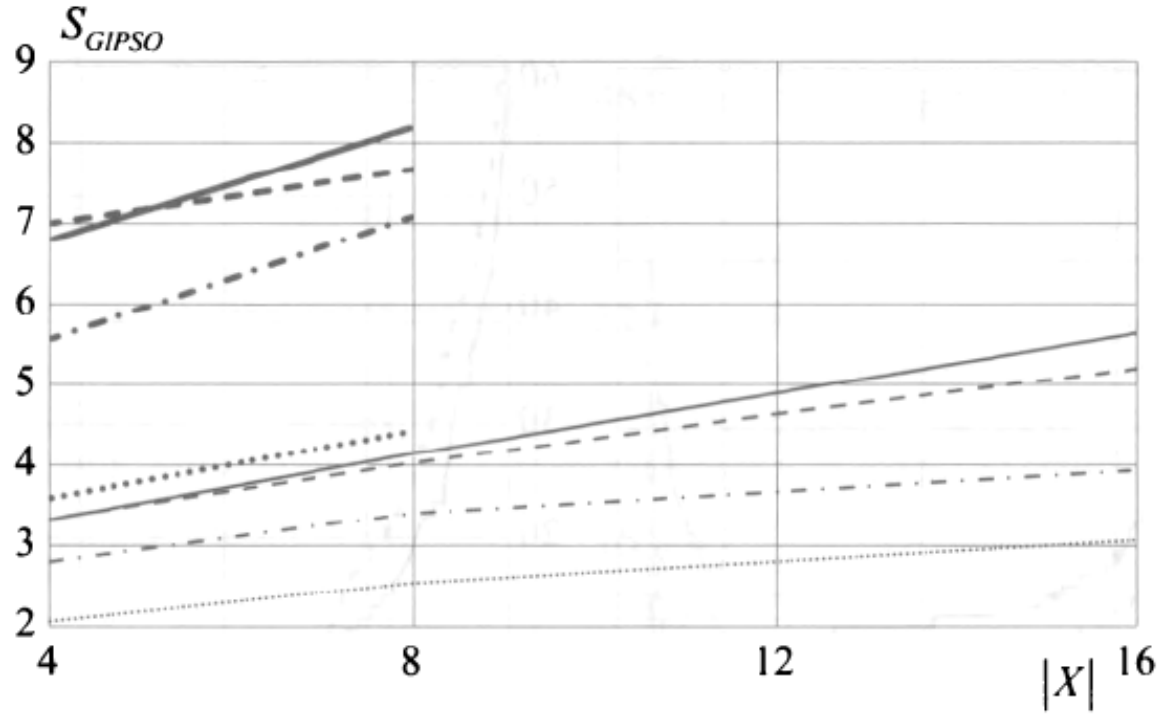
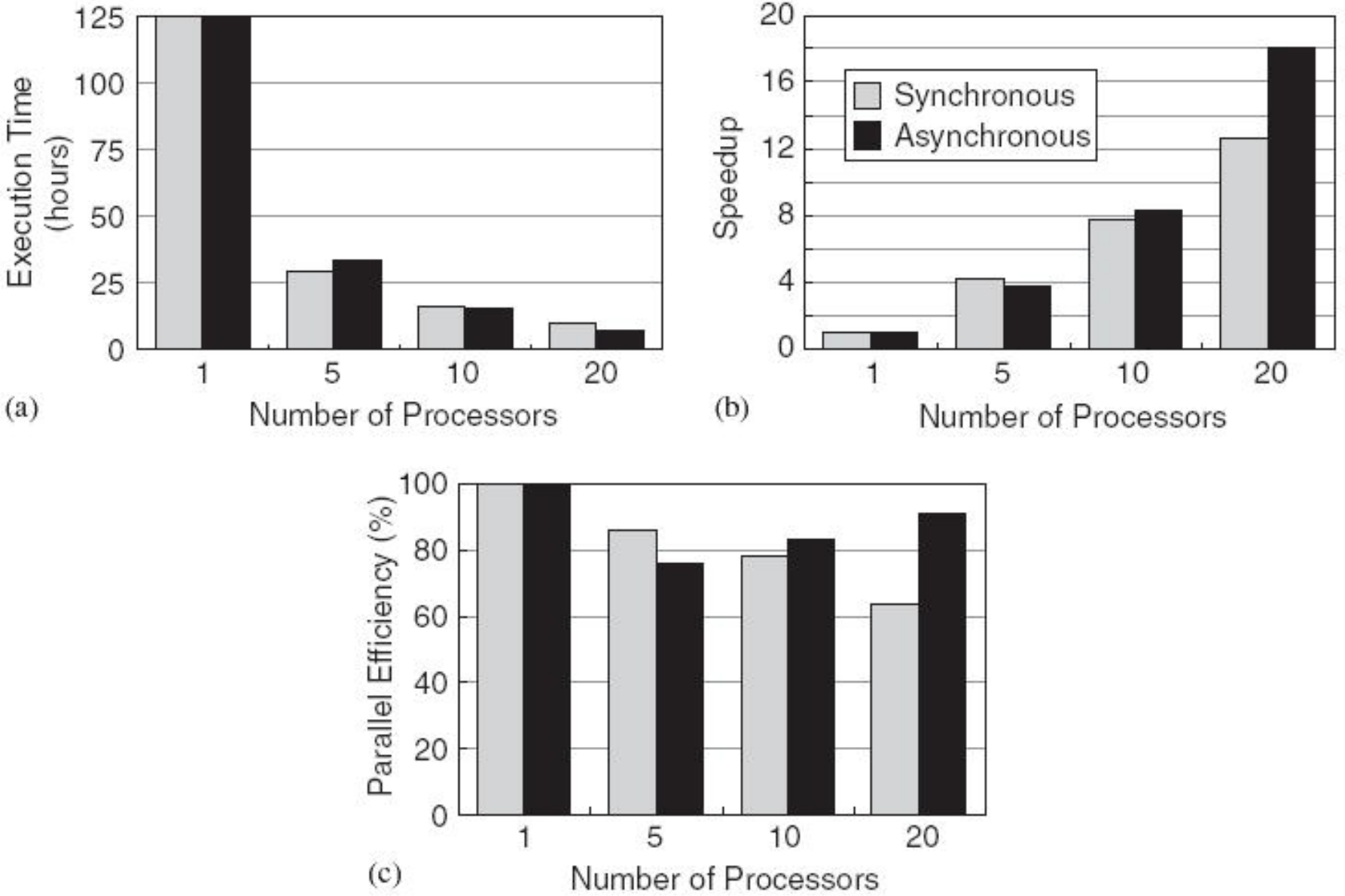


Рис. 9.16. Ускорение алгоритма *GIPSO*, ядро *PsoV2*; $|\boldsymbol{S}| = 4$; $|S_\Sigma| = 32$: —— функция Экли, – – – функция Растригина, ····· функция Розенброка, -·-·- сферическая функция; $|S_\Sigma| = 64$: —— функция Экли, – – – функция Растригина, ····· функция Розенброка, -·-·- сферическая функция

Оценка производительности. Обозначим τ_{GIPSO}, τ_{PSO} времена решения задачи алгоритмами *GIPSO* и *PSO* соответственно. В качестве меры производительности алгоритма *GIPSO* используем ускорение

$$S_{GIPSO} = \frac{\tau_{PSO}}{\tau_{GIPSO}}. \qquad (9.9)$$

Sync vs Async



Multi-objective Optimization with PSO

Одно- и многороевый алгоритмы MOPSO

Алгоритм многоцелевой оптимизации роем частиц (*Multi-Objective Particle Swarm Optimization*, *MOPSO*) предложили Мостагхим (*S. Mostaghim*) и Тейч (*J. Teich*) в 2003 г. Алгоритм использует канонический алгоритм роя частиц и алгоритм динамического соседства (см. 8.3.2). Схема одной итерации алгоритма имеет следующий вид.

1) В соответствии с алгоритмом динамического соседства для каждой из частиц s_i популяции S находим глобально лучшую частицу s_i^{**}.

2) В соответствии с каноническим алгоритмом *PSO* перемещаем все частицы популяции на один шаг, используя в качестве локально лучшего вектора X_i^* последние сохраненные в архиве координаты частицы s_i, а в качестве глобально лучшего вектора X_i^{**} – координаты частицы s_i^{**}.

3) Если новое положение X_i' частицы s_i доминирует некоторое решение из архива Θ^X, то это решение заменяем решением X_i'. Если решение X_i' не сравнимо ни с одним из архивных решений, то добавляем его в архив и объявляем новым локально лучшим решением X_i^*. Соответствующим образом корректируем архив Θ^F.

Итерации в алгоритме *MOPSO* могут продолжаться до тех пор, пока архивы недоминируемых решений Θ^X, Θ^F не перестанут изменяться, либо до достижения заданного числа итераций.

Решение задачи Парето алгоритмом MOPSO

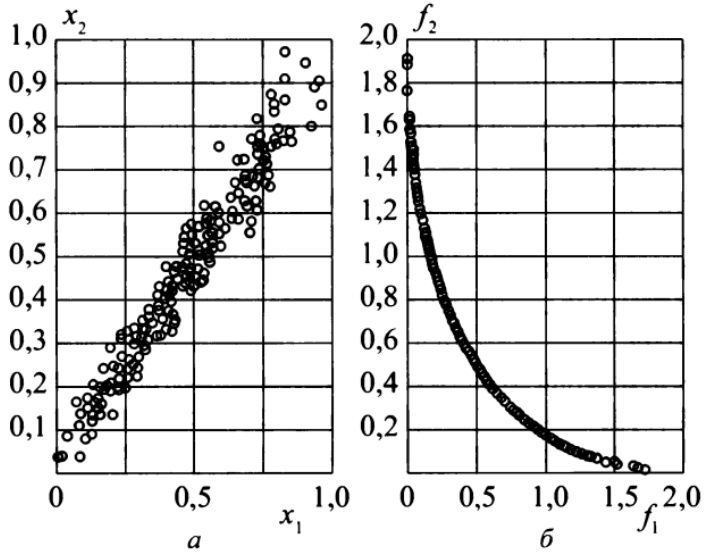


Рис. 8.28. Аппроксимация множества Парето (*а*) и фронта Парето (*б*), пример 8.1; алгоритм *MOPSO*

Таблица 9.1 **Эффективность распараллеливания алгоритма *MOPSO***

$\|S\|$	κ	τ_{CPU}, с	τ_{GPU}, с	S_{MOPSO}	$\|S\|$	κ	τ_{CPU}, с	τ_{GPU}, с	S_{MOPSO}
32	1	0,06	0,21	0,3	128	1	0,54	0,66	0,8
	10	0,30	0,27	1,1		10	2,50	0,80	3,1
	100	2,39	0,66	3,6		100	28,50	2,61	10,9
	1000	25,64	4,34	5,9		1000	219,03	15,41	14,2
64	1	0,15	0,36	0,4	256	1	3,36	1,23	2,7
	10	0,64	0,42	1,5		10	12,44	1,50	8,3
	100	6,56	1,14	5,8		100	81,44	4,45	18,3
	1000	69,83	7,64	9,1		1000	690,44	29,38	23,5